Received: 7 February 2022 | Revised: 20 May 2022 | Accepted: 4 July 2022

DOI: 10.1002/widm.1475

ADVANCED REVIEW

WIREs DATA MINING AND KNOWLEDGE DISCOVERY WILEY

Review of automated time series forecasting pipelines

Stefan Meisenbacher[1] | **Marian Turowski**[1] | **Kaleb Phipps**[1] | **Martin Rätz**[2] | **Dirk Müller**[2,3] | **Veit Hagenmeyer**[1] | **Ralf Mikut**[1]

[1]Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany

[2]Institute for Energy Efficient Buildings and Indoor Climate, RWTH Aachen University, Aachen, Germany

[3]Forschungszentrum Jülich GmbH, Institute of Energy and Climate Research – Energy Systems Engineering (IEK-10), Jülich, Germany

Correspondence
Stefan Meisenbacher, Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, 76344, Germany.
Email: stefan.meisenbacher@kit.edu

Abstract

Time series forecasting is fundamental for various use cases in different domains such as energy systems and economics. Creating a forecasting model for a specific use case requires an iterative and complex design process. The typical design process includes five sections (1) data preprocessing, (2) feature engineering, (3) hyperparameter optimization, (4) forecasting method selection, and (5) forecast ensembling, which are commonly organized in a pipeline structure. One promising approach to handle the ever-growing demand for time series forecasts is automating this design process. The article, thus, reviews existing literature on automated time series forecasting pipelines and analyzes how the design process of forecasting models is currently automated. Thereby, we consider both automated machine learning (AutoML) and

Abbreviations: ACF, autocorrelation function; ADF, augmented Dickey–Fuller; AIC, Akaike information criterion; ANN, artificial neural network; AR, autoregression; ARMA, autoregressive moving average; ARIMA, autoregressive integrated moving average; AutoML, automated machine learning; BE, backward elimination; BIC, Bayesian information criterion; BO, Bayesian optimization; BSTS, Bayesian structural time series; CASH, combined algorithm selection and hyperparameter optimization; CNN, convolutional neural network; DBN, deep belief network; DBSCAN, density-based spatial clustering of applications with noise; DEA, differential evolution algorithm; DES, double exponential smoothing; DT, decision tree; E2E, end-2-end; EA, evolutionary algorithm; EDA, estimation distribution algorithm; ELM, extreme learning machine; EMD, empirical mode decomposition; ENN, Elman neural network; ES, exponential smoothing; ETS, error trend seasonality; FCNN, fully connected neural network; FNN, fuzzy neural network; FS, forward selection; GA, genetic algorithm; GBM, gradient boosting machine; GOF, goodness-of-fit; GP, Gaussian process; GRNN, general regression neural network; HPO, hyperparameter optimization; IC, information criteria; IMA, integrated moving average; INF, iterative neural filter; IQR, inter-quartile range; kNN, k-nearest neighbors; KPSS, Kwiatkowski–Phillips–Schmidt–Shin; KS, Kolmogorov–Smirnov; LASSO, least absolute shrinkage and selection operator; LC, locally constant; LDA, linear discriminant analysis; LL, locally linear; LogR, logistic regression; LR, linear regression; LSTM, long short-term memory; MA, moving average; MAD, median absolute deviation; MAE, mean average error; MAPE, mean absolute percentage error; MARS, multivariate adaptive regression splines; MASE, mean absolute scaled error; MIQP, mixed integer quadratic programming; MKL, multiple kernel learning; MLP, multilayer perceptron; MSE, mean squared error; MSTL, multiple seasonal-trend decompositions using loess; NLP, non-linear programming; NNetAR, neural network autoregression; OCSB, Osborn–Chui–Smith–Birchenhall; PACF, partial auto-correlation function; PCA, principal component analysis; PR, polynomial regression; PSO, particle swarm optimization; RBFNN, radial basis function neural network; RF, random forest; RMSE, root mean squared error; RNN, recurrent neural network; RPaRT, recursive partitioning and regression trees; RW, random walk; sARIMA, seasonal autoregressive integrated moving average; SES, simple exponential smoothing; SETARMA, self-exciting threshold autoregressive moving average; sMAPE, symmetric mean absolute percentage error; SMNM, single multiplicative neuron model; STL, seasonal and trend decomposition using loess; SVD, singular value decomposition; SVM, support vector machine; SVR, support vector regression; TBATS, trigonometric Box–Cox transform, ARMA errors, trend, and seasonal components; TES, triple exponential smoothing; TPE, tree parzen estimator; UC, unexplored component; VAR, vector autoregression.

Funding information
Bundesministerium für Wirtschaft und Energie, Grant/Award Number: 03ET1568; Deutsche Forschungsgemeinschaft, Grant/Award Number: 2153; Helmholtz-Gemeinschaft

Edited by: Justin Wang, Associate Editor and Witold Pedrycz, Editor-in-Chief

automated statistical forecasting methods in a single forecasting pipeline. For this purpose, we first present and compare the identified automation methods for each pipeline section. Second, we analyze these automation methods regarding their interaction, combination, and coverage of the five pipeline sections. For both, we discuss the reviewed literature that contributes toward automating the design process, identify problems, give recommendations, and suggest future research. This review reveals that the majority of the reviewed literature only covers two or three of the five pipeline sections. We conclude that future research has to holistically consider the automation of the forecasting pipeline to enable the large-scale application of time series forecasting.

This article is categorized under:
Technologies > Machine Learning
Technologies > Prediction
Algorithmic Development > Spatial and Temporal Data Mining

KEYWORDS
automated machine learning, AutoML, hyperparameter optimization, pipeline, time series forecasting

1 | INTRODUCTION

One of the most prominent forms of collected data is time series. In a time series, the data is arranged sequentially, and each value is explicitly time-stamped, with information such as date and time (i.e., value and time stamp constitute an observation). The progression of a time series over a certain period of time in the future, also known as the forecast horizon, is the subject of time series forecasting. Time series forecasting is applied with various forecast horizons at different temporal scales and aggregation levels in various domains (Hyndman & Athanasopoulos, 2021). Exemplary use cases from different domains are the following: sales forecasting for inventory optimization (*just-in-time supply chain*: Boone et al., 2019), forecasting the generation of renewable energy and the electricity demand in an area to balance the power grid load (*smart grids*: Ahmad et al., 2020), and forecasting the spread of the novel coronavirus COVID-19 (*pandemic control*: Rahimi et al., 2021). As the number and importance of use cases grow, the demand for time series forecasts is increasing steadily.

Designing a time series forecast for a particular use case typically incorporates five sections. The first section of the design process is the data preprocessing to transform the raw data into a desirable form for the forecasting method (Shaukat et al., 2021; Wang & Wang, 2020). The second section is feature engineering, which aims to extract hidden characteristics of the considered time series or to identify useful exogenous information for the forecasting method (Zebari et al., 2020). Each forecasting method contains hyperparameters that have to be set by the data scientist. Therefore, the third section, the hyperparameter optimization (HPO), intends to improve the forecast accuracy over the default hyperparameter configuration (Hutter et al., 2019). Given candidate forecasting methods with optimized hyperparameters, selecting the most suitable forecasting method is crucial for forecast accuracy and is addressed in the fourth section (Zöller & Huber, 2021). The fifth section aims to increase the robustness of the forecast by forecast ensembling (Hajirahimi & Khashei, 2019), that is, bundling multiple forecasts of different forecasting models to avoid occasional poor forecasts (Shaub, 2020).

