An Automated Forecasting Framework based on Method Recommendation for Seasonal Time Series

André Bauer, Marwin Züfle, Johannes Grohmann, Norbert Schmitt, Nikolas Herbst, Samuel Kounev
University of Würzburg, Germany
andre.bauer@uni-wuerzburg.de

ABSTRACT

Due to the fast-paced and changing demands of their users, computing systems require autonomic resource management. To enable proactive and accurate decision-making for changes causing a particular overhead, reliable forecasts are needed. In fact, choosing the best performing forecasting method for a given time series scenario is a crucial task. Taking the "No-Free-Lunch Theorem" into account, there exists no forecasting method that performs best on all types of time series. To this end, we propose an automated approach that (i) extracts characteristics from a given time series, (ii) selects the best-suited machine learning method based on recommendation, and finally, (iii) performs the forecast. Our approach offers the benefit of not relying on a single method with its possibly inaccurate forecasts. In an extensive evaluation, our approach achieves the best forecasting accuracy.

CCS CONCEPTS

• **General and reference → Experimentation**; **Performance**; • **Theory of computation → Unsupervised learning and clustering**; • **Applied computing → Forecasting**.

KEYWORDS

Forecasting, Recommendation, Machine Learning, Feature Engineering, Comparative studies

ACM Reference Format:
André Bauer, Marwin Züfle, Johannes Grohmann, Norbert Schmitt, Nikolas Herbst, Samuel Kounev. 2020. An Automated Forecasting Framework based on Method Recommendation for Seasonal Time Series. In *Proceedings of the 2020 ACM/SPEC International Conference on Performance Engineering (ICPE '20), April 20–24, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3358960.3379123

1 INTRODUCTION

Nowadays, computing systems are pushed to their limits by the fast living and changing requirements of their users. To this end, the autonomic management of these systems is needed. Based on the collected information, these systems can only react to changing requirements with an inherent delay. Thus, integrating time series forecasting into the decision-making process allows to proactively face those changes that cause overheads for their execution.

Indeed, there are different methods, such as statistical methods or machine learning approaches, that support accurate forecasting results. Due to the variety of methods in question, the choice and configuration of the best performing method for a given time series remain to be a mandatory expert task to avoid trial and error. Thus, the question arises if there is a single method that forecasts best for all time series. The "No-Free-Lunch Theorem" [19], initially formulated for optimization problems, denies the possibility of such a method. It states that improving the performance of one aspect leads typically to a degradation in performance for some other aspect.

In fact, various types of hybrid methods have been introduced in recent years to tackle the "No-Free-Lunch Theorem". While statistical models have their difficulties with complex patterns, machine-learning-based methods struggle with non-stationary data (i.e., high variance and trend). To face these weaknesses, we pose ourselves the research question *RQ1: How to build a generic and hybrid forecasting framework for seasonal time series that dynamically minimizes the disadvantages of each component?"*. The core idea is to decompose the time series into multiple parts. The trend is forecast separately by a statistical method while a machine learning method predicts the complex pattern and then assembles the time series. While developing the generic and hybrid forecasting method, we limit ourselves to univariate time series. In fact, correlated/external data can be used for each time series to improve the forecast. However, the selection and preprocessing of such additional information requires domain knowledge. Also, this knowledge of domain-specific feature engineering cannot yet be fully automated. Consequently, our method would have to be tailored to a particular domain and contradict the goal of a more generic approach.

Keeping the "No-Free-Lunch Theorem" in mind, it is not recommended relying on a specific method. Thus, we target a recommendation system that suggests the best-suited machine learning approach for a given time series. Therefore, we pose ourselves the question *RQ2: "What are suitable time series characteristics for the recommendation?"*. Consequently, the question arises *RQ3: "What are suitable approaches for the recommendation system?"*

Towards addressing the questions above, our contribution in this paper is four-fold: (i) We propose an automated forecasting framework for seasonal time series that decomposes a given time series and extracts characteristics (see Section 3.3), recommends the best-suited machine learning method, which is selected on dynamically learned rules, and finally, assembles the components and performs the forecast (see Section 3.1). (ii) To build the recommendation rules dynamically, a knowledge base is built upon a set of historical time series. (iii) For the recommendation, we introduce

ICPE '20, April 20–24, 2020, Edmonton, AB, Canada

ACM ISBN 978-1-4503-6991-6/20/04…$15.00
https://doi.org/10.1145/3358960.3379123

three different approaches and propose our own time series characteristics (see Section 3.2.1). (iv) In a broad evaluation (see Section 4), we analyze the different approaches, investigate the impact of the time series generation, and compare our forecast framework with state-of-the-art forecasting methods.

Without our framework, a simple and straight-forward approach for choosing the best-suited method for a given time series would be based on trial and error, or the consultation of an expert. However, both possibilities are expensive, time-consuming, or error-prone. That is, through the automation of choosing the best method in conjunction with the hybrid approaches leads to good forecasting results and helps saving time and costs.

2 BACKGROUND

Before explaining our approach in detail, we outline some background concepts. Thus, Section 2.1 gives a short introduction to time series. Afterward, the time series decomposition is explained. Finally, the frequency detection and Fourier terms are outlined.

