Autonomic Forecasting Method Selection: Examination and Ways Ahead

Marwin Züfle, André Bauer, Veronika Lesch,
Christian Krupitzer, Nikolas Herbst, Samuel Kounev
University of Würzburg
Würzburg, Germany
marwin.zuefle@uni-wuerzburg.de, [first].[lastname]@uni-wuerzburg.de

Valentin Curtef
Cosmo Consult Data Science GmbH
Würzburg, Germany
valentin.curtef@cosmoconsult.com

***Abstract*—Proactive adaptation improves the system performance of Autonomic Computing systems as it recognizes adaptation concerns in advance and adapts or prepares adaptation accordingly. To support this, forecasting methods use historical data to predict future system states. According to the "*No-Free-Lunch-Theorem*", there is no general forecasting method that performs best in all scenarios. Usually at design time, expert knowledge is required to decide on the forecasting method based on the anticipated characteristics of the resulting time series data. The uncertainty that results from the gap between design time and runtime for adaptive systems, as well as the environmental uncertainty at runtime, decreases the possibility that a forecasting method chosen at design time can cope with runtime demands. A common approach to tackle this problem is to use recommendation systems that automatically choose the forecasting methods. In this paper, we introduce a novel approach for forecasting method selection and a recommendation-based ensemble forecasting approach. We compare our approaches with one of the most widely used recommendation approaches for time series forecasting. Whereas the reference system uses static recommendation rules, we contrast a modified version which supports dynamic rule learning. The results of the evaluation show that our approaches outperform the original approach with static rule learning.**

***Index Terms*—Forecasting, rule induction, recommendation, univariate time series, data characteristics, method selection**

I. Introduction

In order to react to changes in the system or the environment through adaptation, Autonomic Computing systems integrate a mechanism for analyzing data monitored about system resources and the environment. Traditionally, this analysis uses models or thresholds and targets the current status of the system. Through the increasing omnipresence of computational power – e.g., by integrating cloud resources – the integration of proactive adaptation through analysis of predicted systems states becomes a suitable alternative to reactive adaptation [1], [2]. This eliminates delays in the adaptation process as the system recognizes adaptation concerns beforehand.

To enable proactive adaptation, the system integrates forecasting methods to predict future system states. Forecasting is an established and important discipline in many research areas enabling estimations of future observations by examining a series of past ones. Based on the "*No-Free-Lunch Theorem*" [3] from 1997, stating that there is no optimization algorithm best suited for all scenarios, an analogy can be drawn to the domain of time series forecasting: there is no forecasting method that performs best for all time series. In other words: forecasting methods have their advantages and drawbacks depending on the specific use case and considered time series.

Usually, choosing the best forecasting method relies on expert knowledge which can be expensive, might have a subjective bias, and may take a long time to deliver results. Additionally, Autonomic Computing systems need to handle complex types of uncertainties [2], i.e., when it is not possible to model *a priori* the situations in which such a system might reside at run time. This also influences the choice of the suitable forecasting method, as the characteristics of the time series data might be unknown at design time. For this reason, the integration of expert knowledge at design time is not applicable for Autonomic Computing scenarios like proactive auto-scaling of cloud environments. Here, the forecasts of future workloads have to be delivered within a very short period of time so that scaling decisions can be executed in time. Accordingly, autonomously choosing the forecasting method at runtime can cope with unanticipated situations as well as context switches.

In order to automate the process of applying forecasting methods, while avoiding to rely on expert knowledge, several hybrid approaches have been developed in the literature. One approach is *ensemble forecasting* where multiple forecasting methods are applied on the same time series and a weighted average of their forecasts is returned as a final result [4]–[9]. Another approach is based on *time series decomposition* where a time series is split into components and a different forecasting method is applied to each of the components [10]–[14]. The results of the component forecasts are then combined to derive a forecast for the entire time series. Finally, a further class of hybrid forecasting approaches is based on *forecasting method recommendation* [15]–[19]. Here, a set of rules is used to select a proper forecasting method based on specified features of the considered time series. The set of rules can either be created manually or generated automatically by using more sophisticated algorithms.

The most popular recommendation system for time series forecasting is from Wang *et al.* [17], postulating general rules for selecting forecasting methods. The approach is quite popular due to its detailed introduction to the topic of

TABLE I
FORECASTING METHODS TO BE RECOMMENDED.

Name	Brief functionality	Implementation
Random Walk	The assumption is made that the last observation is most likely to be seen next. That is, the predicted value equals the last value of the history.	self-implemented
ETS	The components season, trend, and error are used to construct an exponential smoothing state space model. The combination of the components can be either additive, multiplicative, or not present, whereas their weightings are adjustable [20].	`ets` method of the *R* package `forecast` [21]
(s)ARIMA	ARIMA models combine autoregressive models, moving average models, and differentiation to handle non-stationary time series. sARIMA is an extension of ARIMA that is able to model seasonal data by adding a seasonal component to each non-seasonal component [22].	`auto.arima` method of the *R* package `forecast`
ANN	A multilayer perceptron with a single hidden layer. The ANN is trained with lagged values of the time series. In this implementation, the number of lags is chosen automatically [21].	`nnetar` of the *R* package `forecast`

forecasting method recommendation. Surprisingly, despite the high citation count, the reliability of these rules has never been evaluated thoroughly. To address this issue, we focus on rule generation for forecasting method recommendation and propose a novel approach to dynamically (re-)learn recommendation rules based on a growing collection of data sets in this paper. We study the following research questions:

RQ1: What is the recommendation quality of the rules by Wang *et al.* on the original training data set?
RQ2: Are the recommendation rules by Wang *et al.* transferable to other data sets?
RQ3: How can the quality of forecasting method recommendations be improved?

The paper is structured as follows: Next, we introduce the foundations of time series forecasting, the applied methods, and forecasting method recommendation in Section II. In Section III, we summarize related work and the static recommendation system proposed by Wang *et al.* Following, we propose our own approach to dynamically learn recommendation rules in a context specific manner in Section IV. In Section V, we investigate the quality of the static rules by Wang *et al.*, i.e., we address RQ1 and RQ2. To address RQ3, we show how our approach improves the selection quality by dynamic rule learning. Lastly, Section VI concludes the paper.

II. Background

A univariate time series is a sequence of data points ordered by time. Typically, the data points are measurements and the time between two measurements is equally spaced. A time series can also be multivariate containing additional information for each timestamp, however, in this work, we focus on univariate time series.

