Neurocomputing 122 (2013) 375–385

Contents lists available at ScienceDirect

Neurocomputing

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

Models of performance of time series forecasters

Mario Graff [a,*], Hugo Jair Escalante [b], Jaime Cerda-Jacobo [a], Alberto Avalos Gonzalez [a]

[a] *División de Estudios de Postgrado, Facultad de Ingeniería Eléctrica, Universidad Michoacana de San Nicolás de Hidalgo, Mexico*
[b] *Computer Science Department, Instituto Nacional de Astrofísica, Óptica y Electrónica, Mexico*

ARTICLE INFO

Article history:
Received 15 February 2013
Received in revised form
22 May 2013
Accepted 30 May 2013
Communicated by P. Zhang
Available online 1 July 2013

Keywords:
Algorithm selection problem
Time series
Time series features
Forecasting
Performance prediction

ABSTRACT

One of the first steps when approaching any machine learning task is to select, among all the available procedures, which one is the most adequate to solve a particular problem; in automated problem solving this is known as the *algorithm selection problem*. Of course, this problem is also present in the field of time series forecasting, there, one needs to select the forecaster that makes the most accurate predictions. Generally, this selection task is manually performed by analyzing the characteristics of the time series, thus relying on the expertise that one has on the available forecasters. In this paper, we propose an automatic procedure to choose a forecaster given a set of candidates, i.e., to solve the algorithm selection problem on this domain. To do so, we follow two paths. Firstly, we propose to model the performance of the forecasters using a linear combination of features that were previously used to assess the problem difficulty of evolutionary algorithms, together with a set of features we propose in this paper. Then, this model is used to predict the performance of the forecasters and based on these predictions the forecaster is selected. Our second approach is to treat this algorithm selection process as a classification task where the descriptors of each time series are the proposed features. To show the capabilities of our approach, we test the forecasters on the time series of the M1 and M3 time series competitions and used three different forecasters. In all the cases tested, our proposals outperform the performance of the three forecasters indicating the viability of our approach.

1. Introduction

The process of solving a problem often starts by looking at the different procedures available to do the job. Then, after an analysis of these procedures and the problem, one decides which one of them to apply. As expected, this decision process is inferred by prior knowledge. That is, an algorithm is selected based on the experience one has in relation to the problem and/or the algorithm. It is also common that the algorithm is chosen from procedures that have shown some success in solving “similar” problems. This practice is also common in the field of time series forecasting, see for example [30,43] where a number of different forecasting techniques are analyzed and their strengths and weakness are highlighted in order to help in this selection.

In automated problem solving, this selection problem is known as the *algorithm selection problem* [39], the idea is to choose among a set of different algorithms the one which would perform the best, given a particular problem. Closely related to the algorithm selection problem are the algorithm portfolios [17,36,25–29, 46–48]. An algorithm portfolio is a collection of algorithms that are run in parallel or sequentially in order to solve a particular problem. The idea is that the portfolio outperforms the performance of any of the algorithms composing it.

One of the first attempts to automate the process of selecting and/or tuning a forecaster was presented in [9]. The authors proposed a number of rules that were used to combine the forecasts performed by four techniques: random walk, linear regression, Holt's linear exponential smoothing [16], and Brown's linear exponential smoothing [8]. This work was then extended and improved in [1,2], where the authors reduced the number of rules and they proposed an automatic process to identify the most prominent time series features.

To the best of our knowledge, the first fully-automated selection procedure was presented in [5] (and later extended in [6]). In these works an induction-based expert system was used to select the most promising forecasting technique based on time series features. Almost in the same period, a discriminant analysis was used to select the most appropriate forecasting technique (see [40]). It is interesting to note that neither of them used either terms meta-learning or algorithm selection problem (a common issue across different domains that is further discussed in [42]).

More recently, this problem has been addressed by different researchers (see [35,38,44,24]) using a variety of machine learning techniques—including linear combination of features, and decision trees, among others—and proposing novel time series characteristics

* Corresponding author. Tel.: +52 4433223500.
E-mail addresses: mgraffg@dep.fie.umich.mx, mgraffg@gmail.com (M. Graff), hugojair@inaoep.mx (H.J. Escalante), jcerda@umich.mx (J. Cerda-Jacobo), javalos@umich.mx (A. Avalos Gonzalez).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.05.035

that aim to enrich the set of features describing the time series. All these works have in common that all the features used to tackle the algorithm selection problem are based on time series characteristics. That is, these features are tailored specifically to describe time series, a few examples of these are: trend, seasonality, serial correlation, and periodicity, among others.

The main contributions of this manuscript are the proposal of a novel set of time series features, and the use of our previously developed indicators [34,15] on a completely different domain. Our previously developed indicators have successfully been applied in modeling the performance of Genetic Programming (GP) on different problem classes such as: symbolic regression of rational functions, and Boolean induction problems, among others.

The features proposed in this contribution complement (and are competitive with) previous works, being the main difference between them that our features are not problem specific. That is, our features can be easily applied to other domains being based on the notion of finite difference and the distance between a set of references. Nonetheless, this generality does not imply more complexity, as estimating our features involve only basic arithmetic operations.

Inspired by our previous experience in modeling the performance of GP, we decided to generate models of the performance of the time series forecasters using a linear combination of the proposed features. Then, this model is used to predict the performance of the forecasters and based on these predictions the best forecaster is selected.

In order to show that the proposed features are competitive with previous work, we also produce models of performance using the time series characteristics described by Wang et al. [44] and Lemke et al. [24]. These are the two most recent approaches that are closely related to our proposal, besides the features used in both works cover almost all of the time series features that have been used in the literature to solve the problem.

Whereas the linear combination approach resulted very competitive, a linear method might not be the best strategy to select the most prominent forecaster, neither to compare the features proposed here to the ones proposed by Wang et al. and Lemke et al. Therefore, in order to present the complete picture, we decided to test all these features using traditional classification techniques such as: neural networks, support vector machines and random forest, among others. Furthermore, we also tested these features using a novel technique that automatically selects very effective classification models, namely Particle Swarm Model Selection [14] (PSMS).

In order to show the capabilities of our approach, we decided to model three different forecasters implemented in the forecast package of R [20]. These algorithms are: the Exponential smoothing state space model (ETS) (see [21,18,19]), the Auto-Regressive Integrate Moving Average model (ARIMA) (see [41,12]), and the Exponential smoothing state space model with Box-Cox transformation, ARMA errors, Trend and Seasonal components (BATS) (see [10]). These forecasters were tested on the time series of the M1 and M3 competitions [31,32].

Finally, the novel features proposed in this contribution and the time series features used in [44,24] are compared using the algorithms and the time series aforementioned. The results show that our novel set of features are competitive, obtaining the best performance on two of the cases tested; the characteristics used by Wang et al. obtained the best performance in one case; and a combination of all the features described herein obtained the best performance in the remaining case.

The rest of the paper is organized as follows. In Section 2 the time series characteristics are described. Section 3 presents our modeling technique and the process to apply it. Section 4 presents the time series used to test our approach and the forecasters analyzed. The experimental results are shown in Section 5. The conclusions and some possible directions for future work are given in Section 6.

2. Characterizing univariate time series

In this section, we describe the different features used to characterize univariate time series. We start describing the time series characteristics proposed by Wang et al. and Lemke et al. in Sections 2.1 and 2.2, respectively. This is followed by a description of our previous models of performance (see Sections 2.3 and 2.4). Finally, Section 2.5 presents the novel time series features.

2.1. *Wang et al.'s time series characteristics*

Wang et al. [44] proposed a number of metrics to characterize univariate time series. These metrics were then used with self-organizing maps and decision trees to infer the relationship of these characteristics and the performance of the forecasters under study. The result of this work is a set of rules that can be used to decide which algorithm to use given the time series characteristics.