2.1 Time Series

A *univariate time series* is an ordered collection of values of a quantity obtained over a specific period or from a certain point in time. In general, observations are recorded in successive and equidistant time steps (e.g., hours). Typically, internal patterns exist, such as autocorrelation, trend, or seasonal variation.

One of the essential characteristics of a time series is the *stationarity*. Hence, most statistical forecasting methods have the assumption that the time series is either stationary or can be "stationarized" through a transformation. The statistical properties (such as mean, variance, auto-correlation) of a stationary time series do not change over time. Therefore, a stationary time series is easier to model and forecast. In practice, however, time series are usually showing a mix of trend or/and seasonal patterns and are thus non-stationary [1]. To this end, time series are transformed, seasonally adjusted, made trend-stationary by removing the trend, or made difference-stationary by possibly repeated differencing.

2.2 Time Series Decomposition

As a time series consists of different components, a common approach is to break down the time series into its components. These parts can either be used for modifying the data (e.g., removing the trend or the seasonality), or they can be used as intrinsic features (e.g., modeling different recurring patterns).

A common method for decomposing a time series is *STL* (Seasonal and Trend decomposition using Loess) [5]. STL can handle any type of seasonality, allows the seasonal pattern to change over time, and disassembles the given time series into the components trend T, season S, and irregular I (also called remainder). The long-term development in a time series (i.e., upwards, downwards, or stagnate) is called *trend*. Usually, the trend is a monotone function unless external events trigger a break and cause a change in the direction. The presence of recurring patterns within a regular period in the time series is called *seasonality*. These patterns are caused by climate, customs, or traditional habits. The unpredictable part of a time series is called *irregular component*, possibly following a specific statistical noise distribution. It is also considered as the residual time series after all other components have been removed.

2.3 Fourier Terms & Frequency Detection

In many fields, especially for forecasting, it is helpful to know the frequencies, i.e., the lengths of the seasonal patterns. For instance, if the most dominant frequency is unknown for a given time series, the time series cannot be decomposed by the method explained above. By dominant, we mean the most common period, i.e., the seasonal pattern such as days in a year. An established approach for frequency analysis is the Fourier transform, which allows to determine the distribution of frequencies or the *spectral density* of the time series. As a time series can be represented as a weighted sum of sinusoidal components, the found frequencies can be used to retrieves these components, also referred to as *Fourier terms*.

3 APPROACH

As our approach is two-fold, we first introduce the automatic decomposition, feature extraction, and forecasting of a time series. In Section 3.2, we explain the recommendation system for selecting the most suitable machine learning approach. Afterward, the considered time series characteristics are presented. Finally, the used machine learning methods are highlighted.

3.1 Automatic Time Series Forecasting

The assumption of data stationarity is an inherent limitation for time series forecasting. Any time series property that eludes stationarity, such as non-constant mean (trend), seasonality, non-constant variance, or multiplicative effect, poses a challenge for the proper model building. Consequently, we design an automated time series forecasting method that addresses these issues. Figure 1 shows the work-flow of the automatic time series forecasting part. The blue rectangle boxes reflect actions, the green trapezoids machine learning features, the grey rounded boxes the target for the machine learning, and the rounded white boxes everything else. The functioning can be grouped into four steps (dashed red boxes): (i) preprocessing, (ii) recommendation, (iii) forecasting, and (iv) postprocessing. Each part is described in the following.

3.1.1 Preprocessing. This step is responsible for preparing the time series and extracting the intrinsic features for the machine learning algorithm. The first step consists of the frequency estimation. If the time series has a certain frequency, this frequency is chosen. Otherwise, the most dominant frequency is estimated. Next, if the time series has multiplicative effects, the logarithm is used to transform the time series. The Fourier terms (the sine and cosine pair) for the most dominant frequency are determined and used as intrinsic features later on. Although most forecasting methods assume stationary time series, many time series exhibit trend or/and seasonal patterns. To tackle the non-stationarity, our approach decomposes the time series and then handles each part separately. To this end, the time series is decomposed by STL (see Section 2.2) into season, trend, and remainder. The seasonal component is used as an intrinsic feature later on. The remainder is ignored since it is irregular and hard to predict and therefore correlated with a high error rate. Finally, the trend is removed from the time series to

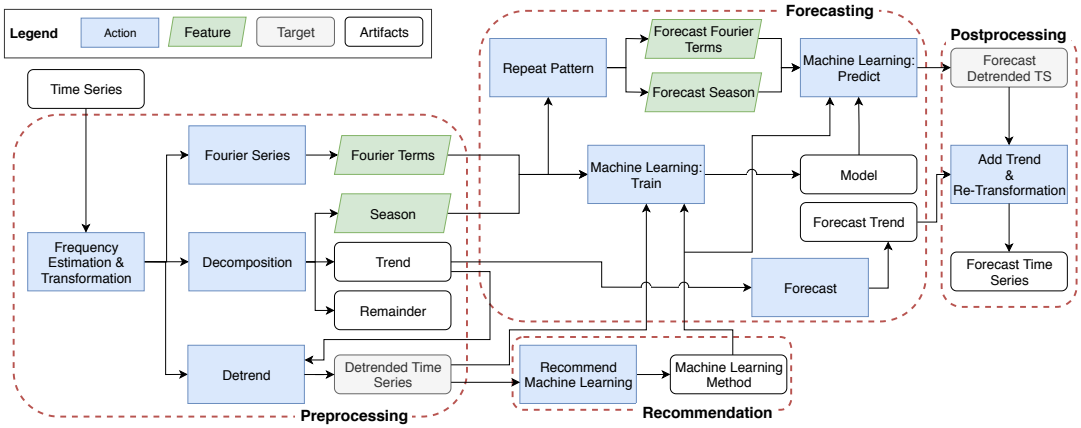

Figure 1: Overview of forecasting process of our framework.

make the time series trend-stationary. The detrended time series is the target value for model building.