A. Time Series Characteristics

Time series can be described by many different characteristics. In order to choose a proper forecasting technique, the exact value of the characteristics is typically less important than its degree of predominance. Thus, most of the characteristics are normalized to the range from zero to one. The larger this normalized value, the stronger the presence of the feature. In the following, the characteristics trend, seasonality, periodicity, skewness, kurtosis, serial correlation, non-linearity, self-similarity, and chaos will be considered since they have also been used by Wang *et al.* for their time series forecasting method recommendation system. More information on these characteristics can be found in the work of Wang *et al.* [17]. The calculation of the following characteristics as well as the normalization was made available by Hyndman, co-author in Wang *et al.* [17], on his website[1].

B. Forecasting Methods

In the research field of forecasting, many different approaches have been developed. However, we focus on four well-established forecasting methods that have also been used by Wang *et al.* [17], i.e., random walk (RW), exponential smoothing (ETS), (seasonal) autoregressive integrated moving average (ARIMA), and artificial neural network (ANN). A brief description of these methods is provided in Table I.

C. Forecasting Method Recommendation

The basic idea behind forecasting method recommendation is to learn rules based on time series characteristics. Thus, several time series characteristics are determined in order to cover most of the relevant aspects. Then, these features are used to learn dependencies between the performance of forecasting methods and the time series characteristics themselves.

Wang *et al.* adapted an architecture for meta-learning from Vilata [17] named "knowledge acquisition mode" and is initially meant for data mining tasks [23]. Our approach also follows this architecture. Figure 1 depicts the basic concept. The approach uses a database of time series examples for generation of the rule set. This approach is tripartite.

On the left hand side, the prediction component is depicted. It forecasts all time series in the example database. To predict future values, all time series are split into two parts. The first part is called history and is used to train the model of the respective forecasting method. The second part is used for validation. Thus, the predicted values are compared to these original values. The forecasting results are stored in the prediction results database.

[1]TS Characteristics: https://robjhyndman.com/hyndsight/tscharacteristics/

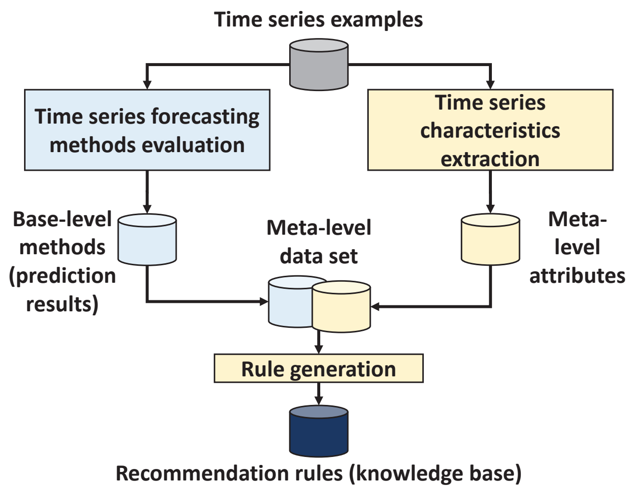

Time series examples
Time series forecasting methods evaluation
Time series characteristics extraction
Base-level methods (prediction results)
Meta-level data set
Meta-level attributes
Rule generation
Recommendation rules (knowledge base)

Fig. 1. Knowledge acquisition mode according to Vilata [23]. Wang *et al.* applied an adapted version in [17].

On the right hand side, the calculation of the characteristics is shown. This step determines, normalizes, and stores the characteristics (cf. Section II-A) in the meta-level attributes database. Moreover, it calculates the characteristics serial correlation, non-linearity, skewness, and kurtosis additionally on the de-trended and de-seasonalized time series. De-trending and de-seasonalizing means that the trend and seasonal components are removed from the original time series. Consequently, 13 characteristics are determined for each time series.

The third part combines the prediction results and the meta-level attributes to the meta-level data set that maps time series characteristics to the forecasting accuracy. Based on this data, rule generation algorithms create rules and store them in the knowledge base.

III. Related Work

To overcome the issue of the "*No Free Lunch Theorem*", many hybrid forecasting methods have been introduced in literature. In this context, a hybrid method consists of at least two individual forecasting methods. Thus, a hybrid method tries to compensate the drawbacks of one method by additionally using other forecasting methods. The existing hybrid forecasting methods in literature can be categorized into three groups of approaches each sharing the same basic concept: (i) ensemble forecasting, (ii) time series decomposition, and (iii) forecasting method recommendation.

Ensemble Forecasting was introduced by Bates and Granger in 1969 [4] and is the historically first group. Here, multiple forecasting methods are applied to the entire history of a time series. A weight is assigned to each of these forecast methods. By applying a weighted average, the final forecast is calculated. Although this concept is rather simple, the assignment of weights is essential for the quality of the prediction. Therefore, the range of approaches for assigning weights extends from simple average to much more sophisticated methods [5], [7], [8].

Time Series Decomposition takes advantage of strengths of individual forecasting methods for specific domains. By doing so, drawbacks of other forecasting methods in the same hybrid model can be equalized. There are two common approaches in literature that use time series decomposition as hybrid models. The first approach applies an individual forecasting method on the entire time series and afterwards uses a second individual method on the residuals [24], [25]. It is important that the second forecasting method has different strengths than the first one so that it is able to extract any more information from the residuals. The second approach utilizes time series decomposition explicitly. That is, the time series is split into several components, e.g., trend, seasonality, and remainder [10], [14]. Then, a specific forecasting method is selected for each component. In this way, the advantages of forecasting methods can be utilized.

Forecasting Method Recommendation aims at generating a set of rules in order to estimate the assumed best forecasting method based on measurable time series characteristics. In literature, there are two common approaches to create such a rule set: (i) expert knowledge and (ii) automation. Collopy and Armstrong introduced as first researchers the idea of using a manually created expert system in 1992 [15]. They used 18 features to generate 99 rules for the selection between four forecasting methods – random walk, regression, Brown's linear exponential smoothing, and Holt's exponential smoothing – and showed that their expert system could improve the forecasting accuracy compared to simple average ensemble forecasts. The modified version of Adya *et al.* reduces human intervention but still requires it [26]. The second type of approaches uses algorithms to automatically derive a rule set. In order to use such rule induction methods, the data set is split into a training and a validation set. The rules are learned based on the time series characteristics on the training set. Then, the rules are applied to the testing set. As first authors, Arinze *et al.* [16] applied artificial intelligence-based methods for rule generation in 1997. They extracted six time series characteristics and selected one out of six forecasting methods. Prudencio and Ludermir showed two case studies with six and seven features, respectively [27]. In the first case study, the recommendation system consisted of two forecasting methods, while three methods were chosen for the second case study.

