
TELESCOPE: AN AUTOMATED HYBRID FORECASTING APPROACH ON A LEVEL-PLAYING FIELD

André Bauer
University of Chicago
Chicago, United States of America
andrebauer@uchicago.edu

Mark Leznik
University of Ulm
Ulm, Germany
mark.leznik@uni-ulm.de

Michael Stenger, Robert Leppich, Nikolas Herbst, Samuel Kounev
Affiliation
City
{firstname}. {lastname}@uni-wuerzburg.de

Ian Foster
University of Chicago
Chicago, United States of America
foster@cs.uchicago.edu

ABSTRACT

In many areas of decision-making, forecasting is an essential pillar. Consequently, many different forecasting methods have been proposed. From our experience, recently presented forecasting methods are computationally intensive, poorly automated, tailored to a particular data set, or they lack a predictable time-to-result. To this end, we introduce Telescope, a novel machine learning-based forecasting approach that automatically retrieves relevant information from a given time series and splits it into parts, handling each of them separately. In contrast to deep learning methods, our approach doesn't require parameterization or the need to train and fit a multitude of parameters. It operates with just one time series and provides forecasts within seconds without any additional setup. Our experiments show that Telescope outperforms recent methods by providing accurate and reliable forecasts while making no assumptions about the analyzed time series.

Keywords Automatic feature extraction · Combining forecasts · Comparative studies · Forecasting competitions · Long term time series forecasting · Time series

1 Introduction

Time series forecasting is an essential pillar in many decision-making disciplines [1]. Accordingly, time series forecasting is an established and active field of research, and thus various methods have been proposed. Due to the variety of approaches, the choice and configuration of the best performing method for a given time series remain a mandatory expert task to avoid trial-and-error. However, expert knowledge can be expensive, may have a subjected bias, and it can take a long time to deliver results.

Thus, the question arises if there is a single forecasting method that performs best for all time series. However, the “No-Free-Lunch Theorem” [2], initially formulated for optimization problems, denies the possibility of such a method. It states that improving one aspect typically leads to a degradation in performance for another aspect. An analogy can be drawn to the domain of forecasting: No forecasting method performs best for all time series. In other words, forecasting methods have their advantages and drawbacks depending on the considered time series.

In fact, different types of hybrid forecasting methods have been proposed in the last years [3] to tackle the challenge stated by the “No-Free-Lunch Theorem”. The core idea is the usage of at least two methods to minimize the disadvantages of individual methods. From our experience, recently presented open-source hybrid methods are computationally intensive, poorly automated, tailored to a particular data set, or they lack a predictable time-to-result. However, many real-world scenarios where forecasting is useful (e.g., auto-scaling) have strict requirements for a reliable time-to-result and forecast accuracy. To achieve a low variance in forecast accuracy, the preprocessing of historical data and the feature handling (extraction, engineering, and selection) must be done in a sophisticated way.

To tackle the mentioned challenges, we pose ourselves the following research questions:

RQ1 *How to design an automated and generic hybrid forecasting approach that combines different forecasting methods to compensate for the disadvantages of each technique?*

RQ2 *How to automatically extract and transform features of the considered time series to increase the forecast accuracy?*

RQ3 *What are appropriate strategies to dynamically apply the most accurate method within the hybrid forecasting approach for a given time series?*

Towards addressing the research questions, we introduce a novel machine learning-based hybrid forecasting approach called *Telescope*¹² that automatically retrieves relevant information from a given time series and splits it into parts, handling each of them separately (addressing RQ1). More precisely, Telescope automatically extracts intrinsic time series features and then decomposes the time series into components, building a forecasting model for each of them (addressing RQ2). Each component is forecast by applying a different method and then the final forecast is assembled from the forecast components by employing a regression-based machine learning algorithm. For non-time-critical scenarios, we additionally provide an internal recommendation system that can be employed to automatically select the most appropriate machine learning algorithm for assembling the time series from its components (addressing RQ3).

To evaluate our approach, we compare Telescope to seven competing methods (including approaches from Uber and Facebook) in more than 1000 hours of experiments using a forecasting benchmark. Telescope outperformed all methods, exhibiting the best forecast accuracy coupled with a low and reliable time-to-result. Compared to the competing methods that exhibited, on average, a forecast error (more precisely, the symmetric mean absolute error) of 29%, Telescope exhibited an error of 20% while being 2556 times faster. In particular, the methods from Uber and Facebook exhibited an error of 48% and 36%, and were 7334 and 19 times slower than Telescope, respectively. When additionally applying the recommendation system, Telescope was able to reduce the forecast error even further down to 19%.

The rest of the article is structured as follows: We first introduce terms as well as definitions for understanding this article in Section 2. Then, we review related work in Section 3. In Section 4, we present Telescope. Afterward, we benchmark Telescope in Section 5 before concluding this article.

2 Foundations on Time Series

In this section, we briefly present the basic terms and definitions related to time series and time series in Section 2.1. Then, we outline the frequency detection of a time series, the time series decomposition, and the time series transformation in Section 2.2, 2.3, and 2.4, respectively.

2.1 Terms and Definitions

A *univariate time series* is an ordered collection of values of a quantity obtained over a specific period or since 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. Mathematically, if $y_t \in \mathbb{R}$ is the observed value at time t and T a discrete set of equidistant time points, a univariate time series is defined by

$$Y := \{y_t : t \in T\}. \quad (1)$$

2.1.1 Components of a Time Series

A time series can also be seen as a composition of trend, seasonal, cycle, and irregular components [1]. The long-term development in a time series (i.e., upwards, downwards, or stagnate) is called *trend*. Usually, the trend is a monotonic 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 typically caused by climate, customs, or traditional habits such as night and day phases. The length of a seasonal pattern is called frequency³. Rises and falls within a time series without a fixed frequency are called *cycles*. In contrast to seasonality, the amplitude and duration of the cycles vary over time. The remaining part of the time series that is not described by trend, seasonality, or cycles is called the *irregular component*. It usually follows a certain statistical noise distribution and is therefore not predictable by most models.

¹Telescope at GitHub: <https://github.com/DcartesResearch/telescope>

²A preliminary idea of our Telescope approach were accepted as a short conference publication [4].

³In the context of time series analysis, the term frequency has a different meaning as, for example, in physics.

2.1.2 Fourier Terms

Following the basic principle in mathematics to break down complex objects into more simpler parts, a time series can be approximated by a linear combination of sinusoid terms. The resulting representation is referred to as *Fourier series* and the sinusoids terms are called *Fourier terms*.

2.1.3 Stationarity

One of the most important characteristics of a time series is the *stationarity* since most forecasting methods assume that the time series is either stationary or can be “stationarized” through a transformation [5]. Loosely speaking, the statistical properties (such as mean, variance, and auto-covariance) of a stationary time series do not change over time. In practice, however, time series are usually showing a mix of trend or/and seasonal patterns and are thus non-stationary [6]. To this end, time series are transformed, seasonally adjusted, made trend-stationary by removing the trend, or made difference-stationary by possibly repeated differencing (i.e., computing the differences between consecutive observations) [1].

2.2 Spectral Analysis and Frequency Detection

In many fields, especially for forecasting, it helps discover the underlying periodicities in the data, that is, knowing the frequencies or 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 (see Section 2.3). In this context, the dominant frequency means the most common period, respectively, the seasonal pattern, such as days in a year. Also, if the dominant frequency is available, the information on the next dominant frequency (e.g., the week within the year) is helpful. A standard method for this data analysis is the *spectral analysis*, also referred to as analysis in the frequency domain. The key idea is to transform the time series from the time domain to the frequency domain as the spectrum reveals the data’s underlying frequencies. The resulting *spectral density* assigns then an intensity to each frequency within the time series. An estimate of the spectral density is the *periodogram* [7]. That is, in a time series that is driven by certain seasonal patterns, the periodogram shows peaks at precisely those frequencies.

2.3 Time Series Decomposition

As a time series consists of different components, a common approach is to break down the time series into its components. The parts can either be used to modify the data (e.g., removing the trend or seasonality) or be used as intrinsic features for augmenting a model capturing the time series.

A common method for decomposing a time series is *STL* (Seasonal and Trend decomposition using Loess) [8]. STL disassembles the given time series $Y(t)$ into the components trend $T(t)$, season $S(t)$, and irregular $I(t)$. More formally, the resulting decomposition of STL is

$$Y(t) := T(t) + S(t) + I(t). \quad (2)$$

Although STL can only handle additive relationships between the components, it can also be applied to multiplicative time series by applying the natural logarithm on the time series beforehand. Figure 1 depicts a decomposition of a time series based on STL.

2.4 Time Series Transformation

As observed data may be quite complex, for example, having high variance and/or multiplicative relationship between the components, an adjustment or simplification of it can improve the forecasting model [1]. To this end, there are different methods that transform time series. A common approach is to apply the logarithm, but the transformed data may not be normally distributed. In contrast, the *Box-Cox transformation* [9] tries to transform the data into “normal shape”. The Box-Cox transformation is defined as follows

$$w_t := \begin{cases} \ln y_t & \text{if } \lambda = 0, \\ (y_t^\lambda - 1)/\lambda & \text{otherwise,} \end{cases} \quad (3)$$

where y_t is the original time series and λ the transformation parameter that determines the function.

Note that if a forecast was conducted based on this transformation, the forecast values have to be re-transformed using the same λ to be in the right scale. Consequently, the re-transformation of the time series is defined as

$$y_t := \begin{cases} e^{w_t} & \lambda = 0, \\ (\lambda w_t + 1)^{1/\lambda} & \text{otherwise.} \end{cases} \quad (4)$$

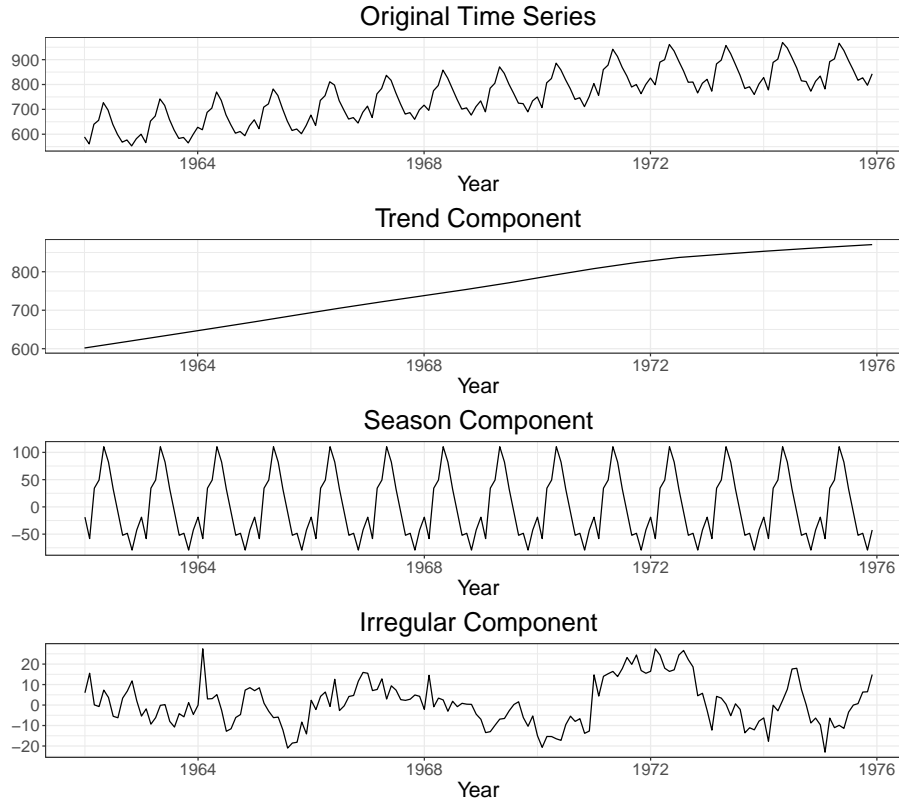


Figure 1: Example of STL decomposition.