The first step in order to compute these characteristics is to perform a Box-Cox transformation [7] to the original time series. Let y be the time series and y_p be the time series after the Box-Cox transformation.[1] y_p is decomposed into its trend, seasonal, and remainder components (see [33]), i.e., $y_p = y_t + y_s + y_e$ where y_t correspond to the trend component, y_s is the seasonal component, and y_e is the remainder. Finally, one can remove from y_p the trend and seasonal components, i.e., $y_f = y_p - y_t - y_s$.

The original and decomposed series, i.e., y, y_f, y_t, and y_s, are used to compute thirteen characteristics, which are used to describe the time series by Wang et al. The first two are related with the seasonal and trend components. These are defined as: $w_1 = 1 - \sigma^2(y_f)/\sigma^2(y_p - y_s)$ and $w_2 = 1 - \sigma^2(y_f)/\sigma^2(y_p - y_t)$, where $\sigma^2(\cdot)$ is the variance. w_3 is the periodicity of $y_p - y_t$ which is computed using the autocorrelation. The next two features, w_4 and w_5, correspond to the serial correlation of y and y_f, respectively. w_6 and w_7 are the nonlinear autoregressive structure measured from y and y_f, respectively. w_8 and w_9 are the skew of y and y_f, respectively; and w_{10} and w_{11} are the kurtosis of y and y_f, respectively. The self-similarity or long-range dependence corresponds to w_{12} and it is measured using y_f. The last characteristic, i.e., w_{13}, is chaos which is computed using the Lyapunov exponent on y.

2.2. *Lemke et al.'s time series characteristics*

Lemke et al. [24] used different machine learning techniques such as: neural networks, decision trees, support vector machines, and zoomed ranking; to tackle the algorithm selection problem. These machine learning techniques were trained with 27 time series features. These features were categorized by Lemke et al. in four blocks.

1. The first block contains ten features based on traditional statistics. These are: standard deviation of detrended series, skewness of y, kurtosis of y, the length, ratio of the standard deviation of y and detrended series, Durbin–Watson statistic, turning points, step changes, non-linearity, and largest Lyapunov exponent.
2. The second block contains five features obtained on the frequency domain. These features correspond to the three biggest values of

[1] We used the BoxCox transformation implemented in [20]. λ is obtained using the loglik method with -1 and 1 as its limits.

the power spectrum frequencies, the biggest value of the power spectrum, and the number of peaks not lower than 60% of the maximum.
3. The third block has perhaps one of the most traditional analysis perform on the time series that is the auto-correlation coefficients, namely *acf* and *pacf*. Here, four features are obtained of the first two coefficients of *acf* and the *pacf*.
4. The last block is different from the previous ones in the sense that it does not include time series characteristics, instead, the features presented measure the diversity of the algorithms being analyzed on the time series under study. That is, this block somehow measures the diversity with respect to the performance of the forecasters. The first feature measures the difference between the error of each forecaster and the error of a forecast composed by the average forecast of all the algorithms. The second is the ratio between the previous values. In total, there are two features for each algorithm being modeled. The third and fourth features are the mean and standard deviation of the correlation coefficients of the forecast made by the algorithms. Finally, the fifth and sixth coefficients are the number of methods in the top performing cluster and the distance between the top performing cluster and the second best. These two latter values were not computed given that in our experiments there is only one cluster, that is, all the algorithms are in the same cluster and only one of them is used to forecast a given time series. In total this block contains eight different features, given that we are analyzing three algorithms.

2.3. Performance models of genetic programming

In [34], we proposed a model to estimate the performance of Genetic Programming (GP) [22,37]) and related techniques. GP is an evolutionary technique commonly used to evolve programs. That is, let f be a problem and Ω be a search space composed by candidate programs that solved f. Then, GP search into Ω trying to find the program that solves the best problem f. The quality of the solution delivered by a program is measured using a *fitness function* and the *fitness* of a program is the value assigned by the fitness function to that particular program.

The model uses the fact that GP searches in the space of programs and that each program has a fitness given f. The idea is that the performance of a GP system on problem f, i.e., $P(f)$ can be estimated using a set of reference programs (elements of Ω) and the fitness of these elements. In formulae, the model is defined as

$$P(f) \approx a_0 + \sum_{s \in \mathcal{S}} a_s \cdot d_s(f) \tag{1}$$

where $\mathcal{S}$ is the set of reference programs, $d_s(f)$ is the fitness of program s on problem f, and a_s are coefficients that need to be identified.

Eq. (1) can be computed given set $\mathcal{S}$ and d_s. In fact these two components can be seen as the characteristics of the time series. That is, given a particular problem f, one can compute the values of d_s for every algorithm in $\mathcal{S}$. In order to show how to create $\mathcal{S}$, let us remember that $\mathcal{S}$ is formed by the programs of the search space. In the case of time series forecasting, Ω is composed by forecasters. Under this circumstance, $\mathcal{S}$ is also composed by forecasters, then we decided to form $\mathcal{S}$ using the forecasters being modeled which are: BATS, ARIMA, and ETS. The last ingredient is to define d_s, that is, the fitness of the forecaster s on a particular problem. Here, d_s is just the error of the forecaster in its training phase, i.e., the error in the in-sample data.

2.4. Difficulty indicators models

Following with models developed to estimate the performance of GP, in [15], we proposed an improvement over our previous modeling technique. This improvement is based on a novel set of features for assessing problem difficulty for GP, these features are very general, essentially being based on the notion of finite difference. These models outperformed our previous models in two ways: firstly, they predict more accurately, and, secondly, the models obtained with these features are simpler having a less number of degrees of freedom than our previous approach.

Our difficulty indicators described in [15] are inspired by the discrete derivative of a function. The discrete derivative of a function f w.r.t. a variable x is defined as

$$\Delta_h f(x) = \frac{f(x+h) - f(x)}{h}, \tag{2}$$

where h is the step size. Normally, one tries to set h to the smallest possible value; however, counterintuitively, the difficulty indicators used are computed varying the value of h. These are defined as

$$\varrho_i(f) = \frac{1}{|I|} \sum_{\mathbf{x} \in I} |\Delta_i f(x)|, \tag{3}$$

where I contains all the input patters, and i is the value of derivative step h.

The features described in Eq. (3) can be easily computed with the help of an example. Firstly, let us represent a time series as a function f where $f(1)$ is the first measurement of the series, $f(2)$ is the second, and so on. Based on this notation, I is $1, 2, \ldots, n$ where n is the length of the series; and i, which correspond to the step in Eq. (2), can take the values from 1 to $n-1$. Now, let f be defined as: $f(1) = 12$, $f(2) = 2$, and $f(3) = -1$ then $\varrho_1(f) = |\Delta_1 f(1)| + |\Delta_1 f(2)|$, $\Delta_1 f(1) = f(2) - f(1) = -10$, $\Delta_1 f(2) = f(3) - f(2) = -3$, and this yields to $\varrho_1(f) = \frac{13}{3}$.

The second difficulty indicator proposed in [15] is more related to the discrete derivative (Eq. (2)) of function f with the only difference that it was decided to use the absolute value. In formulae, these indicators are defined as follows:

$$\varsigma_i(f) = \frac{1}{|I|} \sum_{x \in I} \left| \Delta_h f^{(i)}(x) \right|, \tag{4}$$

where $f^{(i)}(x)$ is the ith order derivative of function f w.r.t. a variable x, $I = 1, 2, \ldots, n$, and h which was set to 1 in [15] is the step size.

The performance model described in [15] was a linear combination of the difficulty indicators presented in Eqs. (3) and (4).