The above sections of the design process are commonly organized in a pipeline structure as shown in Figure 1. Currently, these forecasting pipelines are typically designed manually (Hyndman & Athanasopoulos, 2021; Petropoulos et al., 2022). However, this is becoming increasingly difficult due to the large number of sophisticated methods developed in recent years, as exhaustively summarized by Petropoulos et al. (2022). Considering all these available methods and manually tailoring the forecasting pipeline to a specific use case is time-consuming and challenging because selecting appropriate methods for the pipeline sections is an iterative process and requires expert knowledge. This expert knowledge is particularly crucial, as the forecast accuracy is sensitive to various design decisions

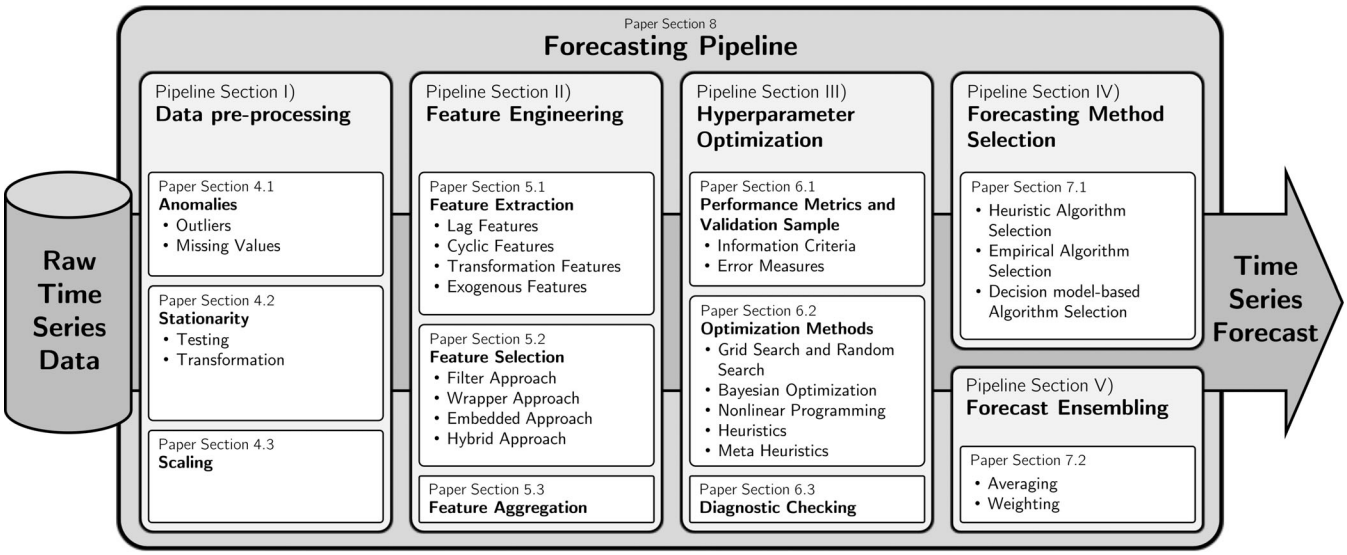

FIGURE 1 The forecasting pipeline systematizes the design process for time series forecasting using five pipeline sections

(Hutter et al., 2019). It is also foreseeable that the number of knowledgeable data scientists cannot handle the ever-growing demand for time series forecasts in the future. Therefore, increasing the efficiency of the design process by automation is required (Tuggener et al., 2019).

To automate design decisions and remove the data scientist from the iteratively performed parts of the design process, a variety of automation methods are available for each section of the forecasting pipeline.[1] The sequential organization of these automation methods and the management of the data flow can be realized by creating a pipeline (Heidrich et al., 2021; Löning et al., 2019). Running the created forecasting pipeline executes all included automation methods and trains the forecasting method—for example, a linear regression (LR)—with historical time series data, resulting in a fitted forecasting model. Thereby, the pipeline automates the design process.

In this context, the long-term objective toward full automation has motivated numerous researchers and led to promising research results in the fields related to this review study, as shown in Figure 2. Several surveys and review studies analyze the automation of single forecasting pipeline sections such as preprocessing (Shaukat et al., 2021; Wang & Wang, 2020), feature engineering (Zebari et al., 2020), HPO and forecasting method selection (Hutter et al., 2019; Zöller & Huber, 2021), and forecast ensembling (Hajirahimi & Khashei, 2019). Moreover, rather than focusing on the automated design of the entire forecasting pipeline, existing studies on time series forecasting only consider the statistical or machine learning forecasting methods themselves (De Gooijer & Hyndman, 2006; Han et al., 2019; Taieb et al., 2012). However, a comprehensive study is lacking that reviews how the design process of the entire time series forecasting pipeline is currently automated, and that also considers the families of both statistical and machine learning methods.

Therefore, the article reviews existing literature on automated time series forecasting pipelines and analyzes how the design process of forecasting models is currently automated. For this, we consider literature from various research directions, including statistical forecasting, machine learning, and deep learning.[2] More specifically, we focus on the interaction and combination of automation methods within the pipeline sections considering both automated machine learning (AutoML) and automated statistical forecasting methods. For this purpose, we first present and systematically compare existing automation methods from the identified literature used in each pipeline section. Second, we analyze the complete forecasting pipeline considering how many pipeline sections are automated in the reviewed literature and highlighting the interaction and dependencies between pipeline sections. For both, we discuss the reviewed literature, identify potential problems, give recommendations, and suggest future research.

After describing the methodology in Section 2 and a brief introduction to time series forecasting in Section 3, this article is organized by respective sections following the forecasting pipeline shown in Figure 1 and concludes in Section 9.

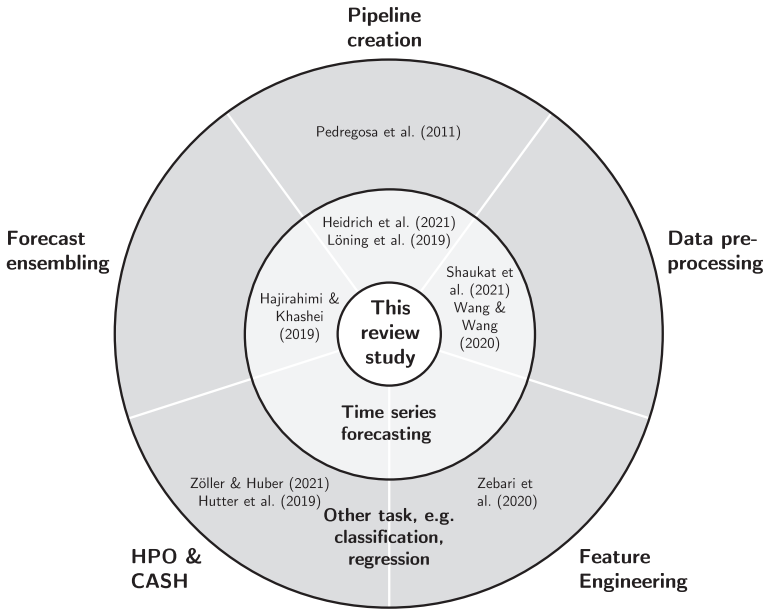

FIGURE 2 Related to this review are the fields of pipeline creation, data preprocessing, feature engineering, hyperparameter optimization (HPO) and combined algorithm selection and hyperparameter optimization (CASH), and forecast ensembling

2 | METHODOLOGY

The methodology of this literature review applies the following fundamental steps suggested by Webster and Watson (2002) for the identification of major contributions, their origin, and evolution.

Literature search: A search-term-based exploration of research articles considering title, abstract, and keywords is conducted using *Scopus*,[3] resulting in 359 hits.[4] Potential predatory journals and publishers, and vanity press listed in *Beall's List*[5] are excluded.

Literature screening: We screen the abstracts for relevance with the following criteria: (1) the task type must be time series forecasting, and (2) at least one iterative element in the forecasting pipeline that previously required human intervention must be automated.

Backward –forward search: We identify additional articles that cite or are cited by the articles passing the screening. The obtained candidates also undergo the screening procedure defined above. Backward–forward search yield a pool of 2152 additional papers, which the keyword filtering reduces to 242 articles.[4]

Review: We present automation methods for each section of the forecasting pipeline and review the articles that pass the screening and full-text analysis. After the abstract screening, 144 of the 601 papers remain, and 71 after analyzing the full text. If there is similar work from a research group, we cite the article with the greatest methodological scope and articles that propose improvements, which reduces the included articles from 71 to 63 articles.[6]

Discussion: We discuss the contributions towards design automation individually for each section of the forecasting pipeline. Afterward, we identify gaps and highlight future research directions by analyzing the coverage of the forecasting pipeline.