3 Related Work

In 1997, the “No-Free-Lunch Theorem” was postulated. It states that there is not a single optimization algorithm that performs best for all scenarios since improving the performance of one aspect leads typically to a degradation in performance for another aspect. Considering the inherent drawbacks and limitations of forecasting methods, it can be concluded that there is no single forecasting method that performs best for all kinds of time series. To face this challenge, many hybrid forecasting methods have been proposed in the literature. The underlying idea of such hybrid approaches is to use at least two forecasting techniques to compensate for the limitations of the individual forecasting approaches. The success of this concept can be demonstrated, for example, by investigating the recent M4-Competition [10]: 12 of the 17 most accurate methods were hybrid forecasting methods. The proposed hybrid methods can be categorized into three groups, each sharing the same basic concept: (i) *ensemble forecasting*, (ii) *forecasting method recommendation*, and (iii) *time series decomposition*. Consequently, Sections 3.1 to 3.3 present top cited and recent hybrid approaches. The delimitation of Telescope from the following approaches is discussed in Section 3.4.

3.1 Ensemble Forecasting

The core idea of the first and historically oldest group is to compute the forecast as a weighted sum of the values derived from applying multiple forecasting methods. The impetus for this idea was provided by J. Bates and C. Granger in 1969 [11]. In their work, they show that the forecast accuracy can be increased by using ensembles compared to applying each forecasting method on its own. However, the accuracy of ensemble forecasting depends highly on the weight estimation algorithm. Recent work applies a weighted sum based on the training error [12] or starts with a simple arithmetic mean of the forecasts and learns then the optimal weights for each forecasting method dynamically [13]. Another approach [14] applies a meta-learner to determine the weights based on each method’s forecast error based on the training set. Z. Wang et al. [15] propose an ensemble of 30 feed-forward neural networks, where each network gets a random subset of the time series. D. Boulegane et al. [16] builds an ensemble base on Bernoulli trial that takes the predicted error and the meta-learner’s confidence for each method into account. P. Montero-Manso et al. [17] applies a meta-learner that learns the ensemble weights based time series characteristics and the forecast error of each method.

3.2 Forecasting Method Recommendation

Methods from the second group build a rule set for estimating the assumed best forecasting method based on analyzing specific characteristics of the considered time series characteristics. The first rule for weighting forecasting methods based on the characteristics of the given time series was introduced F. Collopy and J. Armstrong in 1992 [18]. In their work, the authors manually created an expert system after having interviewed five experts in the field of forecasting from industry and academia. A few years later, X. Wang et al. used clustering and self-organizing maps to select the best forecasting method based on time series characteristics [19]. This work was extended by A. Widodo and I. Budi [20] by adding more methods. In addition to time series characteristics, M. Kück et al. [21] considered for selecting the best forecasting method for a given time series a set of error measures. In another approach [22], the time series are classified into different clusters and the method that has the lowest prediction error in the cluster in which the new time series is classified is recommended. T. Talagala et al. [23] applied a meta-learner to map the best forecasting method to a given time series based on its characteristics. A similar approach is proposed by D. Zhang et al. [24], where the forecast horizon is also considered for the meta-learning.

3.3 Time Series Decomposition

In the last group, a time series is decomposed into components and forecasting methods are applied to each component separately or a time series is forecast with an individual method, and afterward, a second individual method is applied on the residuals. Recent approaches decompose the time series into a linear and non-linear part based on fitting residuals [25, 26], discrete wavelet transformation [27], or empirical mode decomposition [28, 29]. C. Bergmeir et al. [30] decomposed the time series with STL and then, bootstrapped different versions of the irregular part to form new time series. Each new time series is forecast and the final forecast is the median of these forecasts. S. Taylor and B. Letham (Facebook) [31] split the time series into trend, season, and holiday. F. Saâdaoui and H. Rabbouch proposed an approach that splits a time series into trend, seasonal, and irregular parts based on the maximal overlap discrete wavelet transformation [32] and one approach approximating the trend with a linear regression and seasonal component with a sum of sines [33]. The method [34] introduced by S. Sym1 (Uber) deploys exponential smoothing for de-seasonalizing and a recurrent neural network for extrapolating the time series.

3.4 Differentiation from Related Work

One of the key differences to the proposed hybrid methods [15, 18, 24, 28, 30, 31, 32, 34] is that Telescope is generic. That is, Telescope is not tailored to a special scenario and makes no assumptions about the analyzed time series. Further, as Telescope is based on time series decomposition and its recommendation part is only used in non-time-critical scenarios, our approach differs considerably from methods from the field of ensemble forecasting. In contrast to the methods originating from time series decomposition that use different decomposition and forecasting techniques, Telescope explicitly decomposes the time series into trend, season, and irregular part. Also, each part is forecast separately using different forecasting methods. In contrast to the recommendation-based methods that use mainly “classical” time series forecasting methods, Telescope selects regression-based machine learning methods based on time series characteristics. Lastly, our approach augments the original time series set by generating new time series from it to increase the data set’s diversity.

4 The Telescope Approach

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 (i.e., trend), seasonality, non-constant variance, or multiplicative effect, poses a challenge for the proper model building [35]. Consequently, we take all the techniques discussed in the previous section into account to design an automated forecasting workflow called Telescope that automatically transforms the given time series, derives intrinsic features from the time series, selects a suitable set of features, and handles each feature separately. The choice and combinations of different methods and techniques contains the best-performing methods during preliminary experiments.

The workflow of Telescope is briefly illustrated in Algorithm 1 and gets as input a univariate time series ts and the horizon h . The horizon specifies how many values have to be forecast at once. In the first phase (Line 1), the time series is preprocessed and the frequencies of the underlying patterns are extracted. Telescope is intended to handle seasonal time series as many time series are observed or produced by systems subjected to human habits and are thus seasonal. In other words, if a seasonal time series has to be forecast, the second and third phases of Telescope comprise the extracting of relevant intrinsic time series features (Line 3) and building a model that describes the time series based on these features (Line 4). Afterward, the model is used to forecast the behavior of the future time series (Line 5). In the

Algorithm 1: Telescope forecasting workflow.

```

Input: Time series  $ts$ , horizon  $h$ 
Result: Forecast of  $ts$ 
1  $[ts, freqs] = \text{Preprocessing}(ts)$ ; // see Section 4.1
2 if  $freqs[1] > 1$  then //  $ts$  is seasonal
3    $features = \text{FeatureExtraction}(ts)$ ; // see Section 4.2
4    $model = \text{ModelBuilding}(ts, features)$ ; // see Section 4.3
5    $forecast = \text{Forecasting}(model, h)$ ; // see Section 4.4
6 else
7    $forecast = \text{ARIMA}(ts, h)$ ; // see Section 4.6
8 end
9  $forecast = \text{Postprocessing}(forecast)$ ; // see Section 4.5
10 return  $forecast$ 

```

case where no seasonality exists within a time series (Line 7), the time series is modeled and forecast with ARIMA [1]. Finally, the forecast is postprocessed according to the preprocessing phase and returned. In the following, each phase is explained in detail. Afterwards, the limitations and assumptions are discussed in Section 4.8.

4.1 Preprocessing

The first phase of Telescope is called *Preprocessing* and the workflow is depicted in Figure 2. Orange, rounded boxes represent actions, green hexagons the input of a phase, and blue trapezoids the output of a phase. This phase gets as input the *Time Series*, which is prepared for the following phases. As forecasting methods, especially machine learning methods, struggle with changing variance and multiplicative effects within a time series [36], the time series is transformed. More precisely, Telescope applies the *Box-Cox Transformation* (see Section 2.4) to the *Time Series*. We integrated this transformation step as it reduces both variance and multiplicative effects of the time series, leading to an improved forecast model [35, 1]. For estimating the *Transformation Parameter* of the Box-Cox transformation, we apply the method proposed by Guerrero [37] and restrict the parameter to values greater than or equal to zero. As the Box-Cox transformation can result in a logarithmic transformation, the time series has to be “real” positive ($\forall t : y_t > 0$). Consequently, the time series is shifted along the ordinate before the transformation if there is at least one value less than or equal to zero.

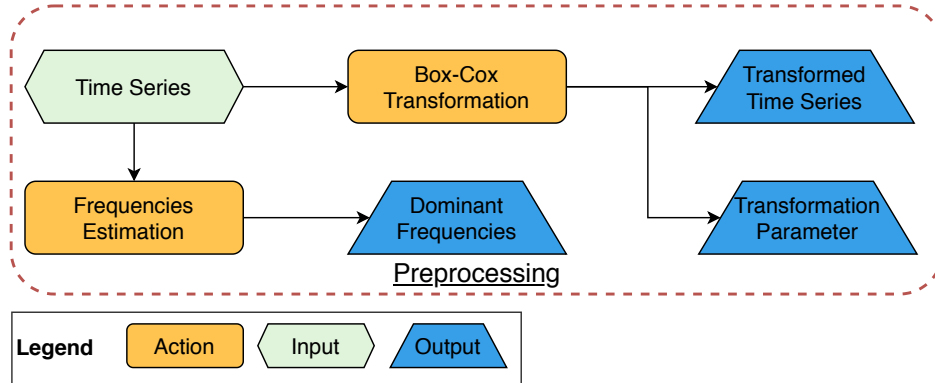


Figure 2: Preprocessing phase of Telescope.

In parallel to the transformation, Telescope performs the *Frequencies Estimation*. In short, the main idea of this step is to retrieve the most dominant⁴ frequencies from the input time series by applying a periodogram (see Section 2.2).

4.2 Feature Extraction

The second phase, *Feature Extraction*, is depicted in Figure 3. As input, this phase gets the *Transformed Time Series* and the *Dominant Frequencies* from the *Preprocessing* phase. Based on the input, Telescope retrieves intrinsic time series features for tackling typical problems or difficulties that may occur during the modeling of a time series. The first

⁴By dominant, we mean the most common period such as days in a year.

difficulty is that time series may have multiple underlying seasonal patterns [1]. To this end, Telescope determines for each dominant frequency the associated *Fourier Terms* (see Section 2.1.2) of the *Transformed Time Series* for modelling the different patterns. More precisely, for each dominant frequency, a sine as well as a cosine with the period length of the corresponding frequency are retrieved from the time series.

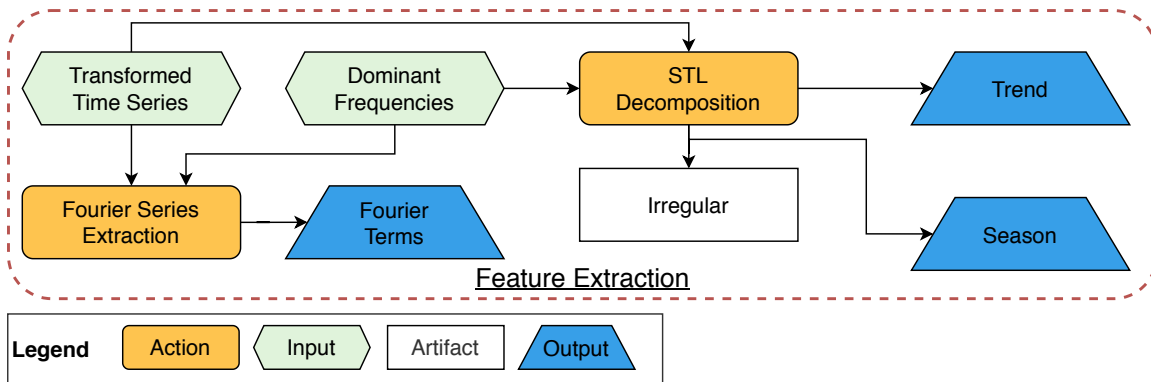


Figure 3: Feature extraction phase of Telescope.