In 2009, ***Wang et al.*** proposed clustering and rule induction algorithms to generate categorical and quantitative rules based on a large variety of time series features [17]. Therefore, the authors first introduced nine time series characteristics that are assumed to have a relation to the performance of four forecasting methods, i.e., ARIMA, ETS, ANN, and random walk. Then, hierarchical clustering and self-organizing maps were presented in the field of time series forecasting. Wang *et al.* described how to use these clustering methods to group similar time series together and then create judgmental and conceptive rules. Moreover, the authors applied a decision tree technique, i.e., the C4.5 algorithm, to generate quantitative rules automatically. Therefore, the forecasting methods are ranked for each time series according to their accuracy. As the

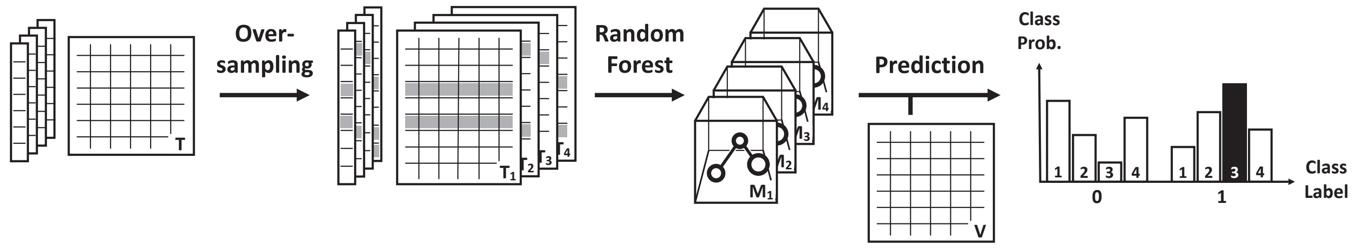


Fig. 2. Schematic process of the rule generation approach. First, oversampling is applied to the class label vectors and the matrix of characteristics T. The oversampled class label vectors and matrices of characteristics T_1 to T_4 are then used to learn a random forest model M_1 to M_4 for binary classification for each forecasting method in the recommendation system. The prediction of those models for a validation time series V results in probabilities for the class labels 0 and 1 for each forecasting method. The forecasting method with the highest probability of class label 1 is returned.

recommendation system only tries to say which forecasting method to choose, Wang *et al.* set the class label to 1 for the best forecasting method for each time series. All other forecasting methods receive the class label 0 for the respective time series. These labels are used as prediction class and the time series characteristics are used as meta-level attributes. Then, the C4.5 algorithm is applied with this data on each forecasting method to generate quantitative recommendation rules. This results in four rule sets, one for each forecasting method. These rules tell whether to use the respective forecasting method or not. Wang *et al.* provide all necessary parameter settings for applying the C4.5 algorithm as well as they present the generated rules. However, neither the conceptual nor the quantitative rules have been evaluated. In addition, the data set used to learn the models is not split and thus, there is no distinction between training and validation data.

Lemke and Gabrys identified another feature set [18]. Although multiple characteristics are similar to Wang *et al.*, they also omitted some and introduced several more. Lemke and Gabrys used this feature set to examine the applicability of different rule generation approaches for the selection between 15 forecasting methods and seven combination techniques on two data sets. Their results show that ensemble forecasting methods provide more robust predictions than single forecasting methods. Lemke and Gabrys also pointed out that the work of Wang *et al.* is so far the most comprehensive treatment of rule induction and forecasting method recommendation [18]. Since comparatively less research has been done in this area in recent years, this still holds true. Thus, we focus on the evaluation of the quantitative rule induction proposed by Wang *et al.* and also introduce two alternative approaches. The first approach is based on oversampling and binary classification, while the second approach combines forecasting method recommendation and ensemble forecasting.

IV. Proposed Alternative Approaches

In this paper, we propose two new approaches that we evaluate against the approach of Wang *et al.* The first approach generates rules by applying binary classification with oversampling. The second approach combines forecasting method recommendation and weighted ensemble forecasting.

A. Binary Classification with Oversampling

Figure 2 schematically shows our rule learning approach. It directly addresses the limitations of the approach by Wang *et al.*: (i) the single tree constructed by the C4.5 algorithm tends to overfit the training data, (ii) the training data are highly unbalanced, and (iii) the rules sometimes do not recommend any forecasting method. In the following, this section describes our approach.

First, the characteristics for all time series in the training set need to be determined. Therefore, the calculation and normalization of characteristics are the same as for the approach by Wang *et al.* Figure 2 depicts the matrix of characteristics for the training set as T on the left hand side. Here, each row of the matrix represents one time series.

Second, in order to evaluate the forecasting accuracy, the first 80% of observations are used to train the four forecasting methods. The remaining 20% are defined as the forecast horizon, from which the accuracy can be calculated. Then, the approach determines for each time series which forecasting method provides the best forecast. This results in a vector of zeros (not the best forecasting method) and ones (best forecasting method) for each forecasting method, i.e., each entry in this vector stands for one time series. Figure 2 shows the resulting four vectors on the left hand side.

Yet, since there are four forecasting methods in competition, it can be expected that each forecasting method will receive more zeros as class labels than ones. This results in a highly unbalanced training set and impacts the quality of the generated rules in a negative way. To handle unbalanced data, several approaches have been developed. On the one hand, performance metrics (e.g., precision, recall, and F1-score) can be adjusted to represent the unbalanced data set correctly. On the other hand, oversampling or undersampling can be applied for balancing. Oversampling creates new instances of the minority either by simply duplicating existing instances or artificially creating new instances. In contrast, undersampling discards instances of the majority class until both prediction classes are balanced. In this work, we use oversampling to avoid reducing our training base. We apply the duplication approach for oversampling because, in a preliminary study, it provided better performance for our scenarios than the approach that synthetically creates new instances. However, this study is out of the scope of the paper. Here, not only the entries of the class label vectors are duplicated, but also the corresponding rows of the matrix of characteristics. Figure 2 illustrates the oversampled matrices as T_1 to T_4 along with the respective class label vectors. The oversampled entries of the vectors and matrices are depicted in grey.

Next, a rule learning algorithm generates rules using the oversampled training data. Wang *et al.* use the C4.5 algorithm which creates a decision tree based on the time series characteristics of the training set. However, a single decision tree might not cover all relevant aspects for decision making and tends to overfit. Thus, we apply an ensemble learning method based on random forest to dynamically learn the rule set as random forest constructs several decision trees. We use a parameterized version to perform binary classification[2]. Thus, this step results in four random forest models, M_1 to M_4, one for each forecasting method (cf. Figure 2).