3 | TIME SERIES FORECASTING

A time series $\{\mathbf{y}[k]; k = 1, 2..., K\}$ reflects a set of $K \in \mathbb{N}^{>0}$ observations typically measured at equidistant points in time (Brockwell & Davis, 2016). A time series forecasting model $f(\cdot)$ estimates future values $\hat{y}$ for one or more time points—the forecast horizon $H \in \mathbb{N}^{>0}$—using current and past values (González Ordiano et al., 2018). It is defined as

$$\begin{aligned}\hat{y}[k+H] = f(&y[k],...,y[k-H_1]\\ &\mathbf{u}^\top[k],...,\mathbf{u}^\top[k-H_1]\\ &\hat{\mathbf{u}}^\top[k],...,\hat{\mathbf{u}}^\top[k-H_1]\\ &\mathbf{w});k,H,H_1 \in \mathbb{N}^{>0};k>H_1\end{aligned} \tag{1}$$

where $H_1 \in \mathbb{N}^{>0}$ indicates the horizon for past values $k-H_1$, the vector $\mathbf{w}$ contains the model's parameters, the vector $\mathbf{u}^\top$ denotes values from exogenous time series, the vector $\hat{\mathbf{y}}^\top$ indicates that the exogenous values originate from another forecast, and y represents values of the target time series (González Ordiano et al., 2018).[7]

In time series forecasting, the following four families of methods exist naïve methods, statistical methods, machine learning methods, and hybrid methods. The families and their main representatives are briefly introduced in the following. Subsequently, we summarize common training methods for fitting the respective forecasting model.

3.1 | Naïve forecasting methods

The simplest method family of forecasting are naïve methods. Common representatives are the averaging method, where the forecasts of all future values are equal to the average of historical data, and the Random Walk (RW) method, where the value of the last observation is used as forecast. For the RW, modifications are available for data with seasonality (sRW) and drift (dRW). The sRW method is a modification of the RW, where the value of the last observation of the same season is used as forecast (e.g., the same day of the previous week). In the dRW method, the drift is assumed to be the average change observed in the historical data and is added to the RW (Hyndman & Athanasopoulos, 2021).

3.2 | Statistical forecasting methods

More sophisticated than naïve methods are statistical methods that use statistics based on historical data to forecast future time series values. The first representatives are autoregression (AR) methods. The autoregressive moving average (ARMA) method (Box et al., 2016) assumes a linear relationship between the lagged inputs and is applicable if the time series is stationary. If trends and seasonal characteristics are present, the time series is nonstationary, and the requirements for applying ARMA are not fulfilled. To address this problem, the autoregressive integrated moving average (ARIMA) method (Box et al., 2016) removes time series trends through differencing and the seasonal ARIMA (sARIMA) method eliminates the seasonality by seasonal differencing (Cheng et al., 2015). The second representatives are exponential smoothing (ES) methods, where the forecast is determined by a weighted average of past observations, with the weights decaying exponentially with their age (Hyndman & Athanasopoulos, 2021). Simple exponential smoothing (SES) is a valid forecasting method for time series data without a trend or seasonal pattern. For time series with a trend, SES is adapted to double exponential smoothing (DES), and triple exponential smoothing (TES) is suitable for time series with seasonality. An extensive discussion of statistical forecasting methods can be found in reference (De Gooijer & Hyndman, 2006).

3.3 | Machine learning forecasting methods

While most statistical forecasting methods are based on assumptions about the distribution of the time series data, machine learning methods have fewer restrictions in terms of linearity and stationarity (Cheng et al., 2015). In addition to statistical forecasting methods, which are specifically developed for time series forecasting, one can use regression methods based on machine learning to forecast multiple time points ahead using the following strategies, either solely or in combination (Taieb et al., 2012):

Recursive strategy: One trains a single regression model $f(\cdot)$ to forecast one-time point ahead. In the operation, one recursively feedbacks the output value to the input for the next time point.

Direct strategy: One trains multiple independent regression models $f_h(\cdot), h=1,...H$, each to forecast the value at time $k+h$.

Multiple output strategy: One trains a single regression model to forecast the whole horizon H at once. Consequently, the output is not a single value but a vector.

Representative machine learning methods are the support vector regression (SVR), decision tree (DT)-based methods like the gradient boosting machine (GBM) and the random forest (RF), fuzzy time series (FTS), and artificial neural networks (ANNs). The capabilities of these forecasting methods strongly depend on the task and the characteristics of the time series, respectively. Hence, data preprocessing and feature engineering are often essential, which in turn makes manual tailoring of the forecasting method to the task time-consuming. Since this review study focuses on automating the design process of forecasting pipelines, we refer to Masini et al. (2021) for more details on machine learning methods for time series forecasting and to Han et al. (2019) for an experimental comparison of different deep learning techniques applied to time series forecasting.

3.4 | Hybrid forecasting methods

The hybrid forecasting method family aims to combine different forecasting methods with their advantages to obtain a forecast that improves over using a single method. These hybrid forecasting methods are based on different hybridization approaches. The first often-used hybridization approach is ensembling, where input data is simultaneously fed into different forecasting methods, and their outputs are combined into one forecast. The second typical hybridization approach includes data preprocessing-based hybrid models. The original time series is decomposed into multiple time series components to simplify the forecasting problem. Then, each component can be addressed by an appropriate forecasting method before recomposing their outputs to a single forecast. In the third common approach for hybridization, several forecasting methods are sequentially connected. For example, the hybrid forecasting method of Kourentzes and Crone (2010) uses the MLP-based INF to identify periodicities in time series, and apply the result to automatically configure the TES method. Recent sequential hybridization approaches combine statistical forecasting methods with deep learning, for example, combining ES and a recurrent neural network (RNN). The ES-RNN captures the seasonality and level as the main components of the time series with ES and learns additional effects with the RNN to correct the ES forecast (Smyl, 2020). Since each hybridization approach involves several design decisions that can be automated, we consider hybrid forecasting methods as forecasting pipelines and only review literature that applies automation methods, according to our methodology defined in Section 2. More details on hybridization approaches in time series forecasting can be found in Hajirahimi and Khashei (2019).

3.5 | Training methods for model fitting

Training a forecasting method with historical time series data results in a fitted forecasting model. The appropriate training method is essential and strongly depends on the respective forecasting method.

Fitting naïve forecasting methods either requires very simple training methods like computing the arithmetic mean of the historical time series data for the naive averaging forecasting method or no training at all as in the RW. Statistical forecasting methods like the ARIMA or ES methods commonly use the maximum likelihood estimation to fit the model. This training method uses an optimization algorithm to find the model's weights that maximize the probability of obtaining the observed historical time series data. In practice, the logarithm of this likelihood is maximized, since the maximum is preserved after the log-transformation, but the optimization problem becomes simpler (Hyndman & Athanasopoulos, 2021).[8] For machine learning-based forecasting methods, the training methods are often very specific to the particular forecasting method. For example, the SVR maps the time series data into a high-dimensional space using a linear or nonlinear kernel function, where it searches for a linear hyperplane containing the maximum number of observations between the hyperplane and a given error margin ε with tolerance C for observations outside of ε. This search can be formulated as a constrained minimization problem in the form of quadratic programming and solved with appropriate solvers.[9] Training a DT, however, starts by sorting the time series data and taking the average as the initial forecast. It is followed by an iterative search for points where splitting the sorted data and taking the average on the data partitions increases the forecast accuracy. Various training methods for creating the decision

nodes exist, for example, the top-down greedy algorithm.[10] The GBM forms an ensemble by commonly starting with an initial DT and iteratively fitting new DTs to the residual between forecast $\hat{y}$ and realized value y, which are added to the ensemble to minimize the residual (boosting). Instead of boosting, the RF uses the bootstrap aggregating (bagging) ensemble learning method to create the DT-based ensemble. The bagging method repeatedly selects a random sub-set of the training data and fits a DT on this sub-set. For the RF forecast, the output of all DTs is averaged. Instead of partitioning data into crisp intervals as in DTs, FTS methods partition the data into fuzzy intervals, that is, an observation can belong to multiple intervals, quantifiable by the degree of membership. Training an FTS involves the steps of defining the so-called universe of discourse, partitioning the data, fuzzification, learning fuzzy logical relationships and building relationship groups, and finally, defining how to defuzzify the membership degrees into a crisp forecast.[11] Similar to the fuzzy logic, which is inspired by human reasoning, ANNs are also inspired by biology. More specifically, ANNs are modeled after biological neural networks, that is, input signals feed-forward through a network of artificial neurons connected in a series of layers, and, depending on the input signal, different connections can be activated, similar to the synapses in a biological brain. The weights of these connections and neurons are usually estimated iteratively based on gradient descent using backpropagation for efficient gradient computation or based on meta-heuristics.[12]