3.1.2 Recommendation. The detrended time series is passed from the preprocessing step and is the basis for the recommendation. The recommendation selects which machine learning algorithm is best suited to model the detrended time series. Thus, time series characteristics are extracted from the detrended time series. Based on these characteristics, a suitable machine learning method is selected. The detailed recommendation is explained in Section 3.2.

3.1.3 Forecasting. To build a suitable forecast model that takes the features derived in the previous step into account, we use the machine learning algorithm recommended by the last step. To reduce the model error and later the forecast error, we exclude the trend and the remainder as features. The trend was removed during the first step to make the time series trend-stationary. The remainder of the time series is not explicitly considered a feature. That is, the machine learning method notices a difference that is missing to fully recreate the target value. In other words, this difference is the remainder and is learned implicitly as the machine learning method tries to explain this difference. Consequently, the considered features include the season and the Fourier terms, and the target value corresponds to the detrended time series. Although seasonality can also violate stationarity, time series models usually explicitly take seasonality into account. Also, machine learning methods are suitable for pattern recognition. To this end, we keep the seasonality as a feature.

To forecast the time series, each feature and the trend has to be forecast separately. As the season and the Fourier terms are recurring patterns per definition, these features can merely be continued. Based on the trend component, an ARIMA[1] model [11] without seasonality is determined that forecasts the future trend of the time series. Simultaneously, the forecast patterns of the season and Fourier terms, in combination with the model, are used to predict the detrended time series.

[1]We select ARIMA as it is able to estimate the trend even from a few points, and we use an automatic version that selects the most suited model [10].

3.1.4 Postprocessing. In this last step, the forecast trend is appended to the forecast detrended time series to assemble the forecast time series. Moreover, if the time series was multiplicative, the forecast time series is re-transformed with the exponential function. Finally, the forecast time series is returned.

3.2 Machine Learning Recommendation

To tackle the problem that arises with the "No-Free-Lunch Theorem", we employ a recommendation system for machine learning approaches. The idea is to choose the best suitable method based on the time series characteristics. Figure 2 shows the recommendation work-flow. The blue rectangle boxes reflect actions, the green trapezoids reflect machine learning features, the grey rounded boxes the machine learning target, and the rounded white boxes everything else. The functioning can be grouped into two phases (dashed red boxes): (i) an offline phase and (ii) an online phase. Both phases are described in the following.

3.2.1 Offline Phase. The offline phase learns the rules for recommendation a specific method based on time series characteristics, during the start or if no forecast is currently conducted. To this end, our approach requires an initial set of time series that are stored in the associated storage. To have a broad training set independent of the amount of original time series, the first step in this phase is to create new time series based on the original time series in the storage. For this purpose, three different methods are used:

(i) The first method splits time series into smaller parts to have a more diverse set of time series with different lengths. The length of a split is the maximum between a freely configurable length and 10% of the original length. (ii) The core idea of the second method is to decompose the time series, modify one component, and assemble the modified component and the two remaining parts to a new time series. More precisely, this method modifies each component one after the other and creates, therefore, three new time series. For the modification, the divisors of the frequency of the time series are determined. For each divisor, the components are modified with the proportion of the frequency and the divisor differently: The trend is getting steeper; the season is compressed, i.e., the period length becomes shorter; the remainder is stretched. (iii) The third method

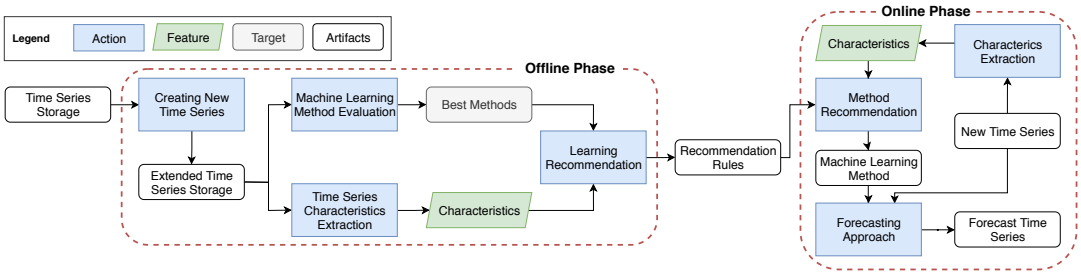

Figure 2: Overview of the recommendation process of our forecasting framework.

also decomposes the time series. More precisely, it combines each component of each time series with each component of the other time series. The length of the resulting time series is equal to the shortest component that was used.

Due to the limitations of STL, which requires at least two full periods, only new time series with a length greater than two times the period plus one are considered valid. Created time series that do not fulfill this requirement are considered invalid and are discarded. This method is able to create a huge training set (including the original time series) with a high diversity of time series characteristics. The rough number of the training set is the number of original time series to the power of three.