Lastly, for recommending the best forecasting method, the matrix of characteristics of the validation time series V is passed to the random forest models. The validation set consists of time series that are not used for training. Now, each of these models delivers two class probabilities per time series. The probability of class label 1 indicates how likely a forecasting method is to be suitable for the respective time series. As soon as the class probabilities for all forecasting methods are available, the one with the highest probability for class label 1 is recommended to ensure that a forecasting method is recommended in any case.

B. Recommendation-based Ensemble Forecasting

To tackle the problem of typical forecasting method recommendation[3] we use a combination of ensemble forecasting and forecasting method recommendation. On the one hand, we apply linear regression for automatically adjusting the weights for a linear combination of forecasting methods. On the other hand, we use an activation function that acts as a sieve to select only the most appropriate methods.

To this end, we derive a linear combination of each forecasting method Y^i (ARIMA, Exponential smoothing, artificial Neural network, and Random walk). Each weight ϑ^i is estimated by a linear regression of the characteristics of the previously seen time series. As the regression only has the values zero and one as input, it delivers values in this range. We interpret this output as the probability of how good the forecasting method is suitable for the time series. In contrast to typical weights in ensemble forecasting, these weights have to be activated. That is, if a weight fulfills the activation function Γ, the weight is used, otherwise, the weight is set to zero. A weight of zero means that the associated forecasting method is not considered for the forecast. The linear combination is also normalized by ϑ, which is the sum of the activated weights. For the activation, different functions can be used. Our function is two-folded, i.e., we activate a weight if it fulfills both criteria: the weight ϑ^i must be (i) greater or equal to the mean of the weights $\overline{\vartheta}$ and (ii) greater or equal to the share α of the maximum weight $\hat{\vartheta}$.

The first condition has the advantage that by using the mean of all weights, the assumed most appropriate forecasting method is chosen in any case and the assumed worst suitable method is omitted. Further, if all weights have the same value, all forecasting methods are considered. As the mean can be influenced by outliers, the second condition allows having a higher threshold than the mean if the outlier is near to zero. The steps can be mathematically expressed as follows:

$$Y = \frac{1}{\vartheta} \sum_{i \in \{A,E,N,R\}} Y^i \cdot \Gamma(\vartheta^i) \quad with \quad \vartheta = \sum_{i \in \{A,E,N,R\}} \Gamma(\vartheta^i) \ ,$$
$$\Gamma(\vartheta^i) = \vartheta^i \cdot \Big(1 - max(sign(max(\overline{\vartheta}, \alpha \cdot \hat{\vartheta}) - \vartheta^i), 0)\Big) \ ,$$
$$\overline{\vartheta} = \underset{i \in \{A,E,N,R\}}{mean}(\vartheta^i) \ , \quad and \quad \hat{\vartheta} = \underset{i \in \{A,E,N,R\}}{max}(\vartheta^i)$$

Both the binary classification with oversampling approach as well as the recommendation-based ensemble forecasting approach can be applied for runtime performance decisions in autonomous systems. In an offline phase, the initial recommendation rules need to be learned. The rules and weights can be used for new and unseen time series at runtime since applying the recommendation is very fast. Moreover, these new time series can then be used to dynamically re-learn the recommendation rules and weights. As we rely on a random forest based approach for rule generation[4], the dynamic re-learning of rules is feasible with a small overhead at runtime.

V. Evaluation

In this section, we address the research questions formulated in Section I. First, Section V-A introduces the experimental setup. The following section investigates the quality of the rules proposed by Wang *et al.*, i.e., addressing RQ1 and RQ2. In Section V-C, we compare our own approaches with a dynamic version of the approach proposed by Wang *et al.* Afterwards, we discuss threats to validity. In Section V-E, we summarize our findings and conclude the evaluation.

A. Experimental Setup

The entire analysis is implemented in *R* version 3.3.2. In order to calculate and normalize the time series characteristics, the original script by Hyndman, which is also written in *R*, is applied. The *R* package `forecast` is integrated in version 7.3 to perform the forecasting methods [21].

Since we aim at evaluating the recommendation performance of the approach by Wang *et al.* and comparing our novel approaches with it, we first use the data set mentioned in the original paper and the same forecasting methods, ignoring other interesting forecasting methods (e.g., tBATS, Theta, SVM, and LSTM). However, as some of the sources are not

[2]Random forest: https://cran.r-project.org/web/packages/randomForest/index.html (classwt = false, mtry = 2, type = binary classification)

[3]By selecting only one forecasting method for each time series, the forecasting accuracy over a set of time series typically shows large variance.

[4]Random forest is typically very fast for both learning and prediction. On our data sets, the rule generation is done within minutes and the recommendation for a single time series within milliseconds.

available anymore, we focus on the time series from the UCR Time Series Classification Archive [28], which are the main part of the original data set. This data set consists of 377 time series and is called UCR in the following. We use this data set to address RQ1. In addition, we use another data set consisting of 1005 time series from the M3 competition [29] to provide an answer to RQ2. Finally, we use both data sets to compare our approaches to the rules by Wang *et al.* and the individual forecasting methods (RQ3).

B. Evaluation of Approach by Wang et al.

As a first contribution, we evaluate the postulated rules of Wang *et al.* (cf. RQ1 and RQ2) using the following aspects:

- Which average rank does the recommendation achieve?
- When comparing the forecasting error of each recommended forecasting method to the actual best performing forecasting method in competition, what is the average degradation in accuracy?

To evaluate the recommendation quality of the approach proposed by Wang et. al [17], we conduct two sets of experiments. The first one uses the UCR data set and the second experiment row uses time series from the M3 competition.

For both data sets, we investigate the accuracy of both one-step-ahead and multi-step-ahead forecasting. Thus, we observe the rank and the forecast accuracy degradation for each forecasting method and each time series, comparing them with the recommendation provided by Wang *et al.*'s approach. The rank reflects on which place the method is sorted according to the forecast accuracy. The forecast accuracy degradation indicates how much lower the associated method performs compared to the best method. The ground truth of which method would have been best is only known ex-post. We chose the mean absolute percentage error (MAPE) as error measure:

$$MAPE = 100 \cdot \frac{1}{k} \sum_{i=1}^{k} |\frac{e_i}{Y_i}|$$

Here, k represents the length of the forecasting horizon, Y_i stands for the original value of the i^{th} observation in the forecasting horizon, and e_i describes the forecasting error at observation Y_i, i.e., the difference between the predicted and the original value.