In addition to the training methods, the loss function, that is, the target function that is optimized during training, is crucial for the model fitting. As described above, the training of most statistical forecasting methods aims to maximize the log-likelihood, while machine learning methods often minimize an error measure, the so-called loss function. Common loss functions to be minimized during training are the mean average error (MAE) and the mean squared error (MSE; both described in Section 6.1). While minimizing the MAE results in a model that forecasts the median, minimizing the MSE results in forecasting the mean (Hyndman & Athanasopoulos, 2021). Especially for ANNs, customizing the loss function is a possibility for manually tailoring the forecast to a task, for example, wind speed forecasting (Chen et al., 2022). Regarding automation, we consider the manual customization of the loss function as a possibility to further improve an automatically created forecasting model with expert knowledge. Therefore, we refer to Koutsandreas et al. (2021) for an overview of loss functions in time series forecasting.

As outlined, the training method for fitting the forecasting model is often tied to the corresponding forecasting method. Apart from the forecasting method, other methods in the pipeline also must be fitted, such as input scalers or feature selectors. For this reason, we do not consider training methods as a separate pipeline section. Training a forecasting pipeline rather comprises the successive fitting of all involved methods using corresponding training methods. Regarding automation, approaches exist to optimal configure these training methods by HPO. Examples include optimizing the learning rate for training MLPs (Donate & Cortez, 2014) or even connecting pipeline sections, such as in the SVR-based embedded feature selection (Valente & Maldonado, 2020). We detail each approach in the corresponding section of this article.

4 | DATA PREPROCESSING

Since most forecasting methods rely on assumptions about data properties, data preprocessing is of crucial importance. Data preprocessing includes anomaly detection and handling, transforming the time series to make it stationary, and scaling the time series. In the following subsections, automated methods for data preprocessing are introduced, and their utilization in forecasting pipelines is exemplified by the reviewed literature.

4.1 | Anomalies

An anomaly is a value that significantly deviates from the rest of the time series (Chandola et al., 2009). Anomalies are induced by rare events or by errors in the data. Apart from anomalous existing values, which we call outliers, anomalies also comprise missing values in the time series. Both outliers and missing values can degrade the forecast accuracy or cause the training to fail. Therefore, appropriate anomaly detection and handling are necessary. Table 1 shows the summary of automated anomaly detection and handling methods used in the literature for time series forecasting pipelines.

TABLE 1 Summary of automated anomaly detection and handling methods for data preprocessing in time series forecasting pipelines

References	Outlier detection	Outlier handling	Missing value handling
Liu et al., 2017	glob. Mean-var. thresh.	Average nearest	
Martínez, Charte, et al., 2019; Yan, 2012	loc. Median thresh.	Average nearest	
Fan et al., 2019	loc. Median thresh.	Average nearest	Median imput.
Widodo et al., 2016	loc. Median-MAD thresh.	Median nearest	
Maravall et al., 2015	ARIMA-MAD thresh.	ARIMA values	
Züfle & Kounev, 2020	glob. Median-robust-var. thresh.	Linear interpolation	Copy-paste imput.

Abbreviations: glob., global; imput., imputation; loc., local; thresh., threshold; var., variance.

4.1.1 | Outlier detection and handling

Automated outlier detection and handling aim to identify abnormal values and replace them with plausible values without human intervention. Liu et al. (2017) define an interval based on the global mean and the variance of the time series

$$[\text{mean}(y) - \alpha_t \cdot \text{var}(y), \text{mean}(y) + \alpha_t \cdot \text{var}(y)], \tag{2}$$

with the threshold α_t and the anomaly detection method considers values outside the interval as outliers. Detected abnormal values are automatically substituted with the arithmetical averages of the nearest previous and posterior normal values. However, anomalous values themselves bias the estimation of the mean and the variance, and the method is only valid for stationary time series. Other authors tackle this weakness by calculating the local median instead of the global mean. Martínez, Charte, et al. (2019), Yan (2012), and Fan et al. (2019) consider an observation as an outlier if its absolute value is four times greater than the absolute medians of the three consecutive points before and after the observation. However, only extreme values above the absolute medians are detected, and extreme values below the absolute medians are not identified. Widodo et al. (2016) apply the Hampel method, which automatically replaces any value that deviates from the median of its neighbors by more than three median absolute deviations (MAD) with that median value. Unlike previous methods, which use statistical measures, the anomaly detection and handling method of Maravall et al. (2015) is based on a forecasting model. The method fits an ARIMA model, evaluates the MAD of the estimation residuals, and automatically replaces detected outliers with the forecast of the ARIMA model.

4.1.2 | Missing value handling

Automated missing value handling aims to reconstruct absent observations without human assistance. The method of Fan et al. (2019) automatically replaces missing values in the time series with the median of 12 consecutive points before and after the observation. Yet, this method is prone to larger gaps of missing data. The method of Züfle and Kounev (2020) automatically imputes missing values by multiplying the known value one season before or after the missing value by the trend factor estimated between the day of the missing values and the day copied. Using this procedure in chronological order allows the imputed values for the imputation of subsequent missing values. After the missing value imputation, the authors apply a similar outlier detection method like (2) with $\alpha_t = 3$. Unlike Liu et al. (2017), Züfle and Kounev (2020) use the robust standard deviation between the 1st and 99th percentile of the data and replace the outliers by linearly interpolating between the two nearest nonanomalous values.[13]

4.2 | Stationarity

In a stationary time series $y[k]$, the statistical properties do not depend on the time of observation k, that is, the distribution of $y[k, \ldots, k+s]$ is independent of k for all s (Hyndman & Athanasopoulos, 2021). Therefore, a time series with

trends or seasonal patterns is not stationary because either the mean of the time series, its variance, or both change over time. Since some statistical forecasting methods assume a stationary time series, their application to nonstationary time series requires an appropriate transformation. Stationarity tests help to identify the type of nonstationarity and support the automated selection of the appropriate transformation. Table 2 shows the summary of automated stationarity testing and transformation methods used in the literature for time series forecasting pipelines, introduced in the following.

4.2.1 | Autocorrelation and differencing transformations

A first approach to automatically identify nonstationarities in time series is proposed by Tran and Reed (2004) based on the autocorrelation function (ACF) and the partial autocorrelation function (PACF). The ACF and the PACF visualize the correlation of a time series with a delayed copy of itself. The authors automatically detect decay patterns by calculating the average rate of change in the magnitude of high frequencies in the ACF and PACF, and consider rates of less than 10% as slow decay. The slow decay patterns indicate trends in the time series. To remove the trends, the time series is differenced by subtracting successive observations d times. After differencing, the ACF and PACF show significant peaks at regular intervals $s$if the time series is seasonal. To remove the seasonality, the time series is seasonally differenced D times by subtracting observations separated by s. Note that the ARIMA forecasting method explicitly includes differencing as hyperparameter d in the model structure and the sARIMA method additionally considers seasonal differencing with D and s.