Another problem are time series violating the stationary property (see Section 2.1.3). Although most forecasting methods assume stationary time series [5], many time series exhibit trend and/or seasonal and thus are non-stationary [6]. To handle the non-stationarity, the core idea of Telescope is to decompose the time series and then deal with each part separately. To this end, the *Transformed Time Series* is split into its components *Trend*, *Season*, and *Irregular* with STL (see Section 2.3). For the decomposition task, the most dominant frequency is used to specify the length of the seasonal pattern and the extraction of the pattern is set to periodic, that is, we assume that the seasonal pattern does not evolve over time. Although STL can only deal with an additive relationship between the components of a time series, it is not examined whether the time series follows an additive or multiplicative decomposition. More precisely, we assume that the Box-Cox transformation in the *Preprocessing* phase has minimized or removed the multiplicative effects.

4.3 Model Building

In the third phase called *Model Building*, the model that reflects the time series is build on the inputs *Transformed Time Series*, *Trend*, *Fourier Terms*, and *Season*. To build a suitable model describing the time series, we apply machine learning for finding the relationship between the time series and the intrinsic features. In a time-critical scenario, that is, the forecast is required with a reliable time-to-result, Telescope implements XGBoost [38] as regression-based machine learning method. We choose XGBoost since boosting tree algorithms are time-efficient, accurate, and easy to interpret [39]. In a scenario where the time-to-result is negligible, Telescope uses its recommendation system for selecting the most appropriate regression-based machine learning method for the given time series.

4.3.1 Time-Critical Scenario

Figure 4 shows the *Model Building* phase in a time-critical scenario. As a strong trend both increases the variance and violates stationarity, the trend introduces challenges for the model building. To this end, the first step is the removal of the trend from the time series. The resulting *De-trended Time Series* is now trend-stationary. Although seasonality can also violate stationarity, machine learning methods are suitable for pattern recognition [40]. Consequently, XGBoost learns during its training procedure how the *De-trended Time Series* can be described by the intrinsic features *Fourier Terms* and *Season*. Note that the irregular part of the time series is not explicitly considered a feature to reduce the model error and later the forecast error. That is, the machine learning method learns the irregular part as the difference that is missing to fully recreate the de-trended time series.

4.3.2 Non-Time-Critical Scenario

The *Model Building* phase in a non-time-critical scenario, which is depicted in Figure 5, is identical to the phase in a time-critical scenario, with the exception that no fixed machine learning method is used. In other words, the machine learning method is selected based on the time series. To this end, the *De-trended Time Series* is passed to the recommendation system that extracts characteristics of the time series. Based on these characteristics, the most

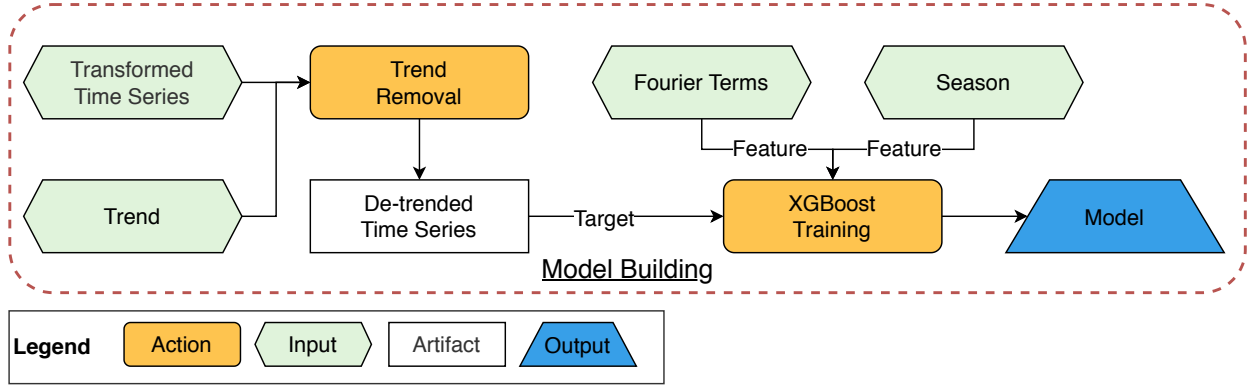


Figure 4: Model building phase of Telescope in a time-critical scenario.

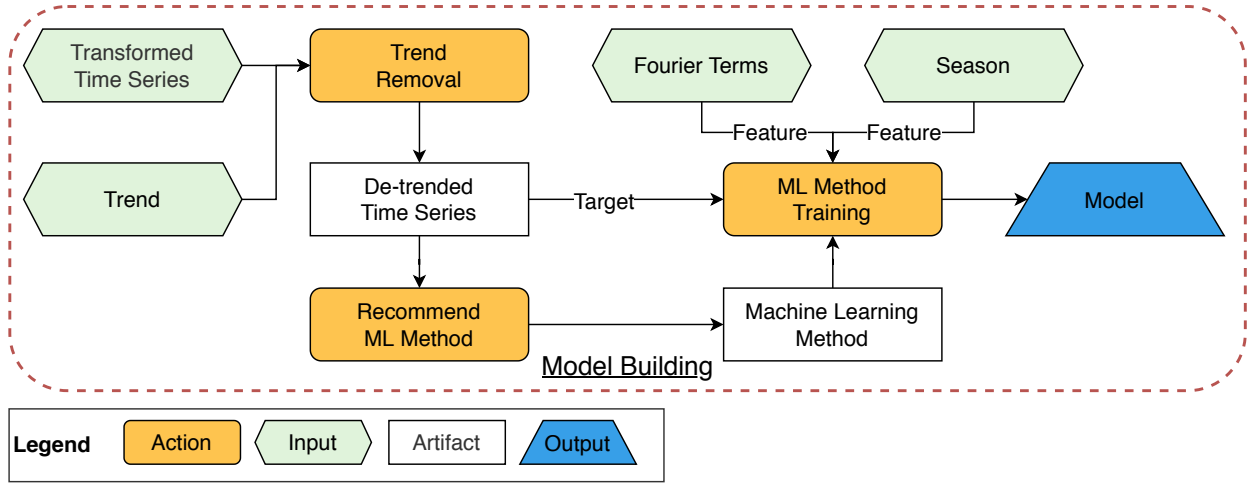


Figure 5: Model building phase of Telescope in a non-time-critical scenario.

suitable machine learning method is selected and used to learn how the intrinsic features can describe the time series. The details of the recommendation system are described in Section 4.7.

4.4 Forecasting

In the *Forecasting* phase, which is illustrated in Figure 6, the different components are forecast separately and then, the future time series is assembled. To this end, this phase gets as input the *Trend*, *Fourier Terms*, *Season* as well as the *Model*. As the *Season* and the *Fourier Terms* are recurring patterns per definition, they can be merely continued. More precisely, the seasonal pattern and each Fourier terms is forecast for the time $n + k$ as follows

$$\hat{y}_{n+k|n} := y_{n+k-m \cdot (\lfloor \frac{k-1}{m} \rfloor + 1)}, \quad (5)$$

where y_t is the observations at time t , n the number of historical observations, m the length of the associated period, and the forecast horizon k being a positive integer. The resulting *Future Season* and *Future Fourier Terms* are used as features in conjunction with the *Model* for predicting the future de-detrended time series. More precisely, the machine learning method regresses a new value of the *Future De-trended TS* for each point in time of the forecast based on the corresponding values of the features.

In parallel to the forecast of the recurring patterns, the trend is also forecast. Since the *Trend* contains no recurring patterns, an advanced forecasting method is required to forecast the *Future Trend*. To this end, we apply ARIMA as it is able to estimate the trend even from a few points. More precisely, we apply `auto.arima`⁵ [41] that automatically finds

⁵Auto.arima conducts a search over possible sARIMA and ARIMA models.

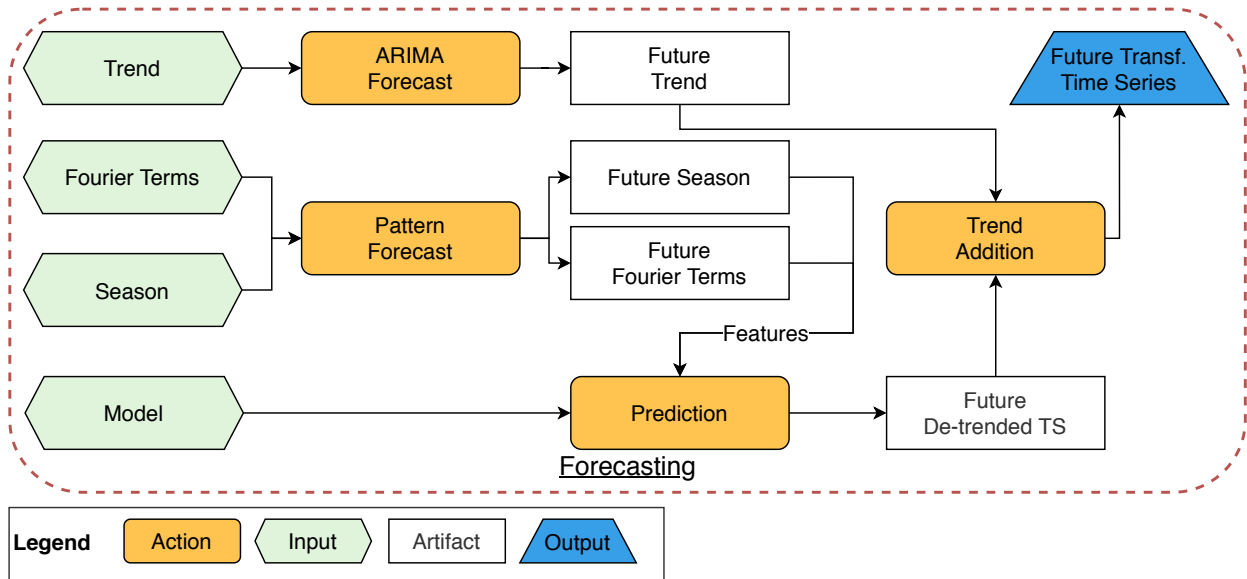


Figure 6: Forecasting phase of Telescope.

the most suitable model parameters for a time series. After the trend is forecast, the last step of this phase is assembling the forecast of the time series. To this end, the *Future De-trended TS* and the *Future Trend* are summed up.

4.5 Postprocessing

The last phase of Telescope is called *Postprocessing* and its workflow is depicted in Figure 7. As the name suggests, this phase is the counterpart of the *Preprocessing* step. More precisely, it gets as input the *Future Transf. Time Series* from the *Forecasting* phase and the *Transformation Parameter* from the *Preprocessing* phase. As the time series was adjusted with the Box-Cox transformation, the *Future Transformed Time Series* has to be re-transformed with the identical transformation parameter from the *Preprocessing* phase. To this end, the inverse Box-Cox transformation with the *Transformation Parameter* is applied to the *Future Transformed Time Series*. If the time series was shifted along the vertical axis in the first phase, the time series also has to be moved back. Finally, the forecast of the original time series is returned.