To provide answers to the first two research questions, Table II presents the average ranks when applying the rules from Wang *et al.* on the UCR (cf. RQ1) and M3 (cf. RQ2) data sets. The rules and individual forecasting methods are applied to both data sets. We evaluate not only the multi-step-ahead performance but also the one-step-ahead accuracy. The average rank of the recommendation system varies from 2.62 to 2.80. Since there are only four forecasting methods in competition, the theoretical expected value of a random guess would be 2.5 if all forecasting methods perform best equally frequent. Moreover, it can be seen that the average ranks of ARIMA, ETS, and ANN are better than the ranks of the recommendation rules for both scenarios of the UCR data set. Only random walk shows a worse average rank for

TABLE II
AVG. RANKS FOR THE RULES BY WANG *et al.* AND INDIVIDUAL METHODS.

Data set	Wang *et al.*	ARIMA	ETS	ANN	RW
UCR one-step	2.62	**2.33**	2.42	2.34	2.92
UCR multi-step	2.80	**1.95**	2.64	2.59	2.83
M3 one-step	2.63	2.38	**2.37**	2.67	2.58
M3 multi-step	2.75	2.28	**2.17**	2.79	2.76

the UCR data set. Please note that Wang *et al.* used the UCR data set to learn their rule set, which makes these results even more surprising. In terms of the M3 data set, ARIMA and ETS again outperform the recommendation system for both scenarios. Even random walk shows a smaller average rank than the rule-based selection for one-step-ahead forecasting.

The distribution of achieved ranks when applying the recommendation system can be seen in Figure 3. Here, the horizontal axis depicts the four ranks and the vertical axis shows the respective probability density. For each rank, four bars are depicted whereupon each bar represents one scenario of a data set, i.e., from left to right: one-step-ahead forecasting on UCR, multi-step-ahead forecasting on UCR, one-step-ahead forecasting on M3, and multi-step-ahead forecasting on M3. The figure shows that the most frequent rank is 4 and thus, the worst forecasting method in the system. The probability density of rank 4 is never less than 30% for any scenario.

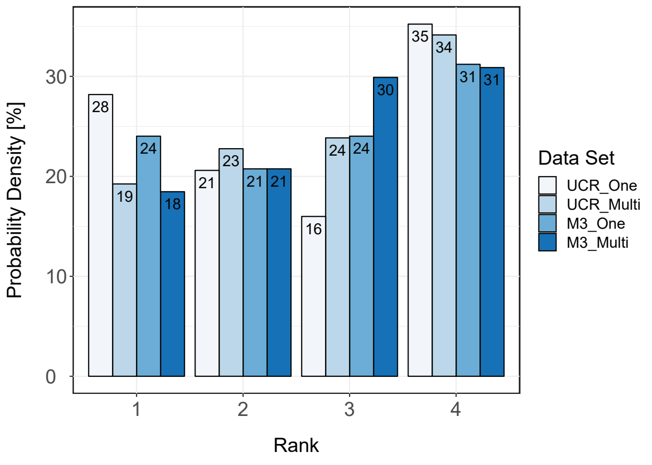


Fig. 3. Histogram of the distribution of achieved ranks for the rules by Wang *et al.* on the UCR and M3 data sets.

As the rules generated by Wang *et al.* provide a separate recommendation for each forecasting method whether to apply it or not, it may happen that the rules do not recommend any of the forecasting methods. However, this does not help autonomous systems in decision making. Table III presents the amount and percentage share of missing recommendations on the UCR and M3 data sets. For each data set, the values are the same for one-step-ahead and multi-step-ahead forecasting. It can be seen that the recommendation system fails to suggest a forecasting method for almost 15% of all time series in the UCR data set and even 44% in the M3 data set.

TABLE III
MISSING RECOMMENDATIONS FOR THE RULES OF WANG *et al.*

Data set	Amount	Percentage share
UCR	55	14.6%
M3	442	44.0%

Besides the ranking and examination of missing recommendations, the degradation of accuracy is investigated. Thus, for each time series and forecasting method, the achieved MAPE is compared to the lowest MAPE value for the respective time series. The average degradation of accuracy is shown in Table IV. For each data set, the first row shows the mean. It can be seen that the mean is typically very large as there are high outliers. In addition, for one-step-ahead forecasting on M3, the mean values of Wang *et al.*, ARIMA, ETS, and ANN are infinity because random walk achieves a MAPE of 0% for some time series. For this reason, the median is also presented in the second row for each data set. Again, ARIMA and ETS provide better accuracy for all data sets and scenarios compared to the rule-based selection. ANN also yields better results than the recommendation system in most cases. Only the median of multi-step-ahead forecasting on M3 is slightly worse than applying the rules of Wang *et al.* In comparison to random walk, neither the rules of Wang *et al.* nor random walk has a significant advantage in the presented scenarios.

TABLE IV
ACCURACY DEGRADATION FOR ALL FORECASTING METHODS AND THE RULES BY WANG *et al.* FOR EACH DATA SET, THE FIRST ROW SHOWS THE MEAN VALUE AND THE SECOND ROW THE MEDIAN.

Data set	Wang *et al.*	ARIMA	ETS	ANN	RW
UCR one-step	992.6% 96.8%	2096.9% **46.1%**	**312.7%** 55.8%	404.0% 48.6%	1053.2% 130.1%
UCR multi-step	282.3% 108.5%	**92.5%** **3.1%**	328.5% 100.1%	391.4% 74.1%	284.6% 89.8%
M3 one-step	Inf 106.4%	Inf **40.4%**	Inf 42.5%	Inf 68.7%	**1150.8%** 62.3%
M3 multi-step	69.4% 33.4%	53.7% 10.0%	**42.6%** **7.9%**	92.0% 35.1%	81.6% 34.0%

To answer RQ1, the postulated rules of Wang *et al.* on the UCR data set achieve average ranks and accuracy degradations worse than those of ARIMA and ETS, i.e., the use of one of these methods would provide better forecasting performance. The quality of recommendations is therefore relatively poor. ARIMA and ETS outperform the rules of Wang *et al.* also on the M3 data set. Moreover, the share of missing recommendation increases dramatically compared to the UCR data set. Thus, RQ2 can be rejected since the postulated rules are not transferable to the M3 data set.

From a statistical perspective, only the next forecast value is important, but planning in advance, for instance in autonomic systems like auto-scaling, requires multiple values. Thus, in the next sections, we investigate multi-step-ahead forecasts.