It is important to note that although Eq. (1) and the difficulty indicators shown in Eqs. (3) and (4) have been tested previously to model the performance of GP, this is the first time these features are tested to model the performance of a forecaster. The difference is that previously, on GP, one is measuring the capabilities of the algorithms to learn a particular function, on other words, the performance of the algorithms is computed using the in-sample data. On the other hand, in this contribution, the performance of the forecasters is measured with the out-sample data, that is, we are modeling the generality of the forecasters.

2.5. Novel time series characteristics

We start proposing new time series features by extending the difficulty indicators previously introduced. Looking at Eq. (4), one can realize that the only reason to set $h = 1$ is to follow closely the concept of discrete derivative; however, the use of absolute value is not used at all in the computation of discrete derivative. Consequently, it is reasonable to investigate whether different values of h would produce feasible indicators. That is, our new set of features is obtained when h is varied from 2 to $n-1$ in Eq. (4). In order to make this explicit and to combine this extension with the original formulae, we decide to introduce in ς a superindex h to indicate the size of the step, i.e., $\varsigma_i^h = 1/|I| \sum_{x \in I} |\Delta_h f^{(i)}(x)|$, where i is the ith order derivative and h is the step used to compute it. It can be observed that when $i=1$ and h is varied from 1 to $n-1$ in ς_i^h one is effectively computing all the indicators computed in Eq. (3), i.e., $\varsigma_1^j \equiv \varrho_j$. Given this equivalence, we substitute all ϱ terms with ς.

Our second set of new time series features follows a completely different approach. Firstly, these are not extensions of previous works, and secondly they are in close relation with the forecasting problem. In order to explain them, let us remember that the problem of time series forecasting is to predict a number of points that are unknown in the process of training the forecaster. In the cases analyzed here, the minimum number of points to forecast is six.

Ideally, the method used to forecast is the one that is more related with the generating model of the particular time series. That is, one desires that the model used to forecast is the one that generates the particular time series. Clearly, this is not possible; however, the expectation is that the forecaster used performs the most accurate predictions.

Under this assumption, it is reasonable to imagine that if one forecasts n points with this ideal forecaster, these points would have similar characteristics that the points used to train the forecaster, namely in-sample points. That is, the values of any of the time series characteristics mentioned previously would be equivalent whether these values are computed using the in-sample or the predicted values of the series.

Our last set of features is inspired by this observation. Firstly, we compute all the time series characteristics proposed by [44,24] and the difficulty indicators (i.e., ς) using the in-sample data. Secondly, we used the forecaster to predict as many points as required by the competition being tested. With these predictions, we compute the time series features previously computed with the in-sample data. The final step is to measure the similarity between the features computed with in and out-sample data. We divided the features based on the study that proposed them, and, then, we measure the similarity between them. That is, one similarity measurement is obtained of the features proposed by Wang et al., another from the characteristics proposed by Lemke et al., and another from ς. In all the cases the symmetric mean absolute percentage error is used to measure the similarity between the features.

To sum up, in this section, we have presented different features that can be used to characterize univariate time series. We started describing the thirteen time series characteristics proposed by Wang et al.. Then, we described the twenty seven features proposed by Lemke et al.. Next, we presented our previous modeling techniques, namely the performance models and the difficulty indicators models. In the last section, we describe our novel features, being the first of them an extension of our difficulty indicators and the second set of features is based on the measuring of the difference between the time series characteristics computed using the in-sample and the out-sample data.

It is interesting to note that all of the features described here can be organized using the categories described by Lemke et al.': The features proposed by Wang et al. can be classified in the first block. That is, these features are computed using traditional statistics. The features used in the performance models can be set in the fourth block. That is, these features somehow are measuring the difference with respect to performance that the algorithms involved in the study have. The difficulty indicators correspond to the first block. Also in this block we can find our extension of the difficulty indicators. Our latest features are harder to categorize under this scheme. This features measure somehow the behavior of the algorithms but not exactly the performance. Furthermore, they use traditional statistics to compute them, so we would set them in the middle of the first and fourth block defined by Lemke et al.

3. Modeling forecasters using time series features

So far, we have described different the time series characteristics, proposed in previous research studies, our previous modeling techniques, and proposed novel time series features. At this point, we are in the position to start describing how these features are used to tackle the algorithm selection problem. As we have described previously, we decide to create models of performance using a linear combination of the time series features. The reason behind using a linear combination is the success of our previous models and the number of algorithms portfolios that follow this approach (see [25–29]).

Our first model is shown in Eq. (1), where $\mathcal{S} = \{\text{BATS}, \text{ARIMA}, \text{ETS}\}$ and d_s is a normalized version of the mean absolute error (NMAE) (see Section 5).

Our second model of performance is based on the difficulty indicators, i.e., ς. As before, we decided to linearly combine these indicators. In formulae, the model of performance is defined as

$$P(f) \approx a_0 + \sum_h^n \sum_i^{n/h} a_i^h \cdot \varsigma_i^h(f). \tag{5}$$

The maximum values of h and i are specified in Eq. (5) where n is the length of the time series; however, in this contribution we decided to set the maximum value of h and i to eight. This leaves us with a total of twenty difficulty indicators.

Based on aforementioned motivation, we decided to combine the thirteen time series characteristics proposed by Wang et al. using a linear equation. In formulae, the model is defined as

$$P(f) \approx a_0 + \sum_i^{13} a_i \cdot w_i(f). \tag{6}$$

Similarly, the time series characteristics used by Lemke et al. are linearly combined. That is

$$P(f) \approx a_0 + \sum_i a_i \cdot l_i(f). \tag{7}$$

The next models of performance correspond to the three last indicators proposed in Section 2.5. We decided to create one model per indicator. All these models have the form

$$P(f) \approx a_0 + \sum_{c \in \mathcal{S}} a_s \cdot \text{SMAPE}(x_{\text{in}}, x_{\text{out}}(s)), \tag{8}$$

where x stands for Wang's features, Lemke's features, and ς; "in" and "out" refer to the data used to compute these features. S are the set of forecasters being modeled, that is, BATS, ARIMA, and ETS.

In order to have a complete picture of different methodologies to model the performance of time series forecasters, we decided to include a model based on perhaps the oldest criteria used in model selection, this is, the Akaike information criterion (AIC) [3].[2] In formulae, this model is defined as

$$P(f) \approx a_0 + \sum_{c \in \mathcal{S}} a_c \text{AIC}(f, c), \tag{9}$$

where $\mathcal{S}$ is the different forecasters, and AIC(f, c) computes the AIC of forecaster c on the time series f.

We have mentioned previously that one of the contributions of this manuscript is the proposal of novel models based on features that have not been used before to characterize time series. Therefore, our next model uses as components all those features that have not been used to characterize time series. These are the components of Eqs. (1), (5), (9), and (8) (using only ς). In formulae,

[2] A recent approach of the use of AIC on radial basic function network on the problem of chaotic time series is described in [49].

this model is defined as

$$P(f) \approx a_0 + \sum_{s \in \mathcal{S}} a_s \cdot d_s(f) + \sum_{h}^{n} \sum_{i}^{n/h} a_i^h \cdot \varsigma_i^h(f) + \sum_{c \in \mathcal{S}} a_c \text{AIC}(f, c) + \sum_{s \in \mathcal{S}} a_s \cdot \text{SMAPE}(\varsigma_{\text{in}}, \varsigma_{\text{out}}(s)) \quad (10)$$

We have introduced a number of models; however, one may wonder what would be the quality of a model composed by all the features and characteristics described so far. In fact, this our last model, a model that is a mixture of all other models and is composed by all the features describe here in. In formulae, the model is defined as

$$\begin{aligned} P(f) \approx\, & a_0 + \sum_{s \in \mathcal{S}} a_s \cdot d_s(f) + \sum_{h}^{n} \sum_{i}^{n/h} a_i^h \cdot \varsigma_i^h(f) \\ & + \sum_{i}^{13} a_i \cdot w_i(f) + \sum_{i} a_i \cdot l_i(f) \\ & + \sum_{c \in \mathcal{S}} a_c \text{AIC}(f, c) + \sum_{s \in \mathcal{S}} a_s \cdot \text{SMAPE}(\varsigma_{\text{in}}, \varsigma_{\text{out}}(s)) \\ & + \sum_{s \in \mathcal{S}} a_s \cdot \text{SMAPE}(w_{\text{in}}, w_{\text{out}}(s)) + \sum_{s \in \mathcal{S}} a_s \cdot \text{SMAPE}(l_{\text{in}}, l_{\text{out}}(s)). \end{aligned} \quad (11)$$

In summary, in this section we have described the models of performance of time series forecasters (see Eqs. (1), (5), (6), (8), (9), (10), and (11)). In order to instantiate these models one needs to identify the coefficients a_i, below we describe the procedure used to instantiate the models.