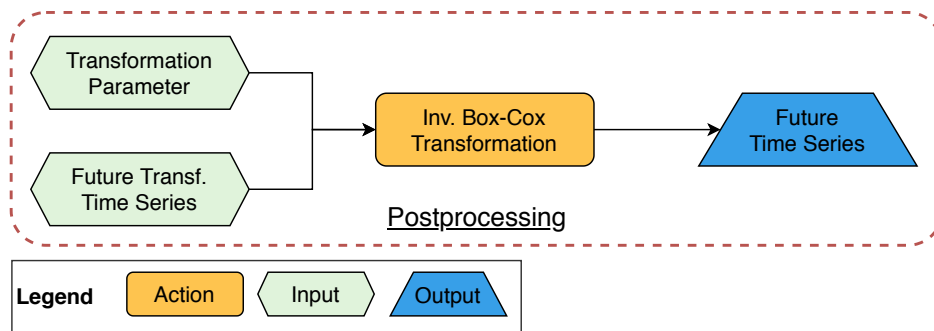


Figure 7: Postprocessing phase of Telescope.

4.6 Fallback for Non-Seasonal Time Series

Telescope’s core idea is to detect recurring patterns within a time series and use this information to retrieve intrinsic features. However, if Telescope has to forecast a non-seasonal time series, the normal workflow cannot be used as the time series lacks recurring patterns. Moreover, the integrated STL also requires a seasonal pattern to decompose the time series. Consequently, Telescope requires a fallback for non-seasonal time series. That is, if the *Preprocessing*

phase returns only the frequency of 1 as dominant frequency, the *Feature Extraction*, *Model Building*, and *Forecasting* phase are omitted and the fallback is executed instead. The fallback receives as input the transformed time series from the *Preprocessing* phase on which a non-seasonal ARIMA model is trained. Then, the forecast is performed and the future transformed time series is forwarded to the *Postprocessing* phase, in which the time series is treated as described in Section 4.5.

4.7 Recommendation System for Machine Learning Method

According to the “No-Free-Lunch Theorem”, which – simply spoken – states that each method has its advantages and weaknesses depending on the specific use case, it is inadvisable to rely only on one particular method. Typically, the choice of the optimal forecasting method is based on expert knowledge, which can be expensive, may have a subjective bias, and may take a long time to deliver results. Consequently, we automate the selection of the best regression-based machine learning method for a given time series. More specifically, we employ a recommendation system based on *meta-learning* to select the most appropriate method based on time series characteristics. In the meta-learning context, the methods, which are available for selection, are referred to as *base-level methods* and characteristics on which the selection is based are called *meta-level attributes*.

4.7.1 Problem Formalization

Following the idea of the *algorithm selection problem* [42] and its formal description [43], the *regression-based machine learning method selection problem* that arises for Telescope can be formally defined as: For a given time series $y \in Y$, with characteristics $f(y) \in F$, find the selection mapping $S(f(y))$ into the algorithm space A , such that the selected algorithm $a \in A$ minimizes the forecast error measure $m(a(y)) \in M$. In this formulation, the problem space Y represents a set of time series; the feature space F contains measurable time series characteristics of each instance of Y , calculated by a deterministic extraction procedure; the algorithm space A is the set of all considered regression-based machine learning methods; the performance space M represents the mapping of each algorithm to the forecast error measure.

4.7.2 Meta-Level Attributes

To have an accurate recommendation system for choosing the most appropriate regression-based machine learning method for a given time series, a sound set of characteristics, which describe the time series, is required. To this end, the considered time series characteristics originate from different sources: statistical measures of a time series (S1–S6) as well as characteristics proposed in a former paper [44] (B1–B8), proposed by Lemke and Gabrys [45] (L1–L4), and proposed by Wang et al. [19] (W1–W6). In contrast to the work of Wang et al. [19], we use the raw values of the characteristics to avoid arbitrary normalization factors. The time series characteristics applied to the meta-learning approach are listed in Table 1. Note that these characteristics are only a subset of the original set, selected by, for example, correlation analysis.

4.7.3 Base-Level Methods

We only consider machine learning methods to learn how the de-trended time series can be described with intrinsic time series features. Telescope implements seven regression-based machine learning methods⁶ as the base-level methods: (i) *CART* [47] is a regression tree that recursively partitions the data set. To prevent the tree from becoming too large, the tree is automatically pruned. (ii) *Cubist* is a rule-based regression method that builds upon the *M5* model tree [48]. The rules are arranged hierarchically, resulting in a tree where each leaf node represents a multivariate linear regression model. (iii) *Evtree* [49] implements an evolutionary algorithm for constructing a regression tree that splits the data so that each partition decision is globally optimal. (iv) *NNetar* [50] is a feed-forward neural network consisting of one hidden layer and is trained with lagged values of the time series. The number of nodes in the hidden layer and the lags are automatically determined. (v) *Random forest* [51] is an ensemble method comprising multiple regression trees. Each tree is built independently of the others, and for each split, only a random subset of the features is considered. (vi) *SVR* [52] uses the same principles as SVM. More precisely, a set of hyperplanes is constructed for separating the data. If the data is not linear in its input space, the values are mapped into a higher dimensional feature space. (vii) *XGBoost* is an ensemble method consisting of multiple regression trees and uses gradient tree boosting, i.e., each tree grows with knowledge from the last trained tree.

⁶The selection contains the seven best-performing methods during preliminary experiments.

4.7.4 Rule Generation Approach

To recommend the most appropriate regression-based machine learning method for a given time series, we propose a regression-based rule generation approach, as illustrated in Figure 8. The idea is to apply a random forest⁷ for learning how meta-level attributes (i.e., time series characteristics) can be mapped to the performance of the base-level methods (i.e., regression-based machine learning methods).

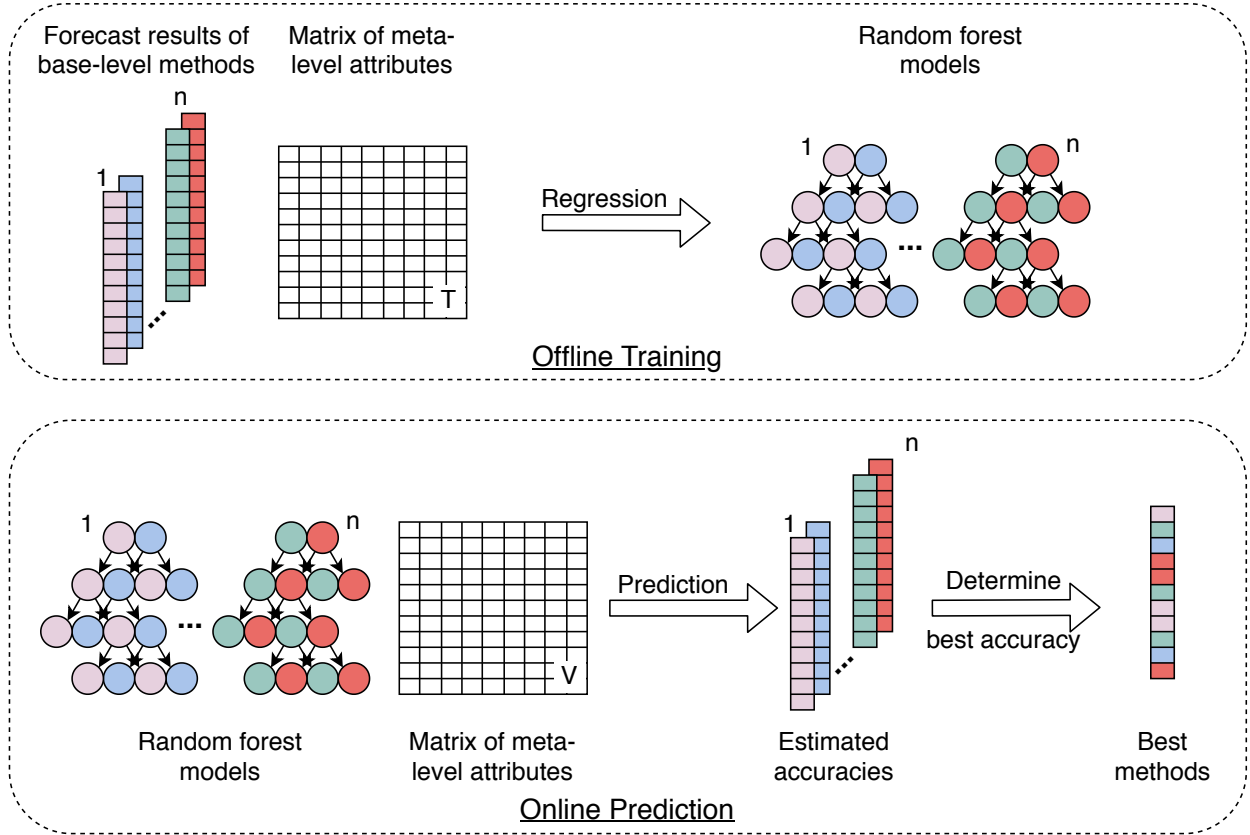


Figure 8: Schematic process of the rule generation.

In detail, for each base-level method, a random forest is trained. Each model estimates the forecast accuracy for a given time series. Then, the method that has the best accuracy based on the estimations is selected. More precisely, the approach calculates for each base-level method and each time series how worse it is compared to the method with the best forecast accuracy. Then, the method that has the best accuracy based on the estimations is selected. More precisely, the approach calculates for each base-level method and each time series how worse it is compared to the method with the best forecast accuracy. We reflect this deterioration with the *forecast accuracy degradation* that can be calculated as

$$\vartheta_i := \frac{\epsilon_i}{\min(\epsilon_1, \dots, \epsilon_n)}, \quad (6)$$

where ϵ_i is the forecast accuracy of the i -th method and n is the number of considered methods. The values of ϑ_i lie in the interval $[1, \infty)$, where 1 indicates that this method has the best forecast accuracy. After the calculation of the degradation, a random forest is used as regressor for each base-level method, where the meta-level attributes are the features and the forecast accuracy degradation vector of the respective method is the target. In other words, the regression task leads to n random forest models, each reflecting the estimate of the forecast accuracy degradation compared to the best method for a given time series. After the training, the meta-level attributes of new time series can be fed to each of the random forest models. Based on these attributes, each model estimates the forecast accuracy degradation vector. Then, the base-level methods with the lowest estimated forecast accuracy degradation are returned for each time series.

⁷The choice of random forest for the recommendation is based on the extensive experiments in which these combinations yielded the best results.

4.7.5 Offline Training

In the *Offline Training* phase, which is depicted in Figure 10, the rules for recommending a specific method based on time series characteristics are learned or updated either when Telescope is started or when no forecast is currently being conducted. To this end, this phase gets a *Set of Time Series* as input. To retrieve precise rules for the recommendation of the most appropriate regression-based machine learning method for a given time series, a set of time series which may be similar to this time series is required. Therefore, Telescope generates n new time series based on the initial set of time series in this phase. The main idea is to decompose all time series in the set into components (trend, season, and irregular part) and then, build randomly new time series based on these components. An example output of this time series generation is illustrated in Figure 9.