TABLE 2 Summary of stationarity testing and transformation methods for data preprocessing in automated time series forecasting pipelines

References	Forecasting method(s)	Stationarity testing	Stationarity transformation
Tran & Reed, 2004	sARIMA	ACF, PACF pattern analysis	diff., s. diff.
Bauer, Züfle, Herbst, et al., 2020	Telescope	Periodogram	Box–Cox, STL
Kourentzes & Crone, 2010	MLP, TES	INF	INF
Crone & Kourentzes, 2010	MLP	ADF, INF	diff., INF
Alzyout et al., 2019	ARIMA	KPSS	diff.
Sekma et al., 2016	AR, VAR	KPSS	diff.
Hyndman & Khandakar, 2008	autoARIMA	KPSS, Canova–Hansen	diff., s. diff.
Eğrioğlu & Bas, 2022	SMNM	ACF, ADF	diff., s. diff
Lu & AbouRizk, 2009	sARIMA	ACF, PACF t-test	log, diff., s. diff.
Maravall et al., 2015	ARMA, sARIMA	log-level, Kendall–Ord, Pierce, Lytras, seasonal frequency peaks	log, diff., s. diff.
Liu et al., 2017	ARIMA	ADF, log-level	diff., log
Anvari et al., 2016	sARIMA	OCSB, KPSS, correlation, *t*-test, ADF	diff., s. diff., log
Amin et al., 2012	ARIMA, SETARMA	KS, KPSS	log, Box–Cox
Martínez, Frías, et al., 2019	kNN		diff., Box–Cox, STL
Fildes & Petropoulos, 2015	autoARIMA, Theta, Damped, RW, sRW, SES, DES, TES	Cox–Stuart	classical decomposition
Widodo et al., 2016	MKL		STL
Yan, 2012	GRNN	Heuristic autocorrelation	diff., s. diff.
Bandara et al. 2020	LSTM		diff., log, STL

Abbreviations: s., seasonal; diff., differencing.

4.2.2 | Frequency filters

Apart from the ACF and PACF, methods based on frequency filters are used to identify seasonality. Bauer, Züfle, Herbst, et al. (2020) use the periodogram to automatically retrieve all frequencies within the time series, iterate over the found frequencies, and match each frequency with reasonable frequencies (e.g., daily, hourly, and yearly) with tolerance to determine seasonal frequencies. Kourentzes and Crone (2010) propose the iterative neural filter (INF) to automatically identify seasonal frequencies. The filter distinguishes between stochastic and deterministic components and iteratively removes seasonalities, trends, and irregularities in the time series.

4.2.3 | Unit root tests

Statistical unit root tests are used to identify nonstationarities before applying transformation methods. The unit root tests used in the literature include the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test (Kwiatkowski et al., 1992) or the Cox–Stuart test (Cox & Stuart, 1955) for testing if the time series is stationary around a deterministic trend, the augmented Dickey–Fuller (ADF) test (Cheung & Lai, 1995) for the existence of stochastic trends in the time series, and the Canova–Hansen test (Canova & Hansen, 1995) or Osborn–Chui–Smith–Birchenhall (OCSB) test (Osborn et al., 1988) for the existence of seasonal patterns over time. According to the trends and seasonalities detected in the unit root tests, the time series is automatically differenced or seasonally differenced, respectively.

4.2.4 | Logarithm and normality transformations

Apart from unit root testing, Maravall et al. (2015) and Liu et al. (2017) propose log-level tests to automatically evaluate if a log-transformation of the time series is beneficial. Amin et al. (2012) use the Kolmogorov–Smirnov (KS) test (Lilliefors, 1967) to determine whether a time series is normally distributed—if not, the log-transformation is applied. Other authors apply the log-transformation without testing to reduce the variance (Anvari et al., 2016; Bandara et al., 2020) or apply various transformations until critical values are satisfied in t-tests of ACF and PACF (Lu & AbouRizk, 2009). For achieving normality and stabilizing the variance, the Box–Cox transformation (Box & Cox, 1964) is often applied without preceding testing, like in references (Amin et al., 2012; Bauer, Züfle, Grohmann, et al., 2020; Martínez, Charte, et al., 2019; Züfle & Kounev, 2020).

4.2.5 | Time series decomposition

In addition to transformation operations to achieve stationarity, time series can be decomposed into the components trend-cycle $T[k]$, season $S[k]$, and irregular $R[k]$ (i.e., the residual), as shown in Figure 3. Two decomposition approaches are possible—the additive decomposition $y[k] = T[k] + S[k] + R[k]$ and the multiplicative decomposition $y[k] = T[k] \cdot S[k] \cdot R[k]$ (Hyndman & Athanasopoulos, 2021). Afterward, each component can be handled by an individual forecasting model (i.e., preprocessing-based hybrid modeling), recombining their outputs according to the respective decomposition approach. Popular decomposition methods are the so-called classical decomposition, the seasonal and trend decomposition using loess (STL), the X-11 method, and the SEATS method, detailed in Hyndman and Athanasopoulos (2021), as well as the Fourier transformation, the wavelet transformation, and the empirical mode decomposition (EMD), surveyed in Rios and de Mello (2012). In automated forecasting pipelines, the most common decomposition method is STL which is used in the references (Bandara et al., 2020; Bauer, Züfle, Herbst, et al., 2020; Martínez, Charte, et al., 2019; Widodo et al., 2016) and can handle any type of seasonality automatically.

4.3 | Scaling

The scale of time series can have an adverse effect on forecasting methods based on machine learning. If the range of values is large, learning methods based on gradient descent may converge much slower or fail due to instability.

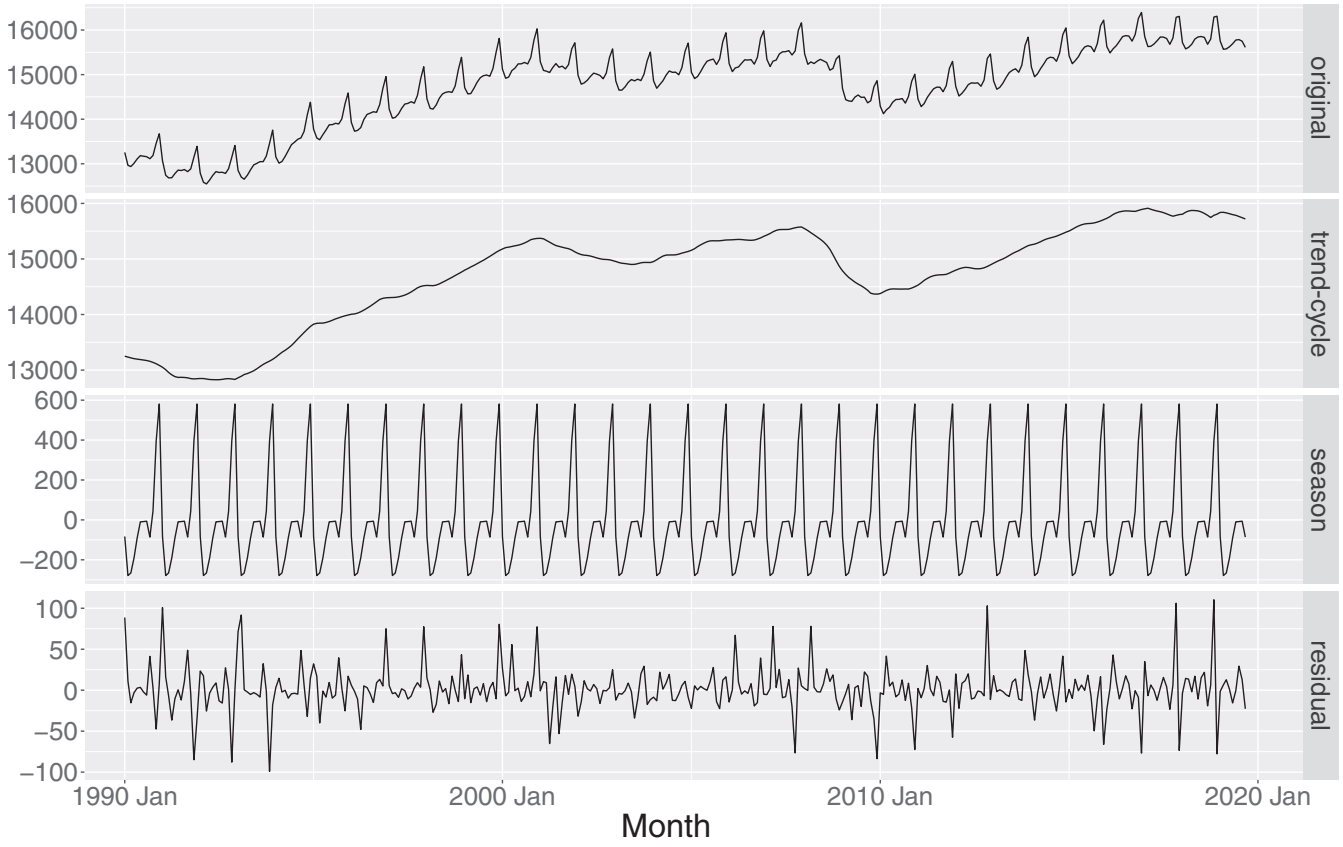

FIGURE 3 Decomposition of a time series into the trend-cycle, season, and irregular components with the additive seasonal and trend decomposition using loess (STL; Hyndman & Athanasopoulos, 2021)

Additionally, in the case of multiple inputs with different scales, the inputs with larger variance dominate the others in the calculation of many distance measures (Pedregosa et al., 2011).