C. Evaluation of Alternative Approaches

To evaluate our proposed alternative approaches, we use 80% of the time series from both data sets to learn the relation between characteristics and the performance of the forecasting methods. The remaining 20% of the time series from both data sets are then used to evaluate the rules. Since such a division is arbitrary, the data sets are divided randomly 100 times and the results of all permutations are averaged.

Binary Classification with Oversampling
In general, it is not possible to cover all aspects of time series in such small training databases like the UCR data set or the M3 data set (cf. Section V-A). Thus, either a huge and diverse time series database is required or the learned recommendation system targets a specific domain. Our evaluation is based on two data sets. Therefore, we create the rules twice, i.e., once for each data set. So, in contrast to Wang *et al.*, we do not provide general rules for the selection of forecasting methods. Instead, we learn the rules dynamically on each data set under investigation resulting in a more realistic setting of a recommendation system optimized for a specific domain.

Table V shows the comparison of (I) the rules by Wang *et al.* who uses the C4.5 algorithm for rule generation to (II) dynamically generating the rules according to the procedure by Wang *et al.* but using the C5.0 algorithm, an extension of the C4.5 algorithm, and to (III) our approach based on binary classification with oversampling. By replacing the C4.5 algorithm with the C5.0 algorithm, which generally achieves more accurate results, the approach of Wang *et al.* is updated to the current version. Dynamically generating the rules means to learn the rules on the training part of each data set and not to learn a global, static set of recommendation rules for all data sets. Besides this adaptation of dynamical rule learning and using C5.0 instead of C4.5, the rule induction process of Wang *et al.* is applied. To compare the three approaches, we apply ranks, missing recommendations, and accuracy degradation as metrics.

In terms of ranks, approach (II) as well as our approach show significant improvements. For the M3 data set, however, approach (II) improves the average rank even more than our approach. Yet, the difference between the ranks of these two methods is rather small, i.e., 0.07. In contrast, our approach outperforms approach (II) on the UCR data set in terms of the average rank by 0.03. Again, the difference between the two approaches is very small.

Our approach and approach (II) provide very similar results on both data sets with respect to the degradation of accuracy. The measure of our approach equals the one of approach (II) on the M3 data set and is only 0.2 percentage points behind approach (II) on the UCR data set. Again, the dynamic rule learning modification of Wang *et al.*, i.e., approach (II), as well as our approach reduce the accuracy degradation distinctly.

Though, when taking a look at the missing recommendations, it can be seen that approach (II) does not provide a recommendation for a very large proportion of time series.

TABLE V
AVG. RANKS, AMOUNT OF MISSING RECOMMENDATIONS, AND AVG. DEGRADATION IN ACCURACY FOR THE RULES BY WANG *et al.* (I), LEARNED RULES USING C5.0 WITHOUT OVERSAMPLING (II), AND OUR BINARY CLASSIFICATION WITH OVERSAMPLING APPROACH (III).

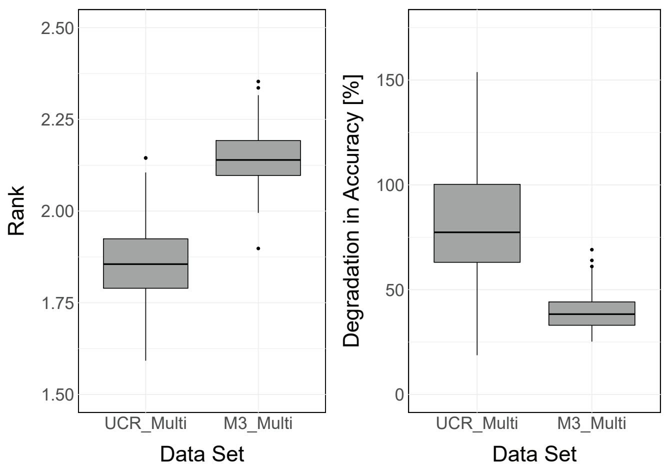
Data set	**(I) Orig. Rules from Wang *et al.***			**(II) Learned Rules - no Oversampling**			**(III) Binary Classification with Oversampling**		
	Rank	No Recom.	Degradation	Rank	No Recomm.	Degradation	Rank	No Recom.	Degradation
UCR multi-step	2.80	14.5%	282.3%	1.89	31.6%	**79.5%**	**1.86**	**0%**	79.7%
M3 multi-step	2.75	44.0%	69.4%	**2.07**	79.6%	**39.5%**	2.14	**0%**	**39.5%**

In fact, the percentage share of missing recommendations increases drastically compared to the rules by Wang *et al.*, i.e., from 14.5% to 31.6% and 44.0% to 79.6%. Thus, approach (II) performs some kind of “cherry-picking”, since it only provides recommendations for time series for which it is relatively confident. In contrast, our approach guarantees to recommend a forecasting method. So, our approach also recommends a forecasting method, even if none of them is perfectly fitting for the time series. For such scenarios, it is hard to estimate which of the forecasting methods would perform best, because the models try to learn which forecasting method performs well for certain characteristics. However, in this scenario, there are none of the forecasting methods performing well, but still, the one with the smallest forecasting error should be recommended. Thus, it is reasonable that the average rank and accuracy degradation is slightly larger than for approach (II). On top of that, not recommending any forecasting method is not feasible for most autonomous systems in practice since they require this input for their decision making process.

As the results for the binary classification with oversampling approach presented in Table V are only averaged over 100 random splits for training and validation, the results of all splits are shown as box plots: Figure 4a provides the box plots for the ranks and Figure 4b depicts the distributions of degradation. For both figures, the data sets, i.e., UCR and M3 both for multi-step-ahead forecasting, are shown on the horizontal axis.

Figure 4a shows that the interquartile range, indicated by the upper and lower border of the box plot, for the M3 data set is very small as the first quartile has a value of 2.10 and the third quartile is 2.19. Moreover, there are only a few outliers. This shows that the average rank for the oversampling approach is rarely dependent on the choice of split between training and validation data. For the UCR data set, the variation is a bit larger. The interquartile range reaches here from 1.79 to 1.92. However, this range is still acceptable.

In terms of degradation of accuracy, Figure 4b shows that the variation for the M3 data set is very small, i.e., the first quartile is 33.0% and the third quartile is 44.2%. This underlines the previous statement that the performance of this approach does not depend much on the split of data. In contrast, the degradation of accuracy varies highly on the UCR data set. Here, the interquartile range is much larger compared to the M3 data set. It ranges from 63.0% to 100.2%. However, the long whiskers indicate that some of the splits highly affect the degradation of accuracy on the UCR data set presumably as this data set is smaller than the M3 data set and is also intended for time series classification rather than forecasting.