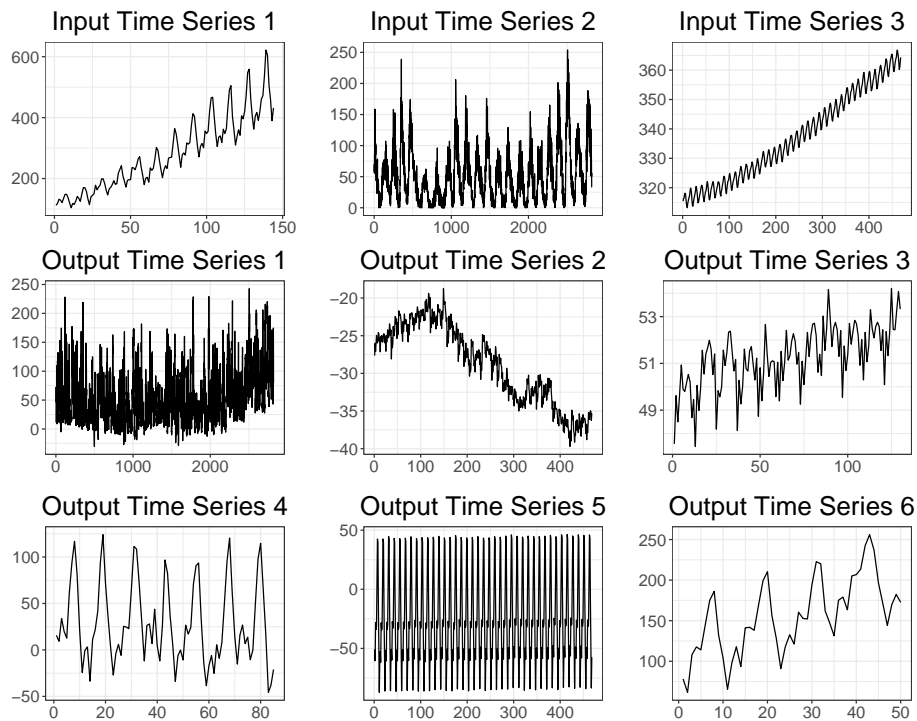


Figure 9: Example of six generated time series.

As the machine learning method has to learn how the de-trended time series can be described by the intrinsic features, each time series in the *Extended Set of Time Series*, which comprises the initial time series and the newly generate time series, has to be de-trended. For this purpose, each time series is transformed with the Box-Cox transformation, de-trended, and the intrinsic features are extracted.

In the *Base-Level Methods Evaluation* step, each base-level method is trained and evaluated on every de-trended time series and its associated intrinsic features (Fourier terms and seasonal pattern). To this end, the time series is split into history (the first 80% of the time series) and in future/test (the remaining 20%). In parallel, the time series characteristics (i.e., meta-level attributes) of each de-trended time series are extracted. The *Forecasting results* from the *Base-Level Methods Evaluation*, in this case the forecast error based on sMAPE (see Section 5.1.3), and the *Time Series Characteristics* are used to form the *Meta-Level Data Set*. This data set is used in the *Rule Generation* step to retrieve the rules for the recommendation of the best suited method.

4.8 Assumptions and Limitations

In the development of Telescope, we limit ourselves to univariate time series. In fact, correlated/external data can be used for each time series to improve forecast accuracy. However, the selection and preprocessing of such additional information require domain knowledge. In other words, this knowledge about domain-specific feature engineering cannot yet be fully automated. Consequently, our method would have to be tailored to a specific domain and, therefore, contradict the goal of a generic forecasting approach. Besides this limitation, Telescope has the following assumptions that are either based on the integrated tools or design decisions: (i) The time series must not contain missing values, and

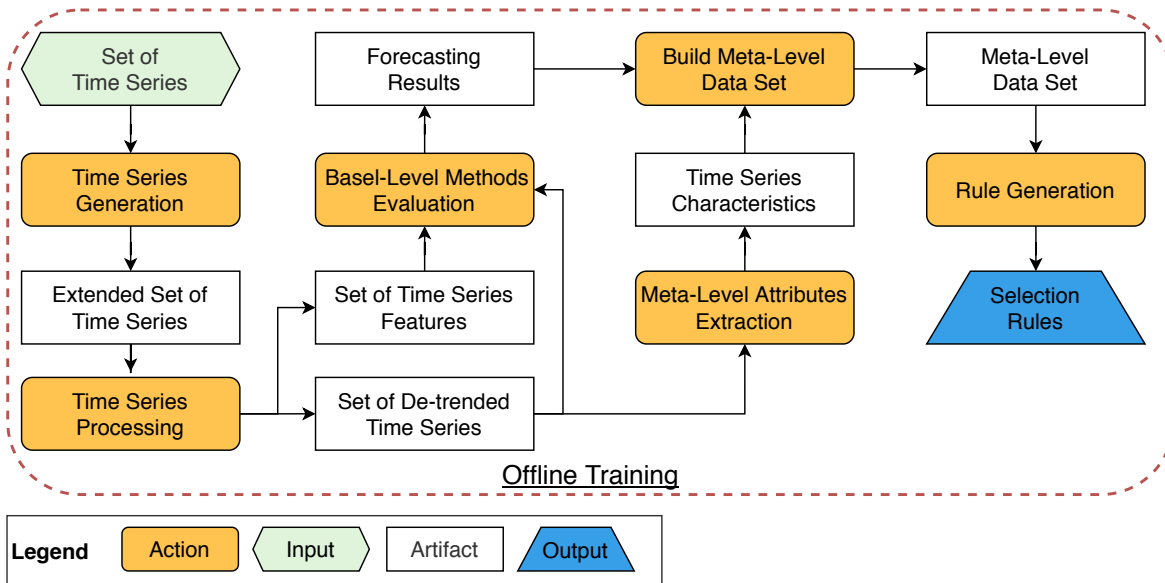


Figure 10: Offline training phase of Telescope.

each value must be numeric. Indeed, missing values can be interpolated. However, if the gaps are large or frequent, the values must be treated with caution, as they can affect the model’s accuracy. (ii) For the recommendation of the most appropriate regression-based machine learning method, Telescope assumes that the time series to be forecast and the time series that are used for the training (i.e., the initial time series set plus the generated time series) originate from the same population and therefore, have a similar distribution of time series characteristics. In other words, Telescope assumes the rules that are retrieved based on the training set can also be applied to new time series. (iii) Usually, many systems are driven by human interactions. In other words, the time series produced or observed by these systems are subject to human habits (e.g., day/night phases) and, therefore, seasonal. Consequently, Telescope assumes that the found frequencies within time series are multiples of natural frequencies. (iv) We expect that the seasonal pattern does not evolve over time.

5 Forecasting Method Competition

To compare Telescope with recent hybrid forecasting methods, we compare Telescope to seven competing methods using a forecasting benchmark. We start with the description of the experiments. Then, we assess the performance of Telescope and the competing methods in Section 5.2. A more detailed investigation of the forecasting methods takes place in Section 5.3. Lastly, we summarize the results and discuss threats to validity in Section 5.4.

5.1 Experimental Description

In this evaluation, we investigate both the pure Telescope approach and Telescope using the recommendation system. For the training of the recommendation system, we assembled a broad data set of 150 real-world time series with varying length and frequencies. Note that the applied benchmark and this training data set consist of different time series. Moreover, Telescope augments the training data set consisting of 150 time series to 10,000. In the following, we refer to Telescope using the recommendation system as *Telescope**.

5.1.1 Applied Forecasting Benchmark

For benchmarking the forecasting methods, we apply *Libra* [53]. *Libra* is a benchmark for time series forecasting methods comprising four different use cases, each covering 100 heterogeneous time series taken from different domains. In the following evaluation, we benchmark all methods with *Libra* across all use cases (i.e., 400 time series). Moreover, the evaluation type of *Libra* was set to multi-step-ahead forecasts. That is, each time series was divided into history (first 80% of the time series) and test (remaining 20% of the time series). Based on the history, each method learned a model that was used for forecasting future values of the time series (i.e., the test part) at once with a single execution. This

forecasting procedure (i.e., receiving the time series, estimating the parameters, building the model, and forecasting the time series) was repeated ten times for each time series. Consequently, the reported measures were determined on the average values of each time series.

5.1.2 Competing Methods

For having a broad and representative forecasting method competition, we compare different methods from different fields. The competitors can be grouped into three categories: (i) “classical” time series forecasting methods, (ii) regression-based machine learning methods, and (iii) hybrid forecasting methods (i.e., taking advantage of at least two methods). For the first and second groups, we consider their best performing representative on the benchmark (i.e., sARIMA and XGBoost) [53]. As representatives for the hybrid methods, we investigate recent existing approaches. The considered seven forecasting methods are listed and briefly described below: (i) *BETS* decomposes the time series into the components trend, season, and irregular. Different versions of the irregular part are then simulated, resulting in different versions of the original time series. Then, these versions are forecast separately by ETS [54], and the final forecast is the median of all forecasts. (ii) *ES-RNN* is a hybrid forecasting method based on time series decomposition developed by Uber. The basic idea is to de-seasonalize the time series using exponential smoothing and to use a neural network for extrapolation of the time series. (iii) *FFORMS* is based on forecasting method recommendation. More precisely, a random forest is applied as a classifier to map the most appropriate forecasting method (ETS, NNetar, sARIMA, sNaïve [50], TBATS [55], and Theta [56]) to a specific time series described by a set of time series characteristics. (iv) *Hybrid*⁸ performs an ensemble forecast. The considered methods comprise ETS, NNetar, Theta, sARIMA, and TBATS. For the ensemble forecast, each method performs a forecast, and the final forecast is the average of these forecasts. (v) *Prophet* is a hybrid forecasting method based on decomposition developed by Facebook. The time series is decomposed into the components trend, season, holiday, and error. Each component is forecast by a different approach. Then, the forecast parts are assembled to form the final forecast. (vi) *sARIMA*⁹ extends the ARIMA model [57] by adding a seasonal counterpart to each component (autoregressive model for the past values, moving average for the past forecast errors, and time series differencing for stationarity). (vii) *XGBoost* is an ensemble of decision trees based on gradient tree boosting, that is, the trees are growing sequentially with knowledge from their preceding tree. To reduce overfitting, XGBoost applies regularization objects, shrinkage, and feature subsampling.

As input for the forecasting task, sARIMA, BETS, NNetar, and Hybrid received the time series and the respective frequency. Prophet also needs the timestamps of the time series. XGBoost received the time series and a synthetic seasonal pattern (a vector with the indices modulus the frequency) as input. FFORMS was trained on the M4- and M3-Competition [58] and received as input also the time series and the respective frequency. In contrast to the other methods, ES-RNN requires, besides the time series and frequency, a set of time series. According to the workflow of Libra, ES-RNN got the same time series several times. Note that we used all methods “out-of-the-box” since the results of the M3-Competition have shown that the methods were kept simple and that complex models do not necessarily perform better [58]. That is, there was no parameter tuning, and the methods were used with their default settings. Recall the “No-Free-Lunch Theorem”, stating that improving a method for one aspect leads to deterioration in performance for another aspect. Moreover, also the techniques deployed in Telescope were used with their default settings.

5.1.3 Considered Measures

To compare and quantify the performance of the different forecasting methods, we report the symmetrical mean absolute percentage error (sMAPE) and a normalized time-to-result¹⁰. The sMAPE is defined as following where n is the forecast length, y_t the actual value, and f_t the forecast value:

$$\text{sMAPE} := \frac{200\%}{n} \sum_{t=1}^n \left| \frac{y_t - f_t}{y_t + f_t} \right|. \quad (7)$$

In the following, the measures \bar{e} , \tilde{e} , and σ_e reflect the average error, median error, and the standard deviation of the sMAPE, while the measures \bar{t}_N , \tilde{t}_N , and σ_{t_N} reflect the average time, median time, and the standard deviation of the time-to-result normalized by the time required by a naïve forecast¹¹.

⁸Hybrid forecasting method: <https://cran.r-project.org/web/packages/forecastHybrid/vignettes/forecastHybrid.html>

⁹In the experiments, we use `auto.arima` [1] to find the most suitable model for the time series automatically.

¹⁰The time-to-result for a time series reflects the duration in which the forecasting method receives the time series, estimates the parameters, creates the model, and performs the forecast.