3.1. Model identification

In order to identify the coefficients a_i, we need a training set T of pairs $(f, P(f))$, where f is the time series under study and $P(f)$ is the performance of the forecaster being modeled on f. Note that measuring the performance of an algorithm (i.e., $P(f)$) is a time consuming task that sometimes limits the methodologies – including this contribution – used to solve the algorithm selection problem; however, for the algorithms being tested this is not a concern. In addition to this, as specified by other researchers (see [42]), the collection of problem instances must have variable complexity. In the cases tested here, these differences in complexity were identified by the organizers of the competitions.

Now that, T has been defined, the coefficients a_i can be identified using ordinary least squares (OLS). Let us illustrate this procedure by identifying the coefficients of Eq. (6). Let $\mathbf{b} = (P(f_1), \ldots, P(f_m))$ be a vector containing the performance of the forecaster under study on all the series included in T, $\mathbf{a} = (a_1, \ldots, a_n)$ be a vector composed by the coefficients needed to identify, and m is cardinality of T. Under these circumstances, in order to identify $\mathbf{a}$, one needs to solve the equation $\mathbf{Wa} = \mathbf{p}$, where $\mathbf{W}$ is

$$\mathbf{W} = \begin{pmatrix} 1 & w_1(f_1) & \ldots & w_{13}(f_1) \\ \vdots & \vdots & \ldots & \vdots \\ 1 & w_1(f_m) & \ldots & w_{13}(f_m) \end{pmatrix}.$$

This equation can be solved using OLS, i.e., $\mathbf{a} = (\mathbf{W}'\mathbf{W})^{-1}\mathbf{W}'\mathbf{p}$. However, there are cases where the inclusion of all the covariates (i.e., columns of $\mathbf{W}$) have a detrimental impact on the generality of the model. That is, it may be the case where the inclusion of, for example, instance of w_3 is detrimental to the accuracy of the model, so in this case it would be more convenient not to include that factor. In order to automate this decision, we decided to solve $\mathbf{Wa} = \mathbf{p}$ using the Least Angle Regression (LARS) [13] and a cross-validation technique to decide how many covariates to include in the final model.

LARS works as follows. It starts by setting all of the coefficients $\mathbf{a}$ to zero and finds the covariate most correlated with the response $\mathbf{p}$. Then, it takes the largest possible step in the direction of this covariate until another covariate has as much correlation with the residual. At this point, LARS proceeds in the equiangular direction between the two predictors until a third variable has as much correlation with the residual as these two. The process continues until the last covariate is incorporated into the model. At this point, all $\mathbf{a}$ coefficients have been identified and these are equal to the ones obtained with OLS.

As it can be seen, LARS incorporates one variable at a time into the model. Therefore, one is free to stop this process when only k covariates have been selected. The idea is to select k of them in order to obtain the simpler and more general model. In this contribution, we decided to use a cross-validation technique to determine the value of k, i.e., the number of covariates included in the model.

The cross-validation is applied as follows. $\mathbf{W}$ and $\mathbf{p}$ are split row-wise into five matrices and vectors of equal size. Four of them are joined together and are used to produce a model (i.e., these are used to identified $\mathbf{a}$ using LARS), while the remaining pair is used to assess its generalization. Specifically, the remaining $\mathbf{W}$ and the $\mathbf{a}$ obtained in the previous step are used to estimate the response, namely $\hat{\mathbf{p}}$. This process is repeated 5 times, each time leaving out a different fifth of $\mathbf{W}$ and $\mathbf{p}$. At the end of this process, we have five $\hat{\mathbf{p}}$ that together can be compared against $\mathbf{p}$ to assess the generality of the model.

This cross-validation procedure is iteratively applied to the models produced by LARS for $k = 1, 2\ldots$ with the aim of identifying the value of k which provides the best generalization. We decided to measure the overall generalization via the Relative Squared Error (RSE) [4], which compares the performance of an algorithm as predicted by the model with the actual performance for all the problems in T. This is defined as follows: $\text{RSE} = \sum_i (P_i - \tilde{P}_i)^2 / \sum_i (P_i - \overline{P})^2$, where i ranges over a set of test problems used to evaluate the accuracy of a model, P_i is the average performance recorded for problem i, $\tilde{P}_i$ is the performance predicted by the model, and $\overline{P}$ is the average performance over all problems. The objective is to obtain values of RSE as close as possible to zero.[3]

4. Test problems and forecasters

In order to test our approach, we decided to use three different forecasters implemented in the forecast package of R [20]. These three algorithms are: the Exponential smoothing state space model (ETS) (see [21,18,19]); the Auto-Regressive Integrate Moving Average model (ARIMA) (see [41,12]); and the Exponential smoothing state space model with Box-Cox transformation, ARMA errors, Trend, and Seasonal components (BATS) (see [10]).

These algorithms were tested on 1001 time series corresponding to the M1 time series competition [31] and 3003 time series that correspond to the M3 time series competition [32]. In our experiments, we used M1 to train the models and then test them on M3 time series. For completion we also present the results when the forecasters are trained on M3 and then tested on M1. Furthermore, we decided to test these forecasters as if they were entering each competition. That is, these algorithms performed a *multi-step* forecasting as required by each competition. In addition to this, we also tested each algorithm in *one-step* forecasting. That is, after each forecast, we provide the algorithm with the true value of that measurement.

[3] A value of *rse* close to 1 means that the model is as good (or bad) at predicting performance differences as the mean. A value of *rse* less than 1 means that the model predicts better than the mean, while *rse* > 1 implies worse predictions than the mean.

Table 1
Average performance of forecaster on the different configuration tested.

M1			M3		
BATS	ARIMA	ETS	BATS	ARIMA	ETS
One-step forecasting					
0.1120	0.1446	0.1254	**0.1362**	0.1760	0.1535
Multi-step forecasting					
0.3721	0.3680	**0.3551**	0.4302	**0.4121**	0.4126

We measure the performance of these algorithms using a normalized version of the mean absolute error (NMAE) in these two time horizons, i.e., multi-step and one-step forecasting. NMAE is computed by first transforming the time series in the interval [0, 1], and then, on this domain, we applied the mean absolute error, and, finally, all the NMAE values above one are set to one.

Table 1 shows the average performance of the three forecasters tested. It can be seen from the table, that BATS has the best performance in one-step forecasting in both competitions. For the case of multi-step forecasting, ETS and ARIMA have the best performance for competition M1 and M3, respectively.

5. Results

In this section, we analyze the accuracy of the models described in Section 3. In Section 5.2 our experimental study is complemented with the use of traditional classification algorithms and the features described in Section 2 to tackle the algorithm selection problem.