(a) Ranks. (b) Degradation of accuracy.

Fig. 4. Box plots of ranks and accuracy degradation for 100 splits using the binary classification with oversampling approach on the M3 and UCR data sets.

Figure 5 shows a histogram of ranks for the UCR and M3 data sets. Here, the binary classification with oversampling approach is applied for multi-step-ahead forecasting. On the horizontal axis, the ranks are presented whereas the probability density of each rank is shown on the vertical axis. For each rank, two bars are depicted. The left bar represents the UCR data set and the right bar the M3 data set. In contrast to the histogram of ranks for the original rules, which is shown in Figure 3, the probability densities of the ranks are strictly decreasing. Thus, for both data sets rank one is by far most frequent with 51% and 39% for the UCR data set and the M3 data set, respectively. In addition, the worst forecasting method for each time series, i.e., rank four, is least recommended (9% for the UCR data set and 16% for the M3 data set) with a distinct distance to ranks three and two.

Taking into account the average rank, accuracy degradation, and the distribution of ranks, our binary classification with oversampling approach shows distinct improvements in the recommendation quality. Moreover, when comparing the recommendation with the individual forecasting methods, our approach also outperforms them. On the UCR data set, ARIMA performs best with an average rank of 1.95 (cf. Table II) and accuracy degradation of 92.5% (cf. Table IV). That is, the average rank of our approach is lower by 0.09 and the average accuracy degradation is less by 12.8 percentage points (cf. Table V). For the M3 data set, ETS achieved the

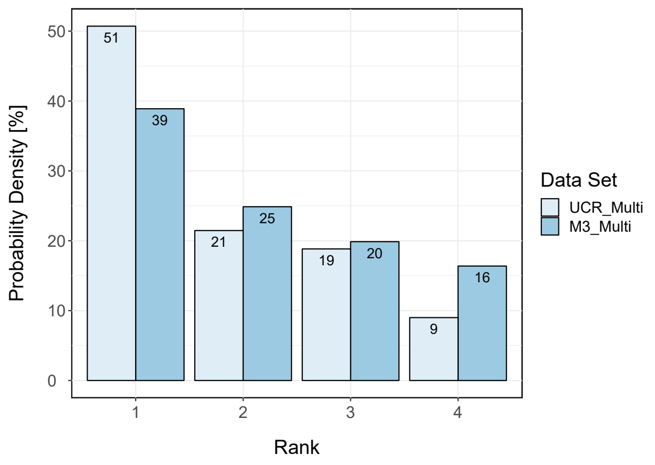


Fig. 5. Histogram of the rank distribution for the approach using binary classification and oversampling on the UCR and M3 data sets.

best results of the individual forecasting methods. However, our approach still provides better results than ETS with a reduction of the average rank by 0.03 (cf. Tables II & V). In addition, our approach decreases the degradation in accuracy by 3.1 percentage points (cf. Tables IV & V).

Recommendation-based Ensemble Forecasting
To evaluate the recommendation quality of the linear model with activation function, Table VI presents the degradation in accuracy for all forecasting approaches averaged over all 100 splits. As this linear model approach creates an entirely new forecast, it can be seen as a fifth forecasting method, in addition to ARIMA, ETS, ANN, and random walk. Thus, comparing ranks would yield completely other results than shown before. Therefore, only the degradation in accuracy is shown for this approach. However, also keep in mind that the values of degradation here are slightly larger than in the previous sections as the optimal forecast can now be our recommendation-based ensemble forecasting approach. Table VI shows that our approach achieves the smallest degradation in accuracy for the M3 data set, followed by our binary classification with oversampling approach and ARIMA. All other approaches are clearly outperformed. In terms of the UCR data set, our binary classification with oversampling approach reaches the best result ahead of ARIMA and our recommendation-based ensemble forecasting approach. Again, the other approaches are all far behind. On the M3 data set, the rules by Wang *et al.* only outperform the ANN and random walk whereas they perform worst for the UCR data set.

D. Threats to Validity

In our work, we try to reconstruct the original data set used by Wang *et al.* that consists mainly of the UCR data set. Unfortunately, the exact time series have not been stated but only the amount. However, by analyzing the year of publication of Wang *et al.*, we were able to identify the UCR data sets. Though, we could not use time series from the other sources since some of them are not available anymore or could not be uniquely identified. Indeed, we use 46 from 62 time series data sets. As this is almost 75% of the original data set, the results for the rules proposed by Wang *et al.* may be slightly better while using the original data set, but the missing time series would not lead to the best results.

Moreover, the time series from the UCR target initially classification and not forecasting. The predictability of these time series is rather poor since their entropy is very low, e.g., some time series only show peaks similar to Dirac impulses. For this reason, the predictions in general are relatively inaccurate. This results in a largely random ranking of the best prediction methods for such time series. Although random walk is typically used as a baseline forecasting method, this approach performs comparably well on these data sets.

As the rules by Wang *et al.* sometimes do not recommend a forecasting method, we omit these time series in the evaluation in Section V-B. By punishing the missing recommendations, the approach of Wang *et al.* would perform even worse.

Since we try to reproduce the setup of Wang *et al.*, we consider only time series with more than 100 and less than 1000 values. Thus, the observed results may only be valid for time series with the specified length.

To ensure comparability, we only consider the forecasting methods of Wang *et al.*: ARIMA, ETS, ANN, and random walk. Thus, the results are only valid while using these methods. However, since there are only four methods in the recommendation system, a distinct improvement in the average rank is hard to validate. Moreover, some of the forecasting methods are very similar to each other, e.g., ETS and ARIMA often provide very close results for short time series with little information content and short seasonality. However, the data sets contain many time series of this kind. In order to validate the recommendation system in more detail, more forecasting methods and in particular more diverse ones are required.

The results of our experiments are easily reproducible as the entire code is written in *R* using only publicly available packages and data sets. Moreover, we are working on providing the *R* scripts on GitHub[5].

[5]GitHub: https://github.com/marwinzuefle/ForecasterRecommendation

TABLE VI
DEGRADATION IN ACCURACY FOR ALL APPROACHES AVERAGED OVER ALL 100 RANDOM SPLITS.