¹¹The naïve forecast needed on average 0.01 seconds per forecast.

5.2 Benchmarking the Telescope Approach

In this section, we compare Telescope against the competing forecasting methods. As an example, Figure 11 shows the different forecasting methods on the AirPassengers [59] time series. This time series, which is often used as a baseline, shows that all forecasting methods are correctly configured and perform reasonable forecasts. Therefore, we investigate the results (i.e., forecast error and time-to-result) of these methods over all time series in the following.

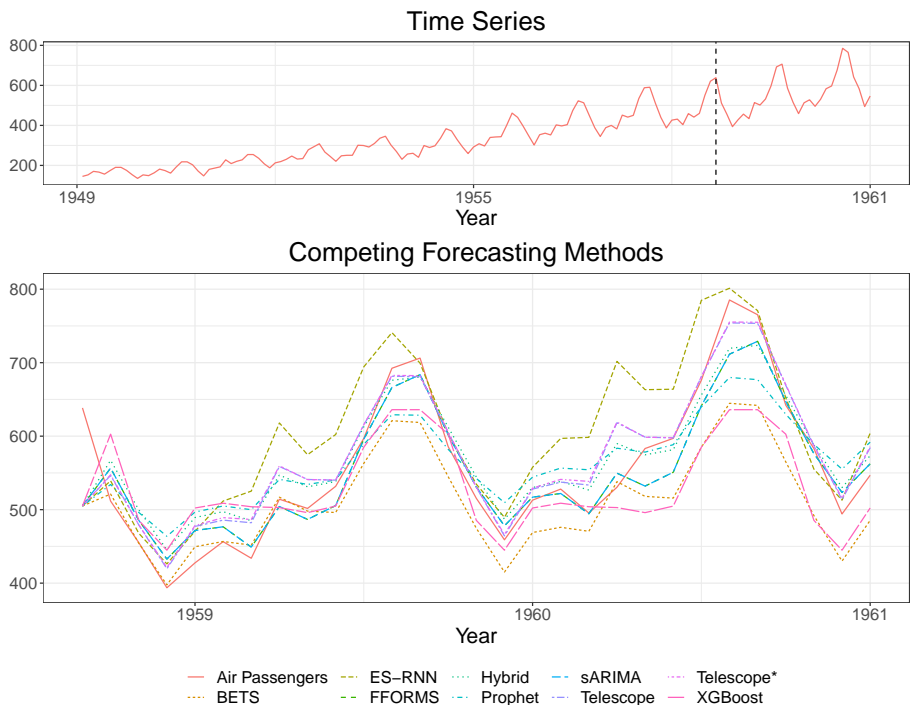


Figure 11: Forecasts for all methods in competition on the airline passengers time series.

Table 2 shows the performance for all competing methods in competition averaged over all time series of all use cases. Each row represents a measure, each column a method, and the best values (the lower, the better) are highlighted in bold. The most accurate forecasting method is Telescope* (19.46%) followed by Telescope (19.95%) and sARIMA (20.63%). By far, the fastest method is XGBoost (6.73). Telescope ($1.43 \cdot 10^2$) has the second-lowest time-to-result while Telescope* is, on average, ten times slower. Although sARIMA has the third-lowest forecast accuracy, it is, on average, almost 6000 times slower than Telescope and almost 300 times slower than Telescope*. More precisely, the maximum actual forecast time of sARIMA for a time series was 465,574 seconds, which corresponds to almost 5.5 days.

We also investigate the variation of the forecast error and the time-to-result as a crucial property. The lowest standard deviation regarding \bar{e} is shown by Telescope* (27.96%) followed by Telescope (31.35%) and BETS (34.25%). In terms of the time-to-result, XGBoost has the lowest variation (15.59) followed by Telescope (98.67).

Although the mean and standard deviation are useful statistical measures, we also examine the distributions of the forecast error and the time-to-result. Figure 12 illustrates the forecast error distribution of all methods. Each distribution is depicted as a box plot, where the horizontal axis shows the different methods and the vertical axis the forecast error in log-scale. The methods FFORMS, sARIMA, Telescope, and Telescope* are showing almost the same distribution (i.e., quite short and similar interquartile ranges) with only a few outliers. Although Hybrid shows a similar distribution between the 25th and 75th quantiles, there are many outliers above the upper whisker. Another group with similar distribution comprises the methods BETS, Prophet, and XGBoost. The method with no outliers above the upper whisker but with the longest interquartile range is ES-RNN. However, the fairly similar error distributions are consistent with the “No-Free-Lunch Theorem”, which states that there is no forecasting method that works best for all scenarios. Figure 13 depicts the time-to-result distribution of all methods. Again, each distribution is illustrated as a box plot, and the vertical axis shows the time-to-result in log-scale. In contrast to the error distribution, the time-to-result distributions are completely different. XGBoost or Telescope exhibit a low variation in the time-to-result. In contrast, FFORMS and sARIMA have a wide range of the time-to-result. Consequently, both methods may be impractical for

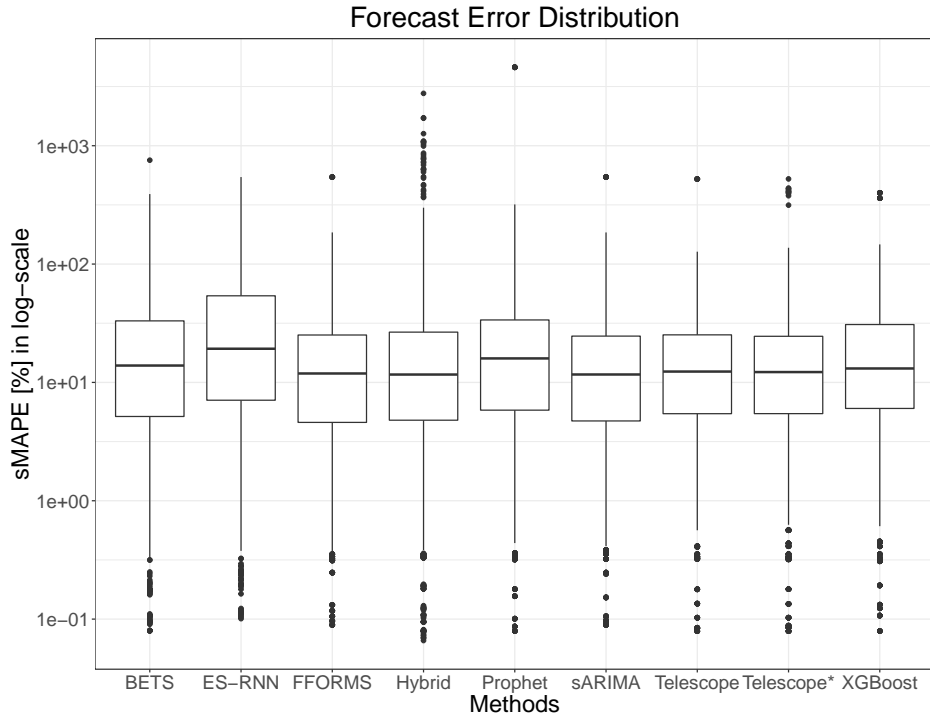


Figure 12: Forecast error distribution.

time-critical scenarios. Telescope* has, in comparison to Telescope, also a huge variation regarding the time-to-result. Thus, Telescope is used in time-critical scenarios and Telescope* in non-time-critical scenarios.

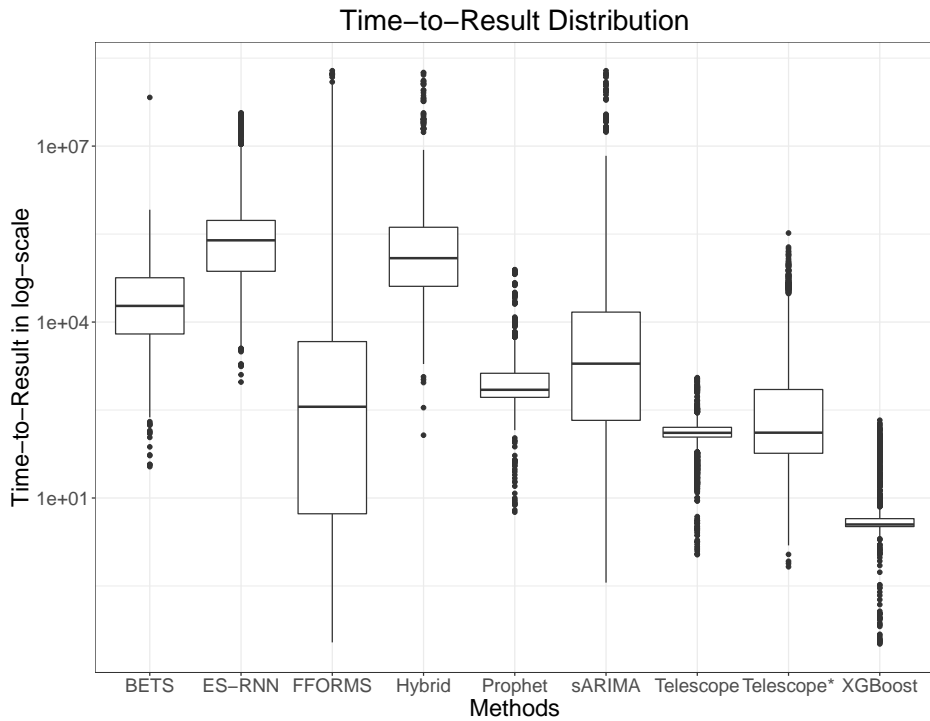


Figure 13: Time-to-result distribution.

5.3 Detailed Examination

Although our contribution Telescope can handle both seasonal and non-seasonal time series, it is intended for long and seasonal time series. To this end, we analyze the forecasting performance of all methods on seasonal (85% of the data set) and non-seasonal (15%) time series. Table 3 lists the results for seasonal and non-seasonal time series. Each row shows a measure and each column a method. The best values (the lower, the better) are highlighted in bold. The lowest forecast error on average for seasonal time series is exhibited by Telescope* (19.77%) followed by Telescope (20.33%) and sARIMA (21.10%). However, Telescope is about 6300 and Telescope* is about 260 times faster than sARIMA. In the case of non-seasonal time series, FFORMS (16.47%) has the lowest forecast error followed by Telescope* (17.65%) and Telescope (17.65%). Note that both Telescope and Telescope* are using the same fallback for non-seasonal time series. Since the fallback, which comprises ARIMA, has a lower error than the sARIMA (17.85%), we are able to see the impact of the Preprocessing and Postprocessing phase of Telescope. In both cases, XGBoost exhibits the lowest time-to-result followed by Telescope.

To investigate the trade-off between forecast error and time-to-result, we compare the methods in a 2-dimensional space spanned by the forecast error and time-to-result. We compute the median time-to-result \tilde{t}_N and median forecast error \tilde{e} over all methods. Based on these both values, we can sort the forecasts of each method for each time series (t_N, e) in one of the four quadrants: (i) $[\tilde{t}_N; \infty[\times [\tilde{e}; \infty[$, (ii) $[0; \tilde{t}_N[\times [\tilde{e}; \infty[$, (iii) $[0; \tilde{t}_N[\times [0; \tilde{e}[$, or (iv) $[\tilde{t}_N; \infty[\times [0; \tilde{e}[$. The best trade-off is achieved in the 3rd quadrant as both the forecast error and time-to-result are lower than the median values. A semi-good performance is reflected by the 2nd and 4th quadrant as either the time-to-result or the forecast error is lower than the associated median value. The worst performance is achieved by forecasts in the 1st quadrant. Here, both the forecast error and time-to-result are worse than the median values. Figure 14 depicts each forecast as a point in the 2-dimensional space. The vertical axis represents the forecast error and the horizontal axis the time-to-result. Both axes are in log-scale. The vertical dashed line represents \tilde{t}_N and the horizontal axis \tilde{e} . Each forecasting method is depicted in an individual color and point shape. The methods XGBoost, Telescope, and Telescope* have compact clusters in the 2nd and 3rd quadrant. In contrast, the methods BETS, ES-RNN, and Hybrid have far-reaching clusters in the 1st and 4rd quadrant. However, almost all methods have time series in each quadrant.