5.1. Models of performance

Let us start describing the process to select a forecaster using models of performance. The first step is to instantiate a model for each of the forecasters, that is, we instantiate three models that correspond to BATS, ARIMA, and ETS. Then these models are used to predict the performance of each forecaster on each time series. These predictions are then used to select the forecaster with the best estimated performance.

The first analysis performed is to show which one of the models performs the most accurate predictions. Fig. 1 shows in percentage the number of times a model makes the most accurate prediction in each of the four cases tested. The models in the figure are arranged as follows: performance models (P) (Eq. (1)), difficulty indicators (DI) (Eq. (5)), Wang et al.'s model (W) (Eq. (6)), Lemke et al.'s model (L) (Eq. (7)), AIC model (AIC) (Eq. (9)), similarity model using DI (DDI) (Eq. (8)), similarity model using W (DW) (Eq. (8)), similarity model using L (DL) (Eq. (8)), mixture of models that do not use time series characteristics (NTSC) (Eq. (10)), and the mixture of all the models (All) (Eq. (11)).

From the figure, it can be observed that for the case of one-step forecasting the combination of all features (11) obtains the highest percentage, the second place is Wang et al.'s model and the third place is for NTSC model and the performance model (P), on the competition M1 and M3, respectively. In the case of multi-step forecasting, the best model is Wang et al.'s model, the second place is the model with all the features, and Lemke et al. in the M1 and M3 competitions, respectively. The third place is for NTSC and all the features in M1 and M3, respectively. Taking only in consideration the task performed, i.e., one-step or multi-step forecasting, it can be observed that the model that has the highest percentage in one-step forecasting is the combination of all of the features (Eq. (11)), followed by Wang et al.'s model (Eq. (6)), and in third place is the NTSC model (Eq. (10)). In the other case, the best and second best changed with respect to the previous case, that is, the best model is Wang et al.'s model and the second is the model with all the features.

At this point, we can conclude that our indicators alone are not performing well with respect to performing the most accurate prediction. However, the combination of our features obtains the third place on either the one-step or multi-step forecasting. This new features are only behind Wang et al.'s model.

Although Fig. 1 shows the quality of the models, it does not indicate whether these models are appropriate to solve the algorithm selection problem. That is, the realm of this problem is that the average performance obtained by selecting the forecaster (i.e., algorithm portfolio) must be better that the average performance of each of the forecasters (see Table 1 for the average performance of the forecasters). Of course, this has a limit and the limit is obtained when one has a perfect model. The performance of a perfect model can be seen in Table 4.

Table 2 shows the average performance of the portfolio for the different models described previously. Here, we also present the results on the cross-validation to indicate how the model would be selected based on this measurement. Let us remember that we have performed a cross-validation in the training set to optimize the number of covariates, then, we can use this information to solve the algorithm selection problem in the training set. This data is in the column label with CV. The column label VS corresponds to the data of the validation set, i.e., data no seen in the identification of the model. In the table, it is also shown the p value of the Kruskal Wallis one-way analysis of variance [23] to indicate whether the differences in performance are statistically significant.

Table 2 is divided horizontally into two blocks, the first block corresponds to the one-step forecasting and the second one is associated to the multi-step forecasting. From the table, it can be seen that the best average performance in the one-step forecasting is the model composed by all of the features, the exception is when the M3 competition is used as validation set; however, in this case is the second best. In this task, we can see that the p values are all below 0.05 indicating that there are statistically-significant differences in the performance among groups of characteristics. In order to identify whether the best algorithm is the one presenting this difference in performance, we compare the best algorithm against all others using a Wilcoxon signed-rank test [45]. In almost all of the cases the algorithm with all the features has a p value below 0.1, indicating, with a confidence of 90% that this algorithm has a better performance and it is statistically significant. The cases where the p values are high are when M1 is used as CV and the algorithm being compared is the model using Wang et al.'s features ($p=0.4814$) and the NTSC model ($p=0.4809$). Based on this, it is not surprising that the best model in the set M3 used as validation set is the model with Wang et al.'s features and the second best is the NTSC model. In this later case, the model with Wang et al.'s features is statistical better than the other model with a confidence interval of 95% as indicated by the p values of the Wilcoxon signed-rank test.

In the case of multi-step forecasting, the NTSC model (see Eq. (10)) is the one that obtained the lowest average performance in M1 used as cross-validation and M3 used as validation set. This model obtained the second best in the remaining cases. When M3 is used as cross-validation the best model is the combination of all features, and the best model on M1 used as validation set is the difference indicators model. It this task the p value of the Kruskal Wallis test indicates that there is not a statistically-significant difference in performance.

The average performance of algorithm portfolios shown in Table 2 complements the information presented in Fig. 1. Based on this information, it can be seen that the combination of all of

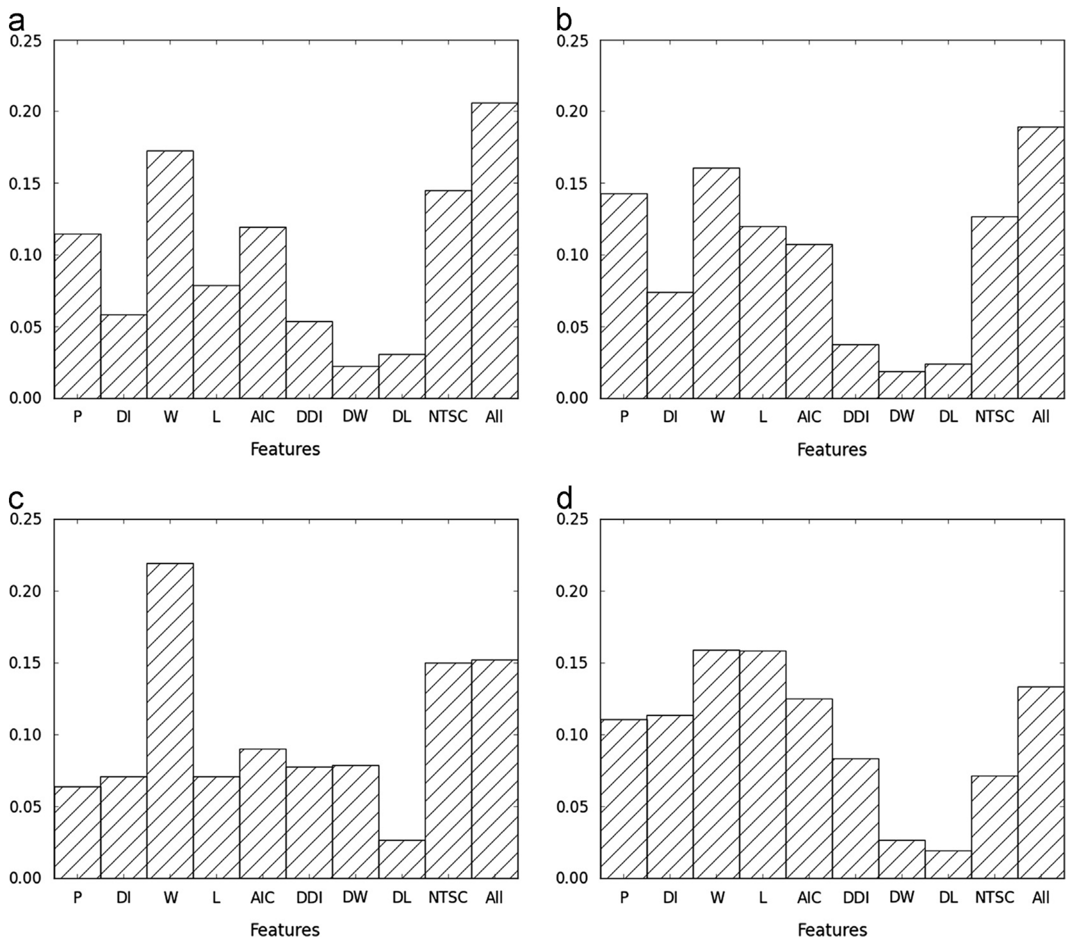

Fig. 1. Percentage of the number of times a model obtain the best prediction. (a) M1, one-step forecasting. (b) M3, one-step forecasting. (c) M1, multi-step forecasting (d) M3, multi-step forecasting.

our features (Eq. (10)) is very competitive with respect to the average performance of the portfolio and also by performing accurate predictions with respect to the other models. Wang et al.'s features are the best single features; however, they do not perform as well when the portfolio is tested on the multi-step case. It is not surprising that the model created with the combination of all features is the most consistent of all the models, this is an indication that LARS is selecting properly the most prominent features.