After the training set is generated, the time series characteristics (see Section 3.3) of each time series are extracted. As the machine learning methods have to handle the detrended time series, the characteristics are also calculated on the detrended time series. At the same time, the machine learning method evaluation is conducted. During the evaluation, each method (see Section 3.4) performs a forecast for each time series. To this end, the time series is split into history (the first 80% of the time series) and in future (the remaining 20%). For the forecasting, each method gets, as explained in Section 3.1, the Fourier terms, and the season as input while the detrended time series is the target. Then, for each time series and each method, the forecast error, in this case, the mean absolute error (MAPE), is calculated:

$$\text{MAPE} := \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{y_t - f_t}{y_t} \right|. \tag{1}$$

In this equation, n is the forecast horizon, y_t the actual value, and f_t the forecast value. To have a comparable forecast measure among all time series, we normalize for each time series the forecast error with the lowest error. This normalization results in values ≥ 1 for each time series. Further, the best method has a value of 1. We define these values as *forecast accuracy degradation* ϑ showing how much worse the forecast accuracy is compared to the best method. For instance, a forecast accuracy degradation of 1.05 means that the method is 5% worse. Based on the forecast accuracy degradation, the best method for each time series is determined.

Based on the time series characteristics and the best method for each time series, the recommendation rules can be learned. For this purpose, we envision three different approaches:

(i) The first approach A_C is a classification task. That is, a random forest is used to map the time series characteristics for the given time series to the machine learning method with the lowest forecast error. (ii) The core idea of the second approach A_R is to learn how much each method is worse than the best method. In more detail, the approach calculates for each method how much worse this method is compared to the method with the lowest forecast error for given time series characteristics. Then, a random forest is used as a regressor for each machine learning method in question for the selection. In other words, the random forest tries to find a function that learns how much worse the method is in comparison to the best method based on the time series characteristics. After each method has estimated how worse the forecast will be for a new time series, the method with the lowest value is chosen. (iii) The third method is a hybrid approach A_H that combines the first two approaches. More specifically, a random forest regressor is used for each machine learning method available to estimate how much worse the method is in comparison to the method with the lowest error. Then, another random forest is used as a classifier to map the estimation of how worse the forecast will be to the best method. The idea is to minimize the regression error of each method. For example, if one method always claims to have the lowest degradation, but it does not perform as well, the classification shall learn this behavior.

3.2.2 Online Phase. This phase takes place when a forecast for a given time series is conducted. First, the characteristics of the time series are extracted. Then, the recommendation rules are applied to the characteristics, and a machine learning method is selected. Afterward, the forecasting approach (see Section 3.1) performs the forecast. Finally, the time series is saved within the time series storage, and new time series can be generated, as explained in Section 3.2.1.

3.3 Time Series Characteristics

To train a machine learning method for choosing the best method, suitable features are required. Thus, we calculate for each time series a set of characteristics. These characteristics contain information about the time series, statistical measures, characteristics proposed by Wang et al. [18], characteristics proposed by Lemke and Gabrys [13], and characteristics we propose in this work. The used time series characteristics and the associated calculation instructions are listed in Table 1. In contrast to the work of Wang et al., we use the raw values of the characteristics to avoid arbitrary normalization factors.