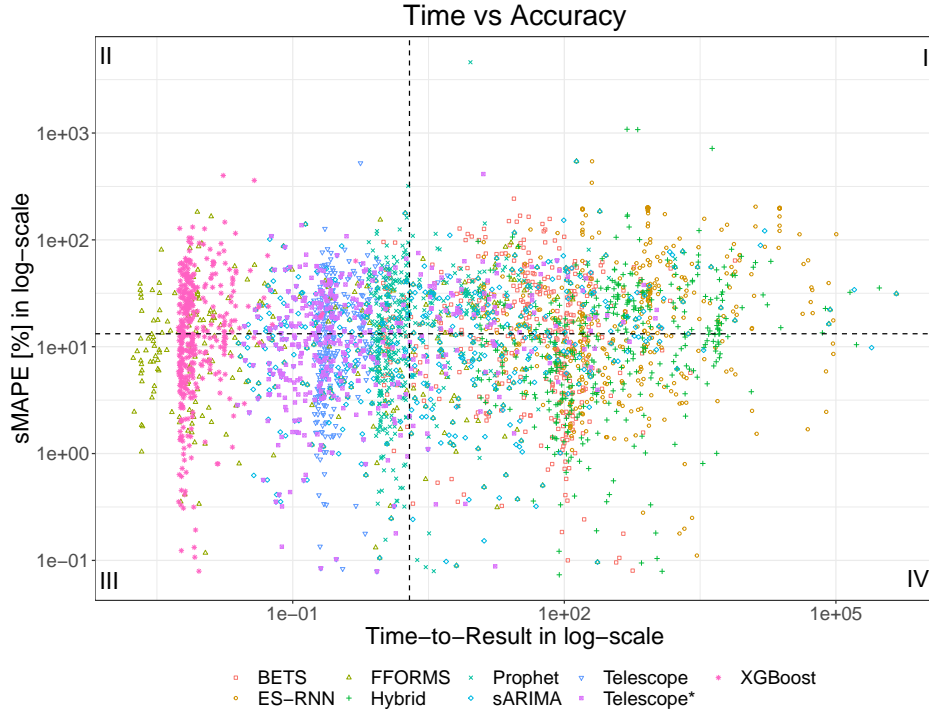


Figure 14: Forecast error vs. time-to-result for all methods.

Table 4 shows the distribution of the time series for each forecasting method in each quadrant. Each row represents a quadrant and each column a method. For instance, XGBoost has all forecasts equally distributed in the 2nd and 3rd quadrant. That is, all forecasts have a lower time-to-result than \tilde{t}_N . In contrast, ES-RNN and Hybrid have all of their forecasts in the 1st and 4rd quadrant. Consequently, all forecasts having a longer time-to-result than \tilde{t}_N . Telescope has

98% of its forecast in the 2nd and 3rd quadrant. More precisely, 51% of these forecasts are located in the 3rd quadrant. In other words, Telescope has the highest number of forecasts exhibiting the best trade-off. The second most time series in the 3rd quadrant has XGBoost (50%) followed by Telescope* (42%).

5.4 Summary of the Results and Threats to Validity

Our experiments showed that both Telescope and Telescope* outperforms the state-of-the-art. More precisely, Telescope* achieves the best forecast accuracy followed by Telescope. Although our approach is intended for seasonal time series, it exhibits the second-best forecast accuracy on non-seasonal time series. Moreover, Telescope is, on average, up to 6000 times faster than the third most accurate method. In all experiments, Telescope* is more accurate than Telescope, but has, on average, a higher time-to-result as well as variation in the time-to-result. Also, we show that the chosen configuration of Telescope has the best trade-off between forecast accuracy and time-to-result. The third most accurate method, sARIMA, suffers from a high variation in the time-to-result. The winner of the M4-Competition, ES-RNN, is tailored for cross-learning to the M4-Competition data and therefore has the highest average forecast error. In summary, Telescope exhibits the best forecast accuracy coupled with a low and reliable time-to-result.

Although we applied a forecasting benchmark, the evaluation results may not be generalized to all time series from all areas. Besides the data set, we also try to have a sound set of recent hybrid forecasting methods based on different techniques. To this end, we also consider methods developed by Facebook and Uber. However, we use all methods with their default settings. Consequently, the observed results may differ if the forecasting methods are tuned to each time series. Moreover, our classification of the time series into long time series and time series with long periods may also affect the results. As Telescope achieves on the whole data set the best performance, the ranking inside the classes may only change. Lastly, we analyze whether the observed forecast accuracy as well as the measured time-to-result are statistically significant. To this end, we apply a non-parametric statistical test. More formally, we use the Friedman test [60], which ranks the forecasting methods separately for each time series and compare the average ranks of the methods. In case of a tie, average ranks are assigned. Thus, we formulate the following hypothesis:

$$H_{0,i} : \text{The methods perform equally}$$

for the forecast error ($i=1$) and for the time-to-result ($i=2$). We conduct both hypotheses with a significance level of 1%. The resulting p-values $p_1 < 2 \cdot 10^{-16}$ and $p_2 < 2 \cdot 10^{-16}$ indicate that both hypotheses can be rejected. Thus, the differences in the exhibited performance of the forecasting methods are statistically significant.

6 Conclusion

This paper introduces Telescope, a novel machine learning-based forecasting approach that automatically retrieves relevant information from a given time series and splits it into parts, handling each of them separately (addressing RQ1). More precisely, Telescope automatically extracts intrinsic time series features and then decomposes the time series into components, building a forecasting model for each of them (addressing RQ2). Each component is forecast by applying a different method and then the final forecast is assembled from the forecast components by employing a regression-based machine learning algorithm. For non-time-critical scenarios, we additionally provide an internal recommendation system that can be employed to automatically select the most appropriate machine learning algorithm for assembling the time series from its components (addressing RQ3).

In more than 1000 hours of experiments comparing Telescope against seven competing methods (including approaches from Uber and Facebook) using a forecasting benchmark, Telescope outperformed all methods, exhibiting the best forecast accuracy coupled with a low and reliable time-to-result. Compared to the competing methods that exhibited, on average, a forecast error (more precisely, the symmetric mean absolute error) of 29%, Telescope exhibited an error of 20% while being 2556 times faster. In particular, the methods from Uber and Facebook exhibited an error of 48% and 36%, and were 7334 and 19 times slower than Telescope, respectively. When additionally applying the recommendation system, Telescope was able to reduce the forecast error even further down to 19%.

We see potential for extending Telescope to support multivariate time series and to detect structural changes in time series. The forecasting of multivariate time series has advantages and disadvantages. On the one hand, there is additional information that can be used to refine the prediction model. On the other hand, each extra piece of information has to be predictable to form the final forecast. If structural changes occur in time series, Telescope is prone to poor forecasts as it assumes, for example, that the seasonal pattern does not change over time. Besides the change in the seasonal pattern, further structural changes include level shifts or breakpoints in the trend. To mitigate this problem, structural changes have to be detected automatically. Based on the found changes, the values before the last change can be discarded if there is no regularity in the changes to consider only the time series's current structure for forecasting the time series.

References

- [1] Rob J Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 2017.
- [2] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr 1997.
- [3] Xavier Fontes and Daniel Castro Silva. Hybrid Approaches for Time Series Prediction. In *Hybrid Intelligent Systems*, pages 146–155, Cham, 2020. Springer International Publishing.
- [4] André Bauer, Marwin Züfle, Nikolas Herbst, Samuel Kounev, and Valentin Courtef. Telescope: An automatic feature extraction and transformation approach for time series forecasting on a level-playing field. In *Proceedings of the 36th International Conference on Data Engineering (ICDE)*, April 2020.
- [5] Peter J Brockwell and Richard A Davis. *Introduction to Time Series and Forecasting*. springer, 2016.
- [6] Ratnadip Adhikari and R. K. Agrawal. An Introductory Study on Time Series Modeling and Forecasting. *CoRR*, abs/1302.6613, 2013.
- [7] Arthur Schuster. The Periodogram of Magnetic Declination as Obtained from the Records of the Greenwich Observatory during the Years 1871-1895. *Transactions of the Cambridge Philosophical Society*, 18:107–135, 1899.
- [8] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. STL: A Seasonal-Trend Decomposition Procedure based on Loess. *Journal of Official Statistics*, 6(1):3–73, 1990.
- [9] George EP Box and David R Cox. An Analysis of Transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252, 1964.
- [10] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: Results, Findings, Conclusion and Way Forward. *International Journal of Forecasting*, 34(4):802–808, 2018.
- [11] John M Bates and Clive WJ Granger. The Combination of Forecasts. *Journal of the Operational Research Society*, 20(4):451–468, 1969.
- [12] Ratnadip Adhikari, Ghanshyam Verma, and Ina Khandelwal. A Model Ranking Based Selective Ensemble Approach for Time Series Forecasting. *Procedia Computer Science*, 48:14–21, 2015.
- [13] Matthias Sommer, Anthony Stein, and Jörg Hähner. Local Ensemble Weighting in the Context of Time Series Forecasting Using XCSF. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.
- [14] Vítor Cerqueira, Luís Torgo, Fábio Pinto, and Carlos Soares. Arbitrated Ensemble for Time Series Forecasting. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 478–494. Springer, 2017.
- [15] Zheng Wang, Irena Koprinska, Alicia Troncoso, and Francisco Martínez-Álvarez. Static and Dynamic Ensembles of Neural Networks for Solar Power Forecasting. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.
- [16] Dihia Boulegane, Albert Bifet, and Giyyarpuram Madhusudan. Arbitrated Dynamic Ensemble with Abstaining for Time-Series Forecasting on Data Streams. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1040–1045. IEEE, 2019.
- [17] Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. FFORMA: Feature-Based Forecast Model Averaging. *International Journal of Forecasting*, 36(1):86–92, 2020.
- [18] Fred Collopy and J Scott Armstrong. Rule-Based Forecasting: Development and Validation of an Expert Systems Approach to Combining Time Series Extrapolations. *Management Science*, 38(10):1394–1414, 1992.
- [19] Xiaozhe Wang, Kate Smith-Miles, and Rob Hyndman. Rule Induction for Forecasting Method Selection: Meta-Learning the Characteristics of Univariate Time Series. *Neurocomputing*, 72(10 - 12):2581–2594, 2009.
- [20] Agus Widodo and Indra Budi. Model Selection Using Dimensionality Reduction of Time Series Characteristics. In *International Symposium on Forecasting, Seoul, South Korea*, 2013.
- [21] Mirko Kück, Sven F Crone, and Michael Freitag. Meta-Learning with Neural Networks and Landmarking for Forecasting Model Selection an Empirical Evaluation of Different Feature Sets Applied to Industry Data. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1499–1506. IEEE, 2016.
- [22] Christiane Lemke and Bogdan Gabrys. Meta-Learning for Time Series Forecasting in the NN GC1 Competition. In *International Conference on Fuzzy Systems*, pages 1–5. IEEE, 2010.