The general formalization for time series scaling is

$$y' = \frac{y - a}{b}. \tag{3}$$

Table 3 shows the scaling methods used in the literature on automated forecasting pipelines compared to the family of the forecasting method. A common approach is min–max scaling, where the time series are scaled in the range $[0,1]$ with $a = \min(y), b = \max(y) - \min(y)$. While min–max scaling guarantees that all-time series are scaled to the same range $[0,1]$, the Z-score normalization scales the time series to zero-mean and unit-variance with $a = \text{mean}(y), b = \text{var}(y)$. Both scaling methods are sensitive to outliers and thus require a preceding anomaly detection and handling. Alternatively, robust scaling methods like *scikit-learn's RobustScaler*[14] can be used, which removes the median and scales the data according to the inter-quartile range (IQR).

4.4 | Discussion

We discuss data preprocessing as the first section of the automated forecasting pipeline, show possible problems, give recommendations, and suggest future research.

As shown in Table 2, automated stationarity testing and transformation methods are predominantly applied to pipelines using methods of the statistical forecasting family. In contrast, Table 3 shows that time series scaling is mainly used in pipelines with methods of the machine learning family. Both forecasting families require methods for automated anomaly detection and handling, shown in Table 1.

TABLE 3 Summary of scaling methods for data preprocessing applied in time series forecasting pipelines

Forecasting method family	Min–max	Zero-mean	*Z*-score
Statistical	Sekma et al., 2016	Liu et al., 2017	Dellino et al., 2018a
Machine learning	Bauer, Züfle, Grohmann, et al., 2020; Züfle et al., 2019; Kourentzes & Crone, 2010; Kourentzes & Crone, 2010; Crone & Kourentzes, 2010; Martínez, Charte, et al., 2019; Violos et al., 2020; Maldonado et al., 2019; Yan, 2012; Panigrahi & Behera, 2020; Widodo & Budi, 2013; Sergio et al., 2016; Donate et al., 2013; Donate & Cortez, 2014		Rätz et al., 2019; Ma & Fildes, 2021; Widodo & Budi, 2013; Widodo et al., 2016; Sagaert et al., 2018

Regardless of their predominant use for automation, both transformation methods—for achieving stationarity and scaling—can be adversely affected by anomalies. Therefore, it is essential that automated anomaly detection and handling is performed before time series transformations. In automated outlier detection and handling, the main concern is that methods applied in the literature on automated forecasting pipelines focus on single outliers, although methods for consecutive outliers are widely available (Blázquez-García et al., 2021). Due to this focus, these methods may have a limited detection and handling performance if several consecutive outliers or abnormal patterns are present. For automated forecasting pipelines, we thus recommend the first step of preprocessing to be detecting and removing isolated outliers, followed by detecting consecutive anomalous values—outliers or missing values—and handling them with an appropriate method. Ideally, the handling also considers domain-specific knowledge (e.g., Weber et al., 2021).

The second step of preprocessing, automated testing for stationarity and time series transformation, is only crucial if the pipeline uses a forecasting method that assumes a stationary time series. For each condition—deterministic and stochastic trends, seasonalities, and normality—only one test must be applied because the use of multiple tests may lead to conflicting answers (Hyndman & Athanasopoulos, 2021). Time series decomposition methods are also suitable to simplify the forecasting problem. After applying the decomposition, each component can be treated with an appropriate forecasting method. In the reviewed literature, the STL decomposition method is predominantly used.[15] Automated methods like STL provide robust default hyperparameters, but these may need to be adapted to the forecasting problem (Hyndman & Athanasopoulos, 2021). Consequently, tailoring the hyperparameters or even choosing the most suitable decomposition method and its hyperparameters (i.e., CASH) can be approached as a hyperparameter optimization problem. However, this approach is not considered in the reviewed literature.

In the third step of preprocessing, we recommend using an appropriate time series scaling method to facilitate the training process of the respective forecasting method.

Given these recommendations, future work on automated forecasting pipelines should also consider domain knowledge to identify anomalies, for example, if only positive values are valid or domain-adapted imputation to improve the reconstruction. Additionally, it should be systematically evaluated whether stationarity transformations improve the forecast accuracy of machine learning methods if applied in the forecasting pipeline.

5 | FEATURE ENGINEERING

A time series feature reflects the observations of an explanatory variable in the process being forecast (i.e., the target variable). Table 4 shows feature engineering as a subsection of the time series forecasting pipeline, consisting of extraction and selection of features.

5.1 | Feature extraction

The feature space of the training data set contains for each explanatory variable a time series of the same resolution and length as the target variable.[16] Feature extraction aims at automatically enriching the feature space with additional

TABLE 4 Summary of feature engineering methods in automated time series forecasting pipelines

References	Forecasting method(s)	Data domain	Feature extraction				Feature selection			Feature aggregation
			Lag	Cyc.	Tran.	Exo.	Filter	Wrapper	Embedded	
Cerqueira et al., 2021	AR, autoARIMA, ETS, TBATS	Several (5)	X	–	X	–				
Züfle & Kounev, 2020	GBM, RF	Human access	X	X	–	X				
Bauer, Züfle, Grohmann, et al., 2020	Telescope	Several (5)	–	X	X	–	Spectral value			
Chakrabarti & Faloutsos, 2002	kNN	Nature	X	–	–	–	Fractal dimension			
Martínez, Frías, et al., 2019	GRNN	Economics	X	–	–	–	PACF			
Yan, 2012	GRNN	Energy	X	–	–	–		Grid search		
Fan et al., 2019	ELM	Several (3)	X	–	–	–		Random search		
Balkin & Ord, 2000	ANN, AR, RW	Economics	X	–	–	–		FS		
Martínez, Charte, et al., 2019	kNN	Economics	X	–	–	–		FS		
Lowther et al., 2020	sARIMA	Human access	–	–	–	X		MIQP		
Son & Kim, 2015	SVR	Energy	–	–	–	X		PSO		
Donate et al., 2013; Donate & Cortez, 2014	MLP	Several (5)	X	–	–	–		EDA, DEA, GA		
Panigrahi & Behera, 2020	MLP	Several (5)	X	–	–	–			DEA	
Valente & Maldonado, 2020	SVR	Energy	X	–	–	X			FS	
Maldonado et al., 2019	SVR	Energy	X	–	–	X			BE	
Rätz et al., 2019	GBM, LASSO, MLP, SVR, RF	Energy	X	–	X	X	Low variance	BO		
Widodo et al., 2016	MKL	Several (5)	X	–	–	–	ACF		Kernel comb.	
Kourentzes & Crone, 2010	MLP	Human access	X	X	–	–	INF			
Sagaert et al., 2018	LASSO	Economics	X	–	–	X		FS	Shrinkage	
Dellino et al., 2018b	sARIMA	Economics	–	–	–	X				PCA

Abbreviations: comb., combination; cyc., cyclical; exo., exogenous; tran, transformation.

explanatory variables. In the following, time series features are introduced, and their usage in automated forecasting pipelines is explored with literature references.

5.1.1 | Lag features

In autocorrelated time series, past observations can be valuable explanatory variables to forecast the target variable. Lag features provide values from prior time points for the forecasting method, that is, at time point k, the model also processes values that date back a certain time horizon H_1. Lag features are useful if the target variable has inertia that is significantly reflected in the resolution of the time series or if exogenous influences affect the target variable in periodic patterns (Rätz et al., 2019).