Table 1: Overview of the considered time series characteristics

Characteristic	Description	Formula
Frequency$^{\diamond}$	The frequency is the length of the most dominant recurring patterns within the time series.	$f = \text{frequency(Y)}$
Length$^{\diamond}$	The total number of observations included in the time series.	$n = \text{length(Y)}$
Standard deviation§	The standard deviations measures the amount of variations within the time series.	$\sigma = \sqrt{var(Y)}$
Skewness§	The skewness measures the symmetry of the value distribution of the time series.	$\frac{1}{n \cdot \sigma^3} \sum_{k=1}^{n} (Y_t - \overline{Y})^3$
Kurtosis§	Kurtosis is a measure of the tailedness of the value distribution of the time series.	$\frac{1}{n \cdot \sigma^4} \sum_{k=1}^{n} (Y_t - \overline{Y})^4$
Remainder SD§	The stand deviation of the remainder.	$\sigma_R = \sqrt{var(R)}$
Remainder skewness§	The skewness of the remainder.	$\frac{1}{n \cdot \sigma_R^3} \sum_{k=1}^{n} (R_t - \overline{R})^3$
Remainder kurtosis§	The kurtosis of the remainder.	$\frac{1}{n \cdot \sigma_R^4} \sum_{k=1}^{n} (R_t - \overline{R})^4$
Proportion remainder$^{\sharp}$	This characteristic reflects how strongly the remainder is prominent in the time series.	$\frac{QR(R)}{QR(R) + QR(S)}$
Proportion season$^{\sharp}$	This characteristic reflects how strongly the season is prominent in the time series.	$\frac{QR(S)}{QR(R) + QR(S)}$
Mean period entropy$^{\sharp}$	This characteristic quantifies the regularity and unpredictability of fluctuations of the time series. For this purpose, the approximate entropy of each period is calculated and then averaged.	$m = \frac{1}{n} \sum_{k=1}^{\lfloor n/f \rfloor} EA(p_i)$
Coefficient of entropy variation$^{\sharp}$	The coefficient measures the standardized entropy distribution over all periods.	$\frac{\sqrt{var(EA(p_i))}}{m}$
Mean cosine similarity$^{\sharp}$	This characteristic describes how similar all periods are. To this end, the cosine similarity of each pair of periods is calculated. Then, the average similarity is determined.	$\frac{2 \sum_{i=1}^{\lceil n/f \rceil} \sum_{j=1;j\neq i}^{\lceil n/f \rceil} \frac{p_i \cdot p_j}{\lvert\lvert p_i \rvert\rvert \cdot \lvert\lvert p_j \rvert\rvert}}{(\lceil n/f \rceil - 1)^2 + (\lceil n/f \rceil - 1)}$
Durbinwatson$^{\sharp}$	This characteristic quantifies how well the seasonal pattern can be approximated by a sinus wave. The Durbin-Watson test is used to check the auto-correlation of the fitter errors.	$\frac{\sum_{t=2}^{n} (e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2}$
Seasonality†	This characteristic reflects the strength of the season within the time series.	$1 - \frac{var(R)}{var(Y)}$
Serial correlation†	This characteristic describes the correlation of the time series with itself to an earlier time. To determine the serial correlation, the Box-Pierce statistics is used.	$n \cdot \sum_{k=1}^{f} r_k(Y)^2$
Remainder serial corr.†	The serial correlation of the remainder.	$n \cdot \sum_{k=1}^{f} r_k(R)^2$
Non-linearity†	This characteristic describes how badly the time series can be written as a linear combination of unknown variables or functions.	R.t. Wang et al. [18, p. 18]$^{(*)}$
Remainder non-lin.†	The non-linearity of the remainder.	R.t. Wang et al. [18, p. 18]$^{(*)}$
Self-Similarity†	This characteristic describes how similar an object is to a part of itself by the Hurst exponent.	R.t. Wang et al. [18, p. 20]$^{(*)}$
Chaos†	The chaos calculates the randomness within the time series via the Lyapunov exponent.	R.t. Wang et al. [18, p. 20]$^{(*)}$
2nd freq‡	The second dominant frequency of the time series.	–
3rd freq‡	The third dominant frequency of the time series.	–
Max spec‡	The maximal spectral value of the spectral density.	–
Num peaks‡	This characteristic reflects how many strong recurring patterns the time series has. To this end, the number of peaks in the spectral density that have at least 60% of the maximum value.	–

Let Y be the time series without trend, S be the season component of the time series, R be the remainder of the time series, $r(x)_k$ be the auto-correlation function with lag k, $QR(x) := Q_{0.95}(x) - Q_{0.05}(x)$ be the range between the 95% percentile and 5% percentile, p_i be the i-th period of Y, $EA(x)$ be the approximated entropy [14, p.3], e_t the fitting error at time t.

$^{(*)}$ We modify the approach and use the time series without trend for the calculation.

$^{\diamond}$ time series information, § statistical measure, $^{\sharp}$ proposed by this work, † proposed by Wang et al. [18], ‡ proposed by Lemke and Gabrys [13]

3.4 Machine Learning Methods

For the forecasting task, we only consider machine learning methods in this paper as statistical methods such as ARIMA can typically only process the time series without additional information. This means that the extracted features (see Section 3.1) cannot be used by such methods. In addition, ML methods can handle any number of features. That is, for a possible extension of our approach with external information, these features can be added. The used machine learning methods (see Section 3.1) are listed in the following: (i) *Catboost* applies gradient boosting of decision trees [15]. (ii) *Cubist* is a regression model that combines the ideas of M5 with additional corrections as described by Quinlan [16]. (iii) *Evtree* implements an evolutionary algorithm for learning globally optimal classification and regression trees [9]. (iv) NNetar is a feed-forward neural network is trained with lagged values of the time series [10]. (v) *Random Forest* (RF) uses bagging for generating samples from the data set used for learning [2]. (vi) *Rpart* trains a regression tree using recursive partitioning, based on the CART algorithm by Breiman et al. [3]. (vii) *Support Vector Regression* (SVR) uses the same principles as SVM for classification [8]. (viii) *XGBoost* uses gradient tree boosting where trees are generated sequentially. That is, each tree is grown with knowledge from the last trained tree [4].

4 EVALUATION

Before discussing the evaluation, we introduce the used data set in Section 4.1. Then, we explain the methodology and the evaluation metrics in Section 4.2. Afterwards, we analyze how well the different machine learning methods perform on the data set. Based on this information, we evaluate our recommendation approaches in Section 4.4. In Section 4.5, we investigate how the diversity of the data set is increased by the time series generation. Finally, we compare our forecasting framework with state-of-the-art methods.

4.1 Data Set

To have a sound and broad evaluation of our approach, a highly heterogeneous data set that covers different domains and characteristics is required. Indeed, there are numerous data sets available online: competitions (e.g., NN3[2], M3[3], and M4), kaggle, R packages, and many more. Although, for instance, the M4 competition set contains 100,000 time series, these time series have low frequencies (1, 4, 12, and 24) and short forecasting horizons (6 to 48 data points). Further, the median length of a time series is 106. That is, we assume that if the data set is used alone, it is not suitable for benchmarking forecasting methods for all kinds of domains.