- [23] Thiyyanga S Talagala, Rob J Hyndman, and George Athanasopoulos. Meta-Learning How to Forecast Time Series. Technical report, Monash University, Department of Econometrics and Business Statistics, 2018.
- [24] Dabin Zhang, Shanying Chen, Ling Liwen, and Qiang Xia. Forecasting Agricultural Commodity Prices Using Model Selection Framework With Time Series Features and Forecast Horizons. *IEEE Access*, 8:28197–28209, 2020.
- [25] G Peter Zhang. Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model. *Neurocomputing*, 50:159–175, 2003.
- [26] Sibarama Panigrahi and Himansu Sekhar Behera. A Hybrid ETS–ANN Model for Time Series Forecasting. *Engineering Applications of Artificial Intelligence*, 66:49–59, 2017.
- [27] Ina Khandelwal, Ratnadip Adhikari, and Ghanshyam Verma. Time Series Forecasting Using Hybrid ARIMA and ANN Models Based on DWT Decomposition. *Procedia Computer Science*, 48(1):173–179, 2015.
- [28] Nian Liu, Qingfeng Tang, Jianhua Zhang, Wei Fan, and Jie Liu. A Hybrid Forecasting Model with Parameter Optimization for Short-Term Load Forecasting of Micro-Grids. *Applied Energy*, 129:336–345, 2014.
- [29] Jinliang Zhang, YiMing Wei, Zhong-fu Tan, Wang Ke, and Wei Tian. A Hybrid Method for Short-Term Wind Speed Forecasting. *Sustainability*, 9(4):596, 2017.
- [30] Christoph Bergmeir, Rob J Hyndman, and José M Benítez. Bagging Exponential Smoothing Methods Using STL Decomposition and Box–Cox Transformation. *International journal of forecasting*, 32(2):303–312, 2016.
- [31] Sean J Taylor and Benjamin Letham. Forecasting at Scale. *The American Statistician*, 72(1):37–45, 2018.
- [32] Foued Saâdaoui and Hana Rabbouch. A Wavelet-Based Hybrid Neural Network for Short-Term Electricity Prices Forecasting. *Artificial Intelligence Review*, 52(1):649–669, 2019.
- [33] Foued Saâdaoui, Hayet Saadaoui, and Hana Rabbouch. Hybrid Feedforward ANN with NLS-Based Regression Curve Fitting for US Air Traffic Forecasting. *Neural Computing and Applications*, pages 1–13, 2019.
- [34] Slawek Smyl. A Hybrid Method of Exponential Smoothing and Recurrent Neural Networks for Time Series Forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.
- [35] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and Machine Learning Forecasting Methods: Concerns and Ways Forward. *PloS one*, 13(3):e0194889, 2018.
- [36] Masashi Sugiyama and Motoaki Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. The MIT Press, 2012.
- [37] Victor M Guerrero. Time-Series Analysis Supported by Power Transformations. *Journal of Forecasting*, 12(1):37–48, 1993.
- [38] Tianqi Chen and Carlos Guestrin. Xgboost: A Scalable Tree Boosting System. In *ACM SIGKDD 2016*, pages 785–794. ACM, 2016.
- [39] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.
- [40] Thomas G Dietterich. Machine learning for Sequential Data: A Review. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 15–30. Springer, 2002.
- [41] Rob J Hyndman and Yeasmin Khandakar. Automatic Time Series Forecasting: The Forecast Package for R. *Journal of Statistical Software*, 26(3):1–22, 2008.
- [42] John R Rice. The Algorithm Selection Problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.
- [43] Kate A Smith-Miles. Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection. *ACM Computing Surveys (CSUR)*, 41(1):1–25, 2009.
- [44] André Bauer, Marwin Züfle, Johannes Grohmann, Norbert Schmitt, Nikolas Herbst, and Samuel Kounev. An Automated Forecasting Framework based on Method Recommendation for Seasonal Time Series. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering, ICPE '20*, page 48–55, New York, NY, USA, April 2020. Association for Computing Machinery (ACM).
- [45] Christiane Lemke and Bogdan Gabrys. Meta-learning for Time Series Forecasting and Forecast Combination. *Neurocomputing*, 73(10-12):2006–2016, 2010.

- [46] James Durbin and Geoffrey S Watson. Testing for Serial Correlation in Least Squares Regression: I. *Biometrika*, 37(3/4):409–428, 1950.
- [47] L Breiman, JH Friedman, R Olshen, and CJ Stone. Classification and Regression Trees. 1984.
- [48] J Ross Quinlan. Combining Instance-Based and Model-Based Learning. In *Proceedings of the tenth international conference on machine learning*, pages 236–243, 1993.
- [49] Thomas Grubinger, Achim Zeileis, and Karl-Peter Pfeiffer. emtree: Evolutionary Learning of Globally Optimal Classification and Regression Trees in R. Technical report, Working Papers in Economics and Statistics, 2011.
- [50] Rob Hyndman, George Athanasopoulos, Christoph Bergmeir, Gabriel Caceres, Leanne Chhay, Mitchell O’Hara-Wild, Fotios Petropoulos, Slava Razbash, Earo Wang, and Farah Yasmeen. *forecast: Forecasting Functions for Time Series and Linear Models*, 2018. R package version 8.4.
- [51] Leo Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001.
- [52] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support Vector Regression Machines. In *Advances in neural information processing systems*, pages 155–161, 1997.
- [53] André Bauer, Marwin Züfle, Simon Eismann, Johannes Grohmann, Nikolas Herbst, and Samuel Kounev. Libra: A benchmark for time series forecasting methods. In *Proceedings of the 12th ACM/SPEC International Conference on Performance Engineering (ICPE)*, New York, NY, USA, April 2021. ACM.
- [54] Rob J Hyndman, Anne B Koehler, Ralph D Snyder, and Simone Grose. A State Space Framework for Automatic Forecasting Using Exponential Smoothing Methods. *International Journal of Forecasting*, 18(3):439–454, 2002.
- [55] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.
- [56] V Assimakopoulos and Konstantinos Nikolopoulos. The Theta Model: A Decomposition Approach to Forecasting. *International journal of forecasting*, 16(4):521–530, 2000.
- [57] G.E.P. Box and G.M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1970.
- [58] Spyros Makridakis and Michele Hibon. The M3-Competition: Results, Conclusions and Implications. *International journal of forecasting*, 16(4):451–476, 2000.
- [59] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.
- [60] Milton Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the american statistical association*, 32(200):675–701, 1937.

Table 1: Meta-Level Attributes

ID	Description
S1	The <i>frequency</i> specifies the length of the most dominant recurring pattern (e.g., 365 daily observations in a yearly pattern) within the de-trended time series.
S2	The <i>length</i> counts the total number of observations included in the de-trended time series.
S3	The <i>standard deviation</i> measures the amount of variations within the de-trended time series.
S4	The <i>skewness</i> reflects the symmetry of the value distribution of the de-trended time series.
S5	The <i>irregular skewness</i> quantifies the skewness of the irregular part of the time series.
S6	The <i>irregular kurtosis</i> reflects the tailedness of the value distribution of the irregular part of the de-trended time series.
B1	The <i>mean period entropy</i> quantifies the regularity and unpredictability of fluctuations of the de-trended time series. For this purpose, the approximate entropy of each full period is calculated and then averaged.
B2	The <i>coefficient of entropy variation</i> describes the standardized entropy distribution over all periods.
B3	The <i>mean cosine similarity</i> states how similar all periods are to each other. The similarity is expressed with the average cosine similarity of each full pair of periods.
B4	The <i>sinus approximation</i> quantifies how well the seasonal pattern of the time series can be approximated by a sinus wave. To this end, the Durbin-Watson statistic [46] is used to measure the autocorrelation of the resulting fitted errors.
L1	The 2^{nd} <i>frequency</i> states the second most dominant frequency of the de-trended time series.
L2	The 3^{rd} <i>frequency</i> refers to the third most dominant frequency of the de-trended time series.
L3	The <i>maximum spectral value</i> specifies the maximum spectral value of the periodogram ²² applied to the de-trended time series.
L4	The <i>number of peaks</i> reflects how many strong recurring

Table 2: Forecast error and time-to-result comparison on all time series.

Measures	BETS	ES-RNN	FFORMS	Hybrid	Prophet	sARIMA	Telescope	Telescope*	XGBoost
\bar{e} [%]	25.52	47.87	21.15	27.89	35.56	20.63	19.95	19.46	23.85
σ_e [%]	34.25	66.99	36.87	89.76	$2.30 \cdot 10^2$	35.63	31.35	27.96	34.67
\bar{t}_N	$6.14 \cdot 10^4$	$1.61 \cdot 10^6$	$5.23 \cdot 10^5$	$1.09 \cdot 10^6$	$2.04 \cdot 10^3$	$8.48 \cdot 10^5$	$1.43 \cdot 10^2$	$3.21 \cdot 10^3$	6.73
σ_{t_N}	$1.07 \cdot 10^6$	$4.92 \cdot 10^6$	$7.93 \cdot 10^6$	$8.08 \cdot 10^6$	$6.15 \cdot 10^3$	$9.08 \cdot 10^6$	98.67	$1.33 \cdot 10^4$	15.59

Table 3: Forecast error and time-to-result comparison on seasonal and non-seasonal time series.

	Measures	BETS	ES-RNN	FFORMS	Hybrid	Prophet	sARIMA	Telescope	Telescope*	XGBoost
Seasonal	\bar{e} [%]	26.79	52.30	21.94	29.40	38.12	21.10	20.33	19.77	
	σ_e [%]	33.24	71.18	39.11	92.16	$2.49 \cdot 10^2$	37.65	32.83	28.96	
	\bar{t}_N	$7.14 \cdot 10^4$	$1.63 \cdot 10^6$	$6.11 \cdot 10^5$	$1.26 \cdot 10^6$	$2.22 \cdot 10^3$	$9.92 \cdot 10^5$	$1.56 \cdot 10^2$	$3.74 \cdot 10^3$	
	σ_{t_N}	$1.16 \cdot 10^6$	$4.92 \cdot 10^6$	$8.57 \cdot 10^6$	$8.73 \cdot 10^6$	$6.62 \cdot 10^3$	$9.81 \cdot 10^6$	93.05	$1.44 \cdot 10^4$	
Non-Seasonal	\bar{e} [%]	18.00	21.74	16.47	18.98	20.53	17.85	17.65	17.65	
	σ_e [%]	20.12	18.73	18.96	19.60	21.47	20.46	21.04	21.04	
	\bar{t}_N	$2.53 \cdot 10^3$	$1.54 \cdot 10^6$	$1.74 \cdot 10^2$	$9.55 \cdot 10^4$	$9.26 \cdot 10^2$	84.73	65.04	65.04	
	σ_{t_N}	$4.42 \cdot 10^3$	$4.90 \cdot 10^6$	$8.20 \cdot 10^2$	$2.07 \cdot 10^5$	$1.07 \cdot 10^3$	$1.25 \cdot 10^2$	95.05	95.05	

Table 4: Distribution of time series in each quadrant for each forecasting method.

		BETS	ES-RNN	FFORMS	Hybrid	Prophet	sARIMA	Telescope	Telescope*	XGBoost
QI:	$[\bar{t}_N; \infty[\times [\bar{e}; \infty[$	46%	60%	21%	45%	22%	30%	1%	15%	0%
QII:	$[0; \bar{t}_N[\times [\bar{e}; \infty[$	4%	0%	26%	0%	33%	17%	47%	33%	50%
QIII:	$[0; \bar{t}_N[\times [0; \bar{e}[$	5%	0%	34%	0%	36%	22%	51%	42%	50%
QIV:	$[\bar{t}_N; \infty[\times [0; \bar{e}[$	45%	40%	19%	55%	10%	31%	1%	9%	0%