Table 4 reveals that lag features are the most applied type of features.

5.1.2 | Cyclic features

In the training data, the time stamps (e.g., [YYYY-MM-DD hh:mm:ss]) of the target variable are unique and specify the sequence of the observations. Cyclical patterns that humans can detect in this format, like the hour of the day, the day of the week, weekday and weekend, or month and year, cannot be processed by machine learning methods without encoding. Cyclic features provide this cyclic relationship by ordinal, interval, or categorical encoding. In the following, we exemplify these encodings for the day of the week (Table 5). Ordinal encoding assigns an integer numerical value to individual days. For an interval encoding, one may utilize a periodical sine-cosine encoding to establish similarities between related observations, for example, for encoding the day of the week, one adds two-time series features that encode each weekday.[17] A categorical encoding is achieved through so-called one-hot encoding. In this example, we create a time series feature for each day of the week, being one on that day and zero otherwise. Since the last category—the Sunday—is already implicitly represented if each other categorical feature is zero, one may omit an explicit feature.

5.1.3 | Transformation features

Time series transformations are widely used to make time series stationary, and can also be applied to extract explanatory variables that we term transformation features. These features include calculating time derivatives, the decomposition into trend, seasonality and residual, moving averages (Cerqueira et al., 2021), as well as applying mathematical operations on existing features, for example, multiplication of exogenous features (Rätz et al., 2019).

In the literature reviewed, cyclic features are not widely applied. Züfle and Kounev (2020) apply RF and GBM methods, using lag features and the cyclic features hour of the day, day of the week, and the exogenous feature holiday. However, the encoding of the cyclic features is not described. Sin–cos encoded cyclic features are used by Kourentzes and Crone (2010) as input for a multilayer perceptron (MLP) forecasting method.

TABLE 5 Exemplification of cyclical encoding methods for the day of the week of a time series

	Ordinal	Sin–cos interval		One-hot categorical					
Day of week	x_{dow}	x_{sin}	x_{cos}	x_{mon}	x_{tue}	x_{wed}	x_{thu}	x_{fri}	x_{sat}
Monday	0	1.000	0.000	1	0	0	0	0	0
Tuesday	1	0.623	0.782	0	1	0	0	0	0
Wednesday	2	−0.223	0.975	0	0	1	0	0	0
Thursday	3	−0.901	0.434	0	0	0	1	0	0
Friday	4	−0.901	−0.434	0	0	0	0	1	0
Saturday	5	−0.223	−0.975	0	0	0	0	0	1
Sunday	6	0.623	−0.782	0	0	0	0	0	0

5.1.4 | Exogenous features

In addition to features that are endogenously derived from the target variable, the forecast can be improved by using exogenous features if the target variable is subject to exogenous influences. In addition, lag, cyclical, and transformation features can also be extracted from exogenous features. Whether and which exogenous influences exist depends on the data domain.[18]

For example, energy data often depends on exogenous weather measures (Maldonado et al., 2019; Rätz et al., 2019; Son & Kim, 2015; Valente & Maldonado, 2020), human access data underlies the influences of public holidays and weather (Lowther et al., 2020; Züfle & Kounev, 2020), and sales data often correlates with economic indicators (Dellino et al., 2018a, 2018b; Sagaert et al., 2018).[19]

5.2 | Feature selection

After automatically extracting several features, they typically undergo a selection to remove features that provide no additional information value, for example, because of redundancy. In the following, methods for automated feature selection are presented, and their application in forecasting pipelines is explored based on the reviewed literature.

5.2.1 | Filter methods

Filter methods use metrics to rank single features or feature combinations and automatically select a promising feature set based on a threshold (Jović et al., 2015). Hence, the filters rely on the general characteristics of the training data and are independent of the forecasting method and other subsequent sections in the forecasting pipeline.

The characteristics used for filtering are manifold. Chakrabarti and Faloutsos (2002) propose an automated filtering method to determine the optimal lag features based on a threshold of the time series' fractal dimension. Martínez, Charte, et al. (2019) automatically select features by identifying significant lags in the PACF. Kourentzes and Crone (2010) propose the INF method to identify seasonal frequencies in time series, which automatically selects lag and sin–cos encoded cyclic features. The method of Bauer, Züfle, Herbst, et al. (2020) automatically filters frequencies of the time series using the periodogram with the threshold of a spectral value greater than 50% of the most dominant frequency. Additionally, cyclic features are extracted by calculating Fourier terms of the dominant frequencies. The most dominant frequency is used to decompose the time series into trend, season, and residual components by STL. Finally, the cyclic features and the seasonal component as a transformation feature are used to forecast the de-trended time series.

5.2.2 | Wrapper methods

In contrast to filter methods, the wrapper methods assess candidate features based on an evaluation criterion on a validation data set. The best-performing feature set is selected by a search method, which tailors the feature set to the forecasting method or the entire forecasting pipeline, respectively. The search methods and forecast accuracy metrics used in the literature are diverse. Independent from the chosen search method and metric, wrapper methods commonly take more computing time than filter methods, as training and validation require considerably more computing effort than calculating statistical measures. However, empirical studies show that the computing effort pays off in a higher achievable forecast accuracy when using the wrapper approach compared to filter methods (Jović et al., 2015).

In the literature reviewed, automated feature selection based on the forecast accuracy of the pipeline is addressed by different methods. The method of Yan (2012) and Fan et al. (2019) automatically determines the optimal lag features based on the forecast error by searching over candidate lags. The candidate lags are either predefined individually (grid search) or drawn randomly between specified boundaries (random search). Balkin and Ord (2000) and Martínez, Frías, et al. (2019) apply forward selection (FS) to automatically identify the optimal lag features. First, several forecasting pipelines are trained for each individual lag feature. Second, the FS selects the best-performing lag feature and repeats the first procedure by adding another lag feature. It retains the combination with the highest improvement and repeats the procedure until the forecast accuracy stops increasing. Besides search heuristics, optimization can be applied for feature selection. For automatically selecting exogenous features, Lowther et al. (2020) adopt a mixed integer quadratic

programming (MIQP) problem (Bertsimas et al., 2016), and Son and Kim (2015) apply the evolutionary particle swarm optimization (PSO). Evolutionary optimization for selecting features is also applied by Aladag et al. (2014). The authors use a genetic algorithm (GA) for automated lag feature selection of an FTS method and simultaneously determine the optimal FTS order. Besides the GA, Donate et al. (2013) and Donate and Cortez (2014) evaluate two other evolutionary optimization approaches for automated lag feature selection of MLPs, where the estimation distribution algorithm (EDA) yields the best convergence speed and the lowest forecast error.

5.2.3 | Embedded methods

Embedded methods integrate the feature selection into the training process of the forecasting method (Jović et al., 2015).[20] During the training, the embedded feature selection commonly estimates the feature importance and weights the features accordingly. Embedded methods require less computing effort than wrapper methods but more than filter methods (Rätz et al., 2019).

The embedded methods in the literature are specific to the forecasting method. The approach of Panigrahi and Behera (2020) automatically determines the optimal lag features by a differential evolution algorithm (DEA). Instead of using gradient-based methods for training an MLP, the authors integrate the weight estimation into the DEA, aiming to increase the convergence speed. Valente and Maldonado (2020) consider the automated selection of lag and exogenous features by an FS embedded into the SVR training process. The FS is based on a contribution metric that takes into account lags whose inclusion minimizes the metric. An automated backward elimination (BE) for SVR using embedded kernel penalization is described by Maldonado et al. (2019). Lag and exogenous features that are irrelevant for the forecast accuracy are successively removed during training. Another BE is proposed by Eğrioğlu and Bas (2022) that successively discards insignificant lag features of the single multiplicative neuron model (SMNM) forecasting method. The input significance test (Mohammadi, 2018) first calculates the outputs for each input, having the other inputs fixed in their arithmetic means, second, applies an LR with the outputs as independent variables and the target values as the dependent variables, and third, tests the LR parameters for significance. Insignificant inputs are excluded and the SMNM is trained again, testing inputs until they all are significant and simultaneously optimizing the number of the SMNM inputs. A lag feature selection, also based on testing the inputs for significance, is proposed for the bootstrapped hybrid ANN (Eğrioğlu & Fildes, 2022), which is an improvement of the method proposed by Yolcu et al. (2021). The values of the input weights vector are tested if they are significantly different from zero using the *t*-test or the sign-rank-test if the normality assumption is violated.