Data set	Wang *et al.*	ARIMA	ETS	ANN	Random walk	Bin. Class. + Oversampling	Recom. + Ensemble
M3 multi-step	69.8%	59.9%	46.2%	91.0%	83.6%	42.3%	**40.6%**
UCR multi-step	404.9%	98.0%	345.6%	386.3%	300.9%	**82.1%**	104.9%

E. Evaluation Findings

The approach proposed by Wang *et al.* was not yet evaluated and thus, we investigated the recommendation quality on the original and an additional well-known time series data set. The key takeaways of our evaluation are:

(I) The experimental results show that the rules proposed by Wang *et al.* are outperformed by all forecasting methods besides random walk on both data sets (RQ1 and RQ2).

(II) Compared to the individual forecasting methods and the rules of Wang *et al.*, our binary classification with oversampling approach yields the best results on both data sets for both average rank and average degradation in accuracy (RQ3).

(III) The approach of using a linear regression model with activation function for the recommendation and combination of forecasting methods achieves the best results on the M3 data set and the third best results on the UCR data set (RQ3).

VI. Conclusion

In this work, we focus on the application of forecasting methods to support proactive decision making in autonomic computing systems. As there is no forecasting method that works best for all time series, usually expert knowledge is required to choose the right method. Recommendation systems aim to tackle this problem, however, those are often not evaluated in literature. Thus, we provide a detailed evaluation of one of the most cited recommendation approaches for time series forecasting. As the experimental results of this approach show that the proposed rules do not perform well, i.e., three out of four individual forecasting methods achieve a better forecasting performance, we introduce two novel approaches. The first approach applies oversampling and binary classification, while the second approach uses recommendation-based ensemble forecasting. We show that both approaches outperform the rules by Wang *et al.* significantly.

As future work, we plan to integrate our approach into a framework for proactive analyzing in autonomic computing systems and apply this framework in a real world system. In this work, the characteristics, forecasting methods, and data sets have been limited to the set presented in the work of Wang *et al.* due to comparability. To offer more flexibility and to compare our approach to other works, we plan to (i) include more time series characteristics, (ii) use other data sets with more diversity in their characteristics, and (iii) integrate more forecasting methods. In addition, we plan to investigate other class labeling methods.

Acknowledgements

This work was co-funded by the German Research Foundation (DFG) under grant No. (KO 3445/11-1) and the IHK (Industrie- und Handelskammer) Würzburg-Schweinfurt.

References

[1] C. Krupitzer, F. M. Roth, S. VanSyckel *et al.*, "A survey on engineering approaches for self-adaptive systems," *Pervasive and Mobile Computing Journal*, vol. 17, no. Part B, 2015.

[2] D. Weyns, "Software engineering of self-adaptive systems: An organised tour and future challenges," in *Handbook of Software Engineering*. Springer, 2017.

[3] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 1, 1997.

[4] J. M. Bates and C. W. Granger, "The combination of forecasts," *Journal of the Op. Research Society*, vol. 20, no. 4, 1969.

[5] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *Int. Journal of Forecasting*, vol. 5, no. 4, 1989.

[6] P. Newbold and C. W. Granger, "Experience with forecasting univariate time series and the combination of forecasts," *Journal of the Royal Statistical Society. Series A (General)*, 1974.

[7] L. M. De Menezes, D. W. Bunn, and J. W. Taylor, "Review of guidelines for the use of combined forecasts," *European Journal of Op. Research*, vol. 120, no. 1, 2000.

[8] T. N. Krishnamurti, C. Kishtawal, Z. Zhang *et al.*, "Multimodel ensemble forecasts for weather and seasonal climate," *Journal of Climate*, vol. 13, no. 23, 2000.

[9] M. Sommer, A. Stein, and J. Hähner, "Local ensemble weighting in the context of time series forecasting using xcsf," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016.

[10] N. Liu, Q. Tang, J. Zhang *et al.*, "A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids," *Applied Energy*, vol. 129, 2014.

[11] J.-X. Xie, C.-T. Cheng, K.-W. Chau, and Y.-Z. Pei, "A hybrid adaptive time-delay neural network model for multi-step-ahead prediction of sunspot activity," *Int. Journal of Environment and Pollution*, vol. 28, no. 3-4, 2006.

[12] A. J. Conejo, M. A. Plazas, R. Espinola, and A. B. Molina, "Day-ahead electricity price forecasting using the wavelet transform and arima models," *IEEE Trans. on Power Systems*, vol. 20, no. 2, 2005.

[13] S. Schlüter and C. Deuschle, "Using wavelets for time series forecasting: Does it pay off?" IWQW discussion paper series, Tech. Rep., 2010.

[14] M. Züfle, A. Bauer, N. Herbst *et al.*, "Telescope: a hybrid forecast method for univariate time series," in *Int. Work-Conference on Time Series*, 2017.

[15] F. Collopy and J. S. Armstrong, "Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations," *Management Science*, vol. 38, no. 10, 1992.

[16] B. Arinze, S.-L. Kim, and M. Anandarajan, "Combining and selecting forecasting models using rule based induction," *Computers and Operations Research*, vol. 24, no. 5, 1997.

[17] X. Wang, K. Smith-Miles, and R. Hyndman, "Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series," *Neurocomputing*, vol. 72, no. 1012, 2009.

[18] C. Lemke and B. Gabrys, "Meta-learning for time series forecasting and forecast combination," *Neurocomputing*, vol. 73, no. 10-12, 2010.

[19] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "An autonomic prediction suite for cloud resource provisioning," *Journal of Cloud Computing*, vol. 6, no. 1, 2017.

[20] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *Int. Journal of Forecasting*, vol. 18, no. 3, 2002.

[21] R. J. Hyndman, G. Athanasopoulos, C. Bergmeir *et al.*, *forecast: Forecasting functions for time series and linear models*, 2018, r package version 8.4. [Online]. Available: http://pkg.robjhyndman.com/forecast

[22] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. Melbourne, Australia: OTexts, 2014.

[23] R. Vilalta, C. G. Giraud-Carrier, P. Brazdil, and C. Soares, "Using meta-learning to support data mining," *IJCSA*, vol. 1, no. 1, 2004.

[24] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, 2003.

[25] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," *Omega*, vol. 33, no. 6, 2005.

[26] M. Adya, F. Collopy, J. S. Armstrong, and M. Kennedy, "Automatic identification of time series features for rule-based forecasting," *Int. Journal of Forecasting*, vol. 17, no. 2, 2001.

[27] R. Prudêncio and T. Ludermir, "Using machine learning techniques to combine forecasting methods," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2004.

[28] Y. Chen, E. Keogh, B. Hu *et al.*, "The ucr time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

[29] S. Makridakis and M. Hibon, "The m3-competition: results, conclusions and implications," *Int. Journal of Forecasting*, vol. 16, no. 4, 2000.