To this end, our data set[4] consists of 150 real-world and publicly available time series. The time series are collected from various sources including Wikipedia Project-Counts, Internet Traffic Archive, R packages, Kaggle, Datamarket, and many more. Further, the data set reflects different use cases, e.g., Internet accesses, sales volume, etc. Moreover, our data set covers the same frequencies as the M4 competition and additional frequencies (7, 48, 52, 60, 96, 144, 168, 365, 2160, and 6480). Further, our forecast horizons range from 8 to 7,304 data points, and the median length is 595.

4.2 Evaluation Methodology

To evaluate our approach, we divide the original data set into 100 training time series and 50 validation time series. To avoid an arbitrary split, we divide the data set in 100 unique splits. In other words, we train and evaluate our approach on 100 different time series train and test sets. We also made sure that all time series are spread across all splits.

As described in Section 3.2.1, our approach expands for each split the size of the training set to have a sound training set for the recommendation. That is, our approach uses in each division the 100 time series for the generation of new time series. In contrast to the description of the approach, we restrict the approach to use only 10,000 instead of the roughly 1,000,000 time series. More precisely, the training data in each split contains the original 100 time series and 9,900 new time series.

4.3 Machine Learning Method Analysis

For reference, we investigate how each of the chosen machine learning methods performs in the forecasting process on the data set, without recommendation, i.e., changing the method depending on the input time series. To this end, we observe for each method how often the method (i) is the best method in each split (best method in split), (ii) has on average the lowest forecast accuracy degradation in each split (on avg. lowest error in split), and (iii) is over all time series the best method (total best method). We report the respective percentages in Table 2 showing these three observations for the training data and test data for each method. While the distribution of percentages of which method is the best over all time series is almost similar for the training and test data, the distributions per split differ considerably. While *Nnetar* was in every split the method achieving the best training forecast accuracy the most often, it reaches only in 73% of the test data splits the same performance. *Cubist* had in 55% of the training splits on average the lowest forecast accuracy degradation. In the test data, *Cubist* has in only 17% of the splits on average the lowest forecast accuracy degradation.

In a nutshell, we see from these results that the dynamic choice of the best performing method is a crucial task with significant potential. Even choosing a method based on straight-forward metrics (for instance, choosing the method which was on average the best method in the training data) based on the training data may lead to a bad performance.

4.4 Evaluation of the Recommendation

As the recommendation of the best suitable method is an essential pillar of our forecasting framework, we examine the recommendation performance of our envisioned approaches (see Section 3.2.1). To have a ground truth for the competition, we define the following three method selecting strategies: (i) Selecting the best method for each time series a-posteriori S^*. (ii) Selecting the method which had the lowest average forecast accuracy degradation in each training split S_L. (iii) Selecting the method, which was most often the best method in each training split S_B. Note that, based on our analysis in Section 4.3, the method *Nnetar* will be chosen.

The results of the comparison between these six methods are presented in Table 3. For each approach/strategy, this table lists the median, average and standard deviation of the accuracy degradation ϑ over all 100 splits.

The best values are shown by S^*. Indeed, this result is not surprising as this strategy has a-posteriori knowledge. Thus, this method has the role of showing the theoretically best possible values. In other words, S^* is the base-line for the recommendation. Consequently, only five methods remain for a fair competition. In terms of the average forecast accuracy degradation, the regression-based approaches (A_H being on average 15.9% worse than always choosing the best method and A_R with a value of 1.172) outperform the remaining approaches/strategies. While taking also the median and the standard deviation of the forecast accuracy degradation into account, it can be seen that the meta-learning layer of A_H is able to improve the performance of A_R in all measures of the forecast accuracy degradation. The worst forecast accuracy degradation is shown S_L followed by A_C. In contrast, A_C exhibits the lowest median followed by the regression-based approaches. The worst median is shown by S_B. While observing the standard deviation of the forecast accuracy degradation, S_L, A_C, A_R exhibit high values. The lowest value is shown by A_H.

The median and mean values can be better understood if the distribution of the ranking of the recommended methods is taken

[2]NN3 competition: http://www.neural-forecasting-competition.com/NN3/
[3]M3 competition: https://forecasters.org/resources/time-series-data/m3-competition/
[4]Time series data set available at https://zenodo.org/record/3508552

Table 2: Investigation of the forecast performance of the different machine learning methods.

		Catboost	Cubist	Evtree	Nnetar	RF	Rpart	SVR	XGBoost
(Train / Test)	Best method in split	(0% / 21%)	(0% / 3%)	(0% / 0%)	(100% / 73%)	(0% / 0%)	(0% / 0%)	(0% / 3%)	(0% / 0%)
	On avg. lowest error in split	(0% / 11%)	(55% / 17%)	(0% / 2%)	(0% / 5%)	(0% / 5%)	(45% / 43%)	(0% / 0%)	(0% / 17%)
	Total best method	(7.2% / 18.3%)	(13.5% / 13.1%)	(5.9% / 9.2%)	(38.1% / 23.9%)	(4.5% / 4.2%)	(12.3% / 8.1%)	(9.2% /16.8%)	(9.0% / 5.3%)

Table 3: Comparison of the recommendation methods.