As mentioned before, the goal of an algorithm portfolio is to perform better that any of the algorithms composing the portfolio. The average performance presented in Table 1 and in column label VS in Table 2, can be used to test whether the models perform better that the single forecasters. On the case of one-step forecasting we can see that in the M1 competition almost all the models obtained better performance than the best forecaster. The two exceptions were the portfolios created with the difference indicators and the distance of Lemke et al.'s features. In the case of M3 competition, only half of the models obtained better average performance, these are Wang et al.'s features, distance of Lemke et al.'s features, AIC, NTSC, and the model with all the features.

In the case of multi-step forecasting, the number of models that obtained better performance is considerably lower, In the M1 competition none of the models obtained better average performance than the performance of the best forecaster. On the M3 case, only four modes have better average performance. These models are: the performance models, the difficulty indicators models, NTSC, and the model composed by all the features.

5.2. Classification algorithms

Results obtained in the previous section motivated us to look for alternative procedures (other than a linear combination) to select time series forecasters. This is in order to determine whether alternative methodologies can obtain similar or better results. Hence we performed experiments with other type of predictive models: classifiers. Opposed to regression models (like OLS and LARS), which perform predictions regarding a continuous variable, classification models aim to make predictions regarding a categorical variable [11]. The algorithm selection problem for time series forecasting can be naturally posed as a classification task: the problem of associating a time series with the forecaster that will result in the best performance.

Hence we face the classification problem of associating a time series with the algorithm that potentially will result in the best performance. In this case, there are three categories or classes that correspond to the three algorithms composing the portfolio. Each time series has a label indicating the algorithm that got the best performance (i.e., lowest NMAE) in that particular time series. Now that we have a label for each of the time series of our collection, we need a set of features describing a time series. Here is where the components used to create the models come in handy. These components can be used as the set of features, given

Table 2
Average performance of the algorithm portfolio when the selection is performed using the performance models. The last row in each group contains the p values of the Kruskal Wallis one-way analysis of variance to indicate whether the difference in performance is statistical significant.

Models	One-step forecasting			
	M1 (CV)	M3 (VS)	M3 (CV)	M1 (VS)
Performance (P)	0.1108	0.1417	0.1362	0.1116
Difficulty indicators (DI)	0.1106	0.1387	0.1344	0.1124
Wang et al.	0.1071	**0.1330**	0.1324	0.1064
Lemke et al.	0.1121	0.1420	0.1349	0.1115
Distance DI (DDI)	0.1115	0.1365	0.1361	0.1119
Distance Wang	0.1120	0.1363	0.1362	0.1118
Distance Lemke	0.1120	0.1362	0.1357	0.1121
AIC	0.1065	0.1342	0.1338	0.1053
NTSC	0.1064	0.1357	0.1327	0.1079
All	**0.1060**	0.1334	**0.1289**	**0.1039**
Kruskal Wallis' p values	0.0491	0.0004	0.0013	0.0197
Models	Multi-step forecasting			
	M1 (CV)	M3 (VS)	M3 (CV)	M1 (VS)
Performance (P)	0.3589	0.4106	0.4088	0.3593
Difficulty indicators (DI)	0.3648	0.4108	0.4091	**0.3576**
Wang et al.	0.3641	0.4144	0.4061	0.3615
Lemke et al.	0.3611	0.4136	0.4069	0.3616
Distance DI (DDI)	0.3571	0.4135	0.4120	0.3626
Distance Wang	0.3555	0.4125	0.4160	0.3681
Distance Lemke	0.3548	0.4123	0.4160	0.3669
AIC	0.3549	0.4142	0.4092	0.3642
NTSC	**0.3540**	**0.4081**	0.4056	0.3579
All	0.3541	0.4104	**0.4027**	0.3599
Kruskal Wallis' p values	0.9862	0.9910	0.5987	0.9970

that we already now that are related with the performance of the algorithm and as consequence with the hardness of the time series. So, each time series f is represented by a vector composed by the values of the components of Eq. (10) or (11), depending on whether the classifications are trained with the features that are no used to characterize time series or with all the features described, respectively.

As with the linear models, we use a training set of labeled time series to build the classification models. Next the classifier is used to make predictions for the test time series. For our experiments we considered 8 of the most representative classifiers within pattern recognition and machine learning (see column 1 in Table 3) [11]. These methods are implemented in the CLOP toolbox.[4] Besides considering standard classification techniques we also considered a method that automatically generates classification models, called, Particle Swarm Model Selection (PSMS) [14]. PSMS explores the search space of all of the classification models that can be obtained by the combination of methods for data preprocessing, feature selection and classifiers using the implementations available in a machine learning toolbox (CLOP). Besides, PSMS optimizes the parameters of all of the models being considered. Hence PSMS automatically obtains models that can include methods for data preprocessing (e.g., normalization of features), feature selection (e.g., using correlation to select the most discriminative features) and classification.

Table 3 shows the average performance obtained with the different classification techniques we considered. We report the forecasting performance obtained in competitions M1 (using as training data the features extracted from time series from the M3 competition) and M3 (using as training data the features extracted from time series from the M1 competition), using the forecaster selected with the corresponding classifier. The table is divided row-wise into two blocks, the first block corresponds to the one-step forecasting and the second block presents the results of the multi-step forecasting. The table shows the results obtained in the two sets used to describe the time series. That is, the first block column (labeled as: NTSC) presents the results when the classification algorithms were trained with our features (see components of Eq. (10)), and the second block column (labeled as: All features) shows the results when the classification algorithms are trained with all the features describe in this contribution (i.e., components of Eq. (11)). The table also presents the p value of the Kruskal Wallis one-way analysis of variance [23].

From Table 3, it can be observed in bold the classification algorithm that obtained the best performance in each set of features. It can be observed that the classifiers that performed the best were those obtained with PSMS in almost all the cases, the only exception is on the one-step forecasting with the M3 competition and using the all the features, in this case the classification having the best performance is Kridge.

Comparing the performance of the classification algorithm on the different set of features, we can see that the Kruskal Wallis' p values are all below 0.05. In order to test whether the best classifier is the one that makes this difference in performance, we compare with all the other classifiers using the Wilcoxon signed-rank test. In almost all the cases, the p values of this test are well below 0.05, there are only two exceptions. These two are on the NTSC cases. On the set M1 the comparison RF vs PSMS has a $p=0.1303$, and on the competition M3 the comparison Klogistic vs Kridge has a $p=0.9952$.

On the other hand, for the case of multi-step forecasting, we see that the Kruskal Wallis' p values are only significant (with 90%

[4] http://clopinet.com/CLOP/.

Table 3
Average performance of the algorithm portfolio when the selection is performed by different classifications techniques. The last row in each group contains the p values of the Kruskal Wallis one-way analysis of variance to indicate whether the difference in performance is statistical significant.