	S^*	S_L	S_B	A_C	A_R	A_H
Avg. ϑ	1.000	1.409	1.235	1.249	1.172	1.159
Median ϑ	1.000	1.045	1.076	1.016	1.035	1.032
SD ϑ	0.000	3.674	0.427	2.458	1.382	0.382

into account. Figure 3 shows the distribution of the rankings. The ranks of S_L are almost equally distributed. S_B selects almost either the best or the worst method. More precisely, it recommends in 51.4% of the time series the worst method. For all recommendation approaches, the distribution of ranks two to five drops. The regression-based approaches select in more than 30% the best or second-best method. However, choosing the worst method is almost as likely as choosing the best method. In contrast to all other methods, A_C chooses with more than 50% the best, second, or third best method, but also has almost 25% of choosing the worst method. In fact, none of the methods show a proper distribution, which decreases with increasing rank.

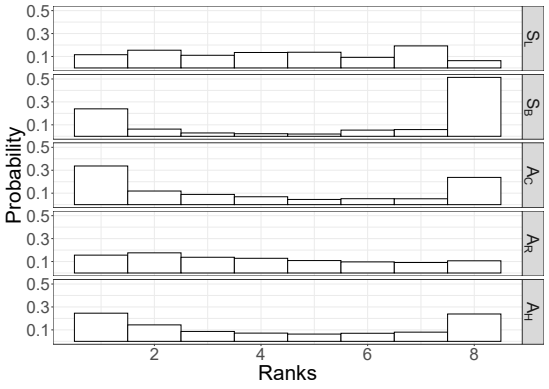

Figure 3: Distribution of the rankings.

4.5 Evaluating the Time Series Generation

One central problem of machine learning is the inherent limitation to predict only what has been learned during the training phase. In other words, machine learning methods have a limited ability for extrapolation. This also holds true for our recommendation. Consequently, we try to consider as many time series with different characteristics as possible to improve the recommendation for unknown time series. Thus, we analyze in this section how the new time series generation affects the diversity of the time series characteristics. To this end, we collect for each time series characteristic the values from the original data and the new generated time series. Then, we normalize with a min-max-scaling for each time series characteristic the data between 0 and 1 for a comparable analysis. On top of this, we depict each characteristic in a spider chart (see Figure 4). In this diagram, the maximal value of new data (grey) and original data (purple), and the minimum values of the new data (green) and original data (blue) are shown. Each edge of this chart represents a time series characteristics. For almost all characteristics, the new generated time series expand the spectrum of the data both in terms of the maximum value and minimum value.

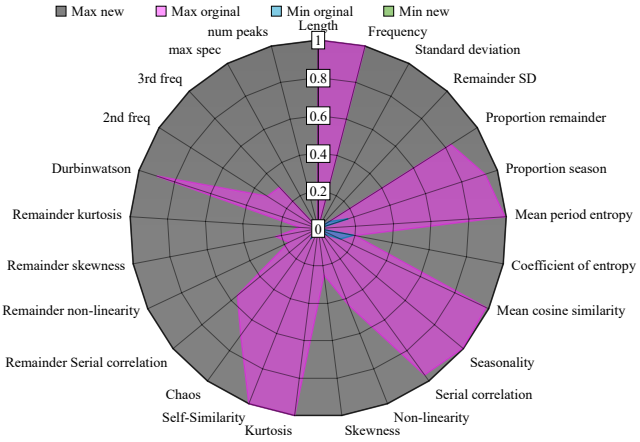

Figure 4: Time series generation result.

4.6 Evaluation of Forecast Accuracy

To investigate how well our forecasting framework performs, we compare the forecasting error (i.e., MAPE) of our approach with three state-of-the-art approaches that are briefly described in the following: *ETS* [12] is a statistical method and builds an exponential smoothing state space model consisting of trend, season, and error. Each component can be combined in an additive or multiplicative manner, or it may be skipped. *tBATs* [7] extends ETS using a trigonometric representation based on Fourier series for the season and an ARMA model for the error. Further, the data is transformed with a Box-Cox transformation. *sARIMA* [11] determines the orders of the autoregressive model, the moving average model, and the differentiation. *sARIMA* models one seasonal pattern, and each non-seasonal component of the ARIMA model is extended with its seasonal counterpart. Table 4 lists the average, median, standard deviation of the forecast error for all 100 splits. Each of our approaches exhibits a lower average MAPE and standard deviation than the state-of-the-art methods. The worst average MAPE (56.96%) is achieved by *ETS*.

In contrast, *tBATS* has the lowest median MAPE (10.83%) followed by A_C (12.31%) while *ETS* again shows the highest median error. To sum up, our approaches are equally accurate in terms of the median forecast error, but having a lower average and standard deviation forecast error than the state-of-the-art methods.

Table 4: Comparison of the forecast error.

MAPE	A_C	A_R	A_H	ETS	tBATS	sARIMA
Avg.	24.40	23.26	23.68	56.96	36.28	28.12
Median	12.31	13.07	13.18	14.47	10.83	13.00
SD	50.31	40.41	38.52	136.22	98.68	64.72

5 RELATED WORK

To face the "No-Free-Lunch Theorem", i.e., minimizing the variance of monolithic forecasting methods, many hybrid mechanisms and forecast recommendation systems have been developed. The first idea of selecting a forecasting method based on rules was introduced by Collopy and Armstrong in 1992 [6]. In their work, they manually created an expert system. The rules based on 18 time series characteristics and include four methods. However, this rule set was created by human experts, and each modification requires human interaction. In 2009, Wang et al. introduced two approaches for forecasting method recommendation [18]. Firstly, they propose hierarchical clustering and self-organizing maps; secondly, a decision tree technique is applied. The generate rules based on 13 time series characteristics and covers four methods. Unfortunately, the proposed rules were not evaluated. In 2010, Lemke and Gabrys investigated the applicability of different meta-learning approaches [13]. In their work, they use 17 time series and six error characteristics while using eighth methods and seven combination approaches. In 2018, Talagala et al. propose in a techpaper a feature-based forecast-model selection [17]. To this end, they simulate time series that are generated by fitting exponential smoothing and ARIMA models to the original data. A random forest classifier is then used to map 25 to 30 time series characteristics (depending on the time series) to the best forecast method. In their work, they consider seven methods. As the work of Wang et al. [18] were not evaluated, Züfle et al. investigate and compare these rules to two proposed dynamic recommendation algorithms [20].

In contrast to the related work that only introduce the selection of the best forecasting method, we propose an overarching framework that combines the selection of the best method and the forecast itself. While the aforementioned works use solely statistical methods, the focus in this work lies in machine learning-based regressor methods. Further, for the evaluation, we use a highly diverse data set. Further, our selection mechanism creates also new time series by combining actual time series to increase the diversity of the data set.

6 CONCLUSION

In this work, we propose an automated forecasting framework that (i) extracts characteristics from a given time series, (ii) selects the best-suited machine learning method based on recommendation, and finally, (iii) performs the forecast. Our approach offers the benefit of not relying on a single method with its possibly inaccurate forecasts. For the recommendation of the best-suited method, we introduce three different approaches, and in addition to time series characteristics from the literature, we propose our own characteristics. In an extensive evaluation, we compare the three proposed recommendation approaches, the impact of time series generation, and compare the forecasting framework with state-of-the-art methods. Although the proposed recommendation approaches perform equally good, our approach achieves the best forecasting accuracy in comparison with the state-of-the-art techniques.

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] Ratnadip Adhikari and R. K. Agrawal. 2013. An Introductory Study on Time Series Modeling and Forecasting. *CoRR* abs/1302.6613 (2013).
[2] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[3] Leo Breiman, Joseph H Friedman, R. A. Olshen, and C. J. Stone. 1983. Classification and Regression Trees.
[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *ACM SIGKDD 2016*. ACM, 785–794.
[5] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* 6, 1 (1990), 3–73.
[6] Fred Collopy and J Scott Armstrong. 1992. Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations. *Management Science* 38, 10 (1992), 1394–1414.
[7] Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. 2011. Forecasting time series with complex seasonal patterns using exponential smoothing. *J. Amer. Statist. Assoc.* 106, 496 (2011), 1513–1527.
[8] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. 1997. Support vector regression machines. In *Advances in neural information processing systems*. 155–161.
[9] Thomas Grubinger, Achim Zeileis, and Karl-Peter Pfeiffer. 2014. evtree: Evolutionary Learning of Globally Optimal Classification and Regression Trees in R. *Journal of Statistical Software, Articles* 61, 1 (2014), 1–29.
[10] Rob Hyndman, George Athanasopoulos, Christoph Bergmeir, Gabriel Caceres, Leanne Chhay, Mitchell O'Hara-Wild, Fotios Petropoulos, Slava Razbash, Earo Wang, and Farah Yasmeen. 2018. *forecast: Forecasting functions for time series and linear models*. http://pkg.robjhyndman.com/forecast R package version 8.4.
[11] Rob J Hyndman and George Athanasopoulos. 2014. *Forecasting: principles and practice*. OTexts, Melbourne, Australia.
[12] Rob J Hyndman, Anne B Koehler, Ralph D Snyder, and Simone Grose. 2002. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of forecasting* 18, 3 (2002), 439–454.
[13] Christiane Lemke and Bogdan Gabrys. 2010. Meta-learning for time series forecasting and forecast combination. *Neurocomputing* 73, 10-12 (2010), 2006–2016.
[14] Steven M Pincus, Igor M Gladstone, and Richard A Ehrenkranz. 1991. A regularity statistic for medical data analysis. *Journal of clinical monitoring* 7, 4 (1991), 335–345.
[15] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*. 6638–6648.
[16] J Ross Quinlan. 1993. Combining instance-based and model-based learning. In *Proceedings of the tenth international conference on machine learning*. 236–243.
[17] Priyanga Talagala, Rob Hyndman, George Athanasopoulos, et al. 2018. *Meta-learning how to forecast time series*. Technical Report. Monash University, Department of Econometrics and Business Statistics.
[18] Xiaozhe Wang, Kate Smith-Miles, and Rob Hyndman. 2009. Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing* 72, 10âĂŞ12 (2009), 2581 – 2594.
[19] D. H. Wolpert and W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (Apr 1997), 67–82.
[20] Marwin Züfle, André Bauer, Veronika Lesch, Christian Krupitzer, Nikolas Herbst, Samuel Kounev, and Valentin Curtef. 2019. Autonomic Forecasting Method Selection: Examination and Ways Ahead. In *Proceedings of the 16th IEEE International Conference on Autonomic Computing (ICAC)*. IEEE.