Selection method	One-step forecasting			
	NTSC		All features	
	M1	M3	M1	M3
Klogistic	0.1066	0.1403	0.1024	0.1327
Neural	0.1124	0.1449	0.1143	0.1367
Kridge	0.1063	0.1374	0.1031	**0.1324**
RF	0.1053	0.1390	0.0999	0.1364
Logitboost	0.1090	0.1466	0.1056	0.1382
Naive	0.1131	0.1730	0.1105	0.1532
GKRidge	0.1107	0.1366	0.1120	0.1362
KNN	0.1177	0.1501	0.1152	0.1562
PSMS	**0.1029**	**0.1356**	**0.0990**	0.1342
Kruskal Wallis' p values	0.0032	0.0000	0.0000	0.0000
Selection method	**Multi-step forecasting**			
	NTSC		All features	
	M1	M3	M1	M3
Klogistic	0.3668	0.4179	0.3655	0.4201
Neural	0.3670	0.4136	0.3651	0.4125
Kridge	0.3670	0.4187	0.3657	0.4192
RF	0.3609	0.4103	0.3587	0.4096
Logitboost	0.3608	0.4159	0.3607	0.4174
Naive	0.3687	0.4287	0.3683	0.4246
GKRidge	0.3701	0.4118	0.3721	0.4126
KNN	0.3581	0.4150	0.3639	0.4165
PSMS	**0.3508**	**0.4057**	**0.3504**	**0.4069**
Kruskal Wallis' p values	0.6007	0.0762	0.6170	0.3439

Table 4
Performance of the portfolio when the selection is done perfectly, using AIC, and the best algorithm in the portfolio.

Selection method	One-step forecasting		Multi-step forecasting	
	M1	M3	M1	M3
Perfect model	0.0859	0.1130	0.2894	0.3284
Best classification algorithm	0.0990	0.1324	0.3504	0.4057
Forecaster with best performance	0.1120	0.1362	0.3551	0.4121

confidence) on the NTSC with the competition M3. In this case the Wilcox signed-rank test perform to test statistical significance difference of the best classifier gives p values below 0.05.

The results present in Table 3 can be compared against the average performance of the forecasters, Table 1, to see whether these algorithm portfolios have a better performance than the forecasters. It can be seen that the only portfolio that outperforms the forecasters in all the cases tested is PSMS. This result evidences the importance that preprocessing and feature selection methods as well as the fine tuning of parameters (as performed by PSMS) may have into the resultant performance of classifiers.

Table 4 presents a comparison of the performance of the portfolio when different techniques are used. In the first case, the table presents the performance when one has a perfect model, this is the best possible performance with these algorithms. It also contains the performance of the best models found using the classification algorithms and the models of performance. From left to right these algorithms are PSMS, Kridge, PSMS, and PSMS. Note that in the case of multi-step forecasting on set M3, the best performance was obtained using the features proposed here, that is, the components of Eq. (10). Finally, the last row presents the performance of a selection algorithm that always choose the algorithm with best average performance over all series of that particular competition. In order to know whether the difference in performance are statistical significant, we perform the Kruskal Wallis one-way analysis of variance [23] on each of the four cases tested (i.e. one-step forecasting with M1 and M3 and multi-step forecasting with M1 and M3). In all the cases, the test reports p values <0.01. To investigate which algorithms make these differences in performance, a pairwise Wilcoxon signed-rank test [45] was performed. All the p values of this test are below 0.05, indicating that these differences in performance are statistical significant.

The information shown in this table allows us to infer the improvement of the portfolio over selecting the forecaster with best average performance. Comparing the last row with the second row of the table, we can realize that in all the cases our selection mechanism is improving the performance of the best algorithm in the portfolio. In fact, we can use also the information of the first column to give a more precise idea of this improvement. Let us define an improvement of 100% to the model that gets the performance of the perfect model and assign a 0% to the selection algorithm that gets the same performance as the algorithm with best average performance. Under this circumstance, we can see that for set M1 and M3 in the case of one-step forecasting, our methodology gets an improvement of 49% and 16%, respectively. In the case of multi-step forecasting this improvement is of 7% and 7%, respectively.

6. Conclusions

In this contribution, we have modeled the performance of time series forecasters using features that have not been previously used to characterize time series. In fact, we considered different features that have shown success on modeling the performance of GP systems. Furthermore, in Section 2.5, we have extended these features, namely the difficulty indicators (i.e., ς); and, we have proposed novel indicators such as the indicators based on the similarity between the in and out-sample data.

The features proposed here complement previous approaches where the algorithm selection problem is solved using time series characteristics. In fact, as the results shown, our features proposed are very competitive obtaining the best performance on one of the four cases tested. In addition to this, our features can be easily implemented being based on the notion of finite differences, the performance of the forecasters on the in-sample data, and the similarity between indicators computed with the in-sample and forecast data.

In all the cases presented here the performance of the algorithm portfolio outperforms the performance of all the algorithms composing the portfolio, in the best case there is an improvement of 49%. This result was obtained with a classification model obtained with PSMS. Although, the performance of most of the linear models tested was highly competitive as well.

There are several future work directions we would like to explore. A first idea is that of generating individual classifiers for each type of characteristics and then build ensemble classification models that combine the outputs of the former classifiers. A second direction is to combine the information generated with both linear regression and classification models to test whether this approach would outperform the procedures presented here. Finally, we would like to perform an exploratory study on the importance of each feature for the classification and regression problems.

References

[1] M. Adya, J. Armstrong, F. Collopy, M. Kennedy, An application of rule-based forecasting to a situation lacking domain knowledge, Int. J. Forecast. 16 (October (4)) (2000) 477–484.

[2] M. Adya, F. Collopy, J. Armstrong, M. Kennedy, Automatic identification of time series features for rule-based forecasting, Int. J. Forecast. 17 (April (2)) (2001) 143–157.

[3] H. Akaike, A new look at the statistical model identification, IEEE Trans. Autom. Control 19 (December (6)) (1974) 716–723.

[4] E. Alpaydin, Introduction to Machine Learning, MIT Press, Cambridge, MA, USA, 2004.

[5] B. Arinze, Selecting appropriate forecasting models using rule induction, Omega 22 (November (6)) (1994) 647–658.

[6] B. Arinze, S.-L. Kim, M. Anandarajan, Combining and selecting forecasting models using rule based induction, Comput. Oper. Res. 24 (May (5)) (1997) 423–433.

[7] G.E.P. Box, D.R. Cox, An analysis of transformations, J. R. Stat. Soc. Ser. B (Methodol.) (1964) 211–252.

[8] R.G. Brown, Statistical Forecasting for Inventory Control, McGraw-Hill, 1959.

[9] F. Collopy, J.S. Armstrong, Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations, Manage. Sci. 38 (October (10)) (1992) 1394–1414.

[10] A.M. De Livera, R.J. Hyndman, R.D. Snyder, Forecasting time series with complex seasonal patterns using exponential smoothing, J. Am. Stat. Assoc. 106 (December (496)) (2011) 1513–1527.

[11] R. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley, 2000.

[12] J. Durbin, S.J. Koopman, A.C. Atkinson, Time Series Analysis by State Space Methods, vol. 15, Oxford University Press, Oxford, 2001.

[13] B. Efron, Least angle regression, Ann. Stat. 32 (2) (2004) 407–499.

[14] H.J. Escalante, M. Montes, L.E. Sucar, Particle swarm model selection, J. Mach. Learn. Res. 10 (June) (2009) 405–440.

[15] M. Graff, R. Poli, J.J. Flores, Models of performance of evolutionary program induction algorithms based on indicators of problem difficulty, Evol. Comput. (November) (2012) http://dx.doi.org/10.1162/EVCO_a_00096, in press.

[16] C.C. Holt, F. Modigliani, J.F. Muth, H.A. Simon, C.P. Bonini, P.R. Winters, Planning Production, Inventories, and Work Force, Prentice Hall, 1960.

[17] F. Hutter, Y. Hamadi, H.H. Hoos, K. Leyton-Brown, Performance prediction and automated tuning of randomized and parametric algorithms, in: *CP*, 2006, pp. 213–228.

[18] R. Hyndman, M. Akram, B. Archibald, The admissible parameter space for exponential smoothing models, Ann. Inst. Stat. Math. 60 (2) (2008) 407–426.

[19] R. Hyndman, A.B. Koehler, J.K. Ord, R.D. Snyder, Forecasting with Exponential Smoothing: The State Space Approach, Springer, August 2008.

[20] R.J. Hyndman, Y. Khandakar, Automatic time series forecasting: the forecast package for R, J. Stat. Software 27 (3) (2008) 1–22.

[21] R.J. Hyndman, A.B. Koehler, R.D. Snyder, S. Grose, A state space framework for automatic forecasting using exponential smoothing methods, Int. J. Forecast. 18 (July (3)) (2002) 439–454.

[22] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems), The MIT Press, 1992.

[23] W.H. Kruskal, W.A. Wallis, Use of ranks in one-criterion variance analysis, J. Am. Stat. Assoc. 47 (December (260)) (1952) 583.

[24] C. Lemke, B. Gabrys, Meta-learning for time series forecasting and forecast combination, Neurocomputing 73 (June (10–12)) (2010) 2006–2016.

[25] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, Y. Shoham, Boosting as a metaphor for algorithm design, in: *CP*, 2003, pp. 899–903.

[26] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, Y. Shoham, A portfolio approach to algorithm selection, in: *IJCAI*, 2003, pp. 1542.

[27] K. Leyton-Brown, E. Nudelman, Y. Shoham, Learning the empirical hardness of optimization problems: the case of combinatorial auctions, in: *CP*, 2002, pp. 556–572.

[28] K. Leyton-Brown, E. Nudelman, Y. Shoham, Empirical hardness models for combinatorial auctions, in: P. Cramton, Y. Shoham, R. Steinberg (Eds.), Combinatorial Auctions, MIT Press, 2006, pp. 479–504. (Chapter 19).

[29] K. Leyton-Brown, E. Nudelman, Y. Shoham, Empirical hardness models: methodology and a case study on combinatorial auctions, J. ACM 56 (4) (2009) 1–52.

[30] V. Mahajan, Y. Wind, New product forecasting models: directions for research and implementation, Int. J. Forecast. 4 (3) (1988) 341–358.

[31] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, R. Winkler, The accuracy of extrapolation (time series) methods: results of a forecasting competition, J. Forecast. 1 (2) (1982) 111–153.

[32] S. Makridakis, M. Hibon, The m3-competition: results, conclusions and implications, Int. J. Forecast. 16 (October (4)) (2000) 451–476.

[33] S. Makridakis, S.C. Whellwright, R.J. Hyndamn, Forecasting Methods and Applications, 3rd edition, John Wiley & Sons, Inc., 1992.

[34] Mario Graff, Riccardo Poli, Practical performance models of algorithms in evolutionary program induction and other domains, Artif. Intell. 174 (October (15)) (2010) 1254–1276.

[35] N. Meade, Evidence for the selection of forecasting methods, J. Forecast. 19 (6) (2000) 515–535.

[36] E. Nudelman, K. Leyton-Brown, H.H. Hoos, A. Devkar, Y. Shoham, Understanding random SAT: beyond the clauses-to-variables ratio, in: *CP*, 2004, pp. 438–452.

[37] R. Poli, W.B. Langdon, N.F. McPhee, A field guide to genetic programming, Published via ⟨http://lulu.com⟩ and freely available at ⟨http://www.gp-field-guide.org.uk⟩, 2008. (With contributions by J. R. Koza).

[38] R.B. Prudencio, T.B. Ludermir, Meta-learning approaches to selecting time series models, Neurocomputing 61 (October (0)) (2004) 121–137.

[39] J.R. Rice, The algorithm selection problem, Adv. Comput. 15 (1976) 65–118.

[40] C. Shah, Model selection in univariate time series forecasting using discriminant analysis, Int. J. Forecast. 13 (December (4)) (1997) 489–500.

[41] R. Shanmugam, Introduction to time series and forecasting, Technometrics 39 (4) (1997) 426.

[42] K.A. Smith-Miles, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Comput. Surv. 41 (1) (2008) 1–25.

[43] J.H. Stock, M.W. Watson, A comparison of linear and nonlinear univariate models for forecasting macroeconomic time series, Working Paper 6607, National Bureau of Economic Research, June 1998.

[44] X. Wang, K. Smith-Miles, R. Hyndman, Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series, Neurocomputing 72 (June (10–12)) (2009) 2581–2594.

[45] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bull. 1 (December (6)) (1945) 80.

[46] L. Xu, H.H. Hoos, K. Leyton-Brown, Hierarchical hardness models for SAT, in: *CP*, 2007, pp. 696–711.

[47] L. Xu, F. Hutter, H.H. Hoos, K. Leyton-Brown, SATzilla-07: the design and analysis of an algorithm portfolio for SAT, in: *CP*, 2007, pp. 712–727.

[48] L. Xu, F. Hutter, H.H. Hoos, K. Leyton-Brown, SATzilla: portfolio-based algorithm selection for SAT, J. Artif. Intell. Res. 32 (June) (2008) 565–606.

[49] P. Xu, A. Jayawardena, W. Li, Model selection for RBF network via generalized degree of freedom, Neurocomputing 99 (January) (2013) 163–171.

Mario Graff is an Associate Professor at the Faculty of Electrical Engineering at Universidad Michoacana de San Nicolas de Hidalgo. He obtained his PhD from the School of Computer Science and Electronic Engineering at the University of Essex, working under the supervision of Professor Riccardo Poli. He is a recipient of a best student paper award and two conference papers have been nominated to best paper award on EuroGP. His research interests are evolutionary computation, genetic programming, theory of evolutionary algorithms, models of algorithms performance.

Hugo Jair Escalante obtained his degree of PhD in computer science from the Instituto Nacional de Astrofisica, Optica y Electronica at Mexico, where he is now associate researcher. Dr. Escalante is member of the Mexican System of Researchers (SNI) since 2012, and co-director of ChaLearn, the Challenges in Machine Learning organization (2011–2013). Hugo Escalante obtained the 2010 Best PhD thesis on AI award from the Mexican Society on artificial intelligence (SMIA). His main research interests are on machine learning and computational intelligence with applications in text mining and high-level computer vision.

Jaime Cerda-Jacobo received the Electrical Engineer degree from University of Michoacan, Morelia, Mexico, in 1990, the M.Sc. in computer science from the Research and Advanced Studies Center at IPN, Mexico City, Mexico in 2000 and the PhD degree in electrical and electronic engineering from University of Southampton, UK in 2010. He is with the University of Michoacan where he has held various academic and administrative positions and is currently an Associate Professor. His actual research activities concentrate in the areas of machine learning, optimization and its decentralization by graph manipulations.

Alberto Avalos Gonzalez (IEEE Member), was born in Yurecuaro Michoacan, Mexico. He is a professor of electrical engineering at the Universidad Michoacana de San Nicolas de Hidalgo, Mexico, and head of the Electrical Machines Laboratory. He obtained his M.Sc. from Universidad Autonoma de Nuevo Leon, Mexico. His research activities are focused on the economic operation, planning, and regulation of electrical power systems, as well as power transformer modeling.