5.2.4 | Hybrid methods

Hybrid methods aim to combine the advantages of the above methods. Rätz et al. (2019) evaluate several filter, wrapper, embedded, and hybrid feature selection methods. Based on their experiments, they propose to use a filter method to automatically remove all features with low variance in the first step. Afterward, they apply Bayesian optimization (BO) to assess the remaining feature candidate combinations, consisting of lag, transformation, and exogenous features based on the forecast accuracy. The approach of Widodo et al. (2016) filters significant lags using ACF, associates the remaining lag features with a kernel, and automatically assigns appropriate weights to the kernels during training of the multiple kernel learning (MKL) method. Sagaert et al. (2018) first filter candidate lag features using automated FS, similar to the stepwise search for the automated ARIMA design of Hyndman and Khandakar (2008). Then, the identified set of lag features together with exogenous features are used as inputs of the embedded least absolute shrinkage and selection operator (LASSO) method that automatically selects features by shrinking the coefficients of irrelevant ones

5.3 | Feature aggregation

Feature aggregation aims to transform the feature space into a low-dimensional representation while retaining the primary properties of the time series. The transformation is advantageous in high-dimensional feature spaces to reduce processing time and avoid the curse of dimensionality.[21] The principal component analysis (PCA), for

example, maps the data to a lower-dimensional space, maximizing the variance in the lower-dimensional representation.[22]

Dellino et al. (2018a) prewhiten the time series data with a PCA, that is, the PCA aggregates exogenous features and discards the principal components with a low variance that are assumed as noise. Instead of aggregating exogenous features, Gao et al. (2021) apply the PCA to aggregate lag features for an FTS. They aggregate the lag features into one feature—the first principle component—aiming to maximize the dependence between the target variable and lag features, allowing a large number of feature candidates to be considered without explicit selection.

5.4 | Discussion

We discuss feature engineering as the second section of the automated forecasting pipeline, highlight a potential issue, provide recommendations, and suggest future work.

As shown in Table 4, most automated pipelines that apply forecasting methods based on machine learning rely on lag features because—unlike statistical forecasting methods—they do not consider time lags implicitly.[23] The sparse use of cyclic and transformation features can be explained because most of the analyzed forecasting pipelines already consider this information by automatically selecting appropriate lags.

In terms of automation, the majority of the literature combines the extraction of predefined features with an automated selection. Regarding feature extraction, the primary concerns remain in the human-defined feature extraction since it requires experience and may be biased. For this reason, extracting a large set of default features—including lag, cyclical, and transformation features—for the automated forecasting pipeline can be valuable since the subsequent automated feature selection removes irrelevant ones. Additionally, exogenous features are a powerful opportunity to integrate domain-specific knowledge into the forecasts.[24]

In the automated feature selection, all four methods—filter, wrapper, embedded, and hybrid—are useful for removing irrelevant features. While an embedded method is computationally beneficial by integrating the feature selection in the training process, it is specifically designed for a forecasting method. Because of its independence from the forecasting method, we recommend combining a filter with a wrapper method (hybrid) to initially reduce the candidate features through filtering and then tailoring the feature selection to the forecasting method or pipeline, respectively.

Since aggregated features often correlate with other features and the number of features can be reduced by an automated feature selection method, we do not consider automated feature aggregation necessary to reach a high forecast accuracy. Moreover, the aggregation of features limits their interpretability regarding their information value for the forecast.

Based on these challenges, future work toward automated forecasting pipelines should consider the automated extraction of default endogenous features,[25] extended by a domain-specific extraction to include exogenous features.

6 | HYPERPARAMETER OPTIMIZATION

Forecasting methods incorporate a wide range of hyperparameters about the forecasting model's structure, training regularization, and algorithm setup; parameters whose values are not directly derived from the data and must be selected by the data scientist. The hyperparameter configuration $\boldsymbol{\lambda}$ includes all considered hyperparameters and their selected values (Feurer & Hutter, 2019). By tailoring $\boldsymbol{\lambda}$ to the specific problem using HPO, one may improve the forecast accuracy over the default setting of common forecasting libraries (Rätz et al., 2019). Besides HPO, the selection of the best forecasting method can improve the forecast accuracy. Since we want to generate a pool of candidates for the selection whose hyperparameters are, in turn, optimally configured, we first address the HPO.

6.1 | Evaluation criteria and validation sample

Most HPO methods assume that the performance of the model is quantifiable. Typically, the hyperparameters are optimized to improve the model's forecast accuracy and measures from information theory and error measures are used as

evaluation criteria.[26] Information criteria (IC) measure the amount of information lost by a statistical model, taking into account the goodness-of-fit (GOF) and the model complexity. The less information a model loses, the higher the expected forecast accuracy. One IC is the Akaike information criterion (AIC)

$$\mathrm{AIC} = 2w - 2\ln\left(\widehat{L}\right) \tag{4}$$

with the number of estimated parameters w and the model's maximum value of the likelihood function $\widehat{L}$. It rewards GOF but penalizes high numbers of model parameters. The penalty is required since adding more parameters may increase the likelihood without being justified by the data (overfitting). Another popular IC is the Bayesian information criterion (BIC)

$$\mathrm{BIC} = w\ln(K) - 2\ln\left(\widehat{L}\right) \tag{5}$$

that additionally considers the number of data points K. Popular error measures are the MSE

$$\mathrm{MSE} = \frac{1}{N}\sum_{n=1}^{N}(y[n] - \widehat{y}[n])^2, \tag{6}$$

and the MAE

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N}(y[n] - \widehat{y}[n]), \tag{7}$$

where $\widehat{y}_i$ is the forecast and y_i the realized value. The smaller the error of the model, the higher the forecast accuracy. There are further error measures that are derived from the MSE and the MAE, such as the root MSE (RMSE), the mean absolute scaled error (MASE), the mean absolute percentage error (MAPE), and the symmetric MAPE (sMAPE). For their definitions, we refer to Hyndman and Athanasopoulos (2021). A detailed comparison of error measures is given by Hyndman and Koehler (2006), recommending the MASE for comparing the forecast accuracy across multiple time series. For evaluating time series with similar characteristics, empirical evaluations indicate that the choice of the error measure does not significantly influence the evaluated candidates' rank (Koutsandreas et al., 2021).

Error measures can be calculated *in-sample* or *out-of-sample*. In-sample means that the forecast error is determined on data points that are part of the training data sample, while out-of-sample uses unseen points from the validation data sample. The out-of-sample error validation is generally considered to be a more trustworthy empirical evidence, as in-sample error validation is prone to overfitting.[27]

To increase the robustness, cross-validation can be performed by partitioning the data several times into different training and validation subsets and averaging the validation error across the folds (Figure 4). The random sample cross-validation is a valid choice if the forecasting method does not assume a sequential dependency of the data. Sequential dependency is maintained by the blocked sample cross-validation, which partially uses data for training that is in the future of the test data. Using future data for training is avoided with the fixed and rolling origin cross-validation. In contrast to the fixed origin cross-validation, the rolling origin update cross-validation uses past values as inputs for the forecasting model as they become available but does not retrain the model as with the rolling origin retrain validation and rolling window cross-validation variants. Table 6 shows which cross-validation variants are used in the reviewed literature to evaluate hyperparameter configurations.

Note, if the forecasting method to be evaluated contains a horizon of past values H_1 (e.g., lag features), these values must be provided during testing, either by using the initial values of the test data set and omitting them for the evaluation or by taking them from the training data set. For the latter, empirical evidence exists that the resulting intersection of training and test data has no impact on the evaluation quality (Bergmeir et al., 2018; Bergmeir & Benítez, 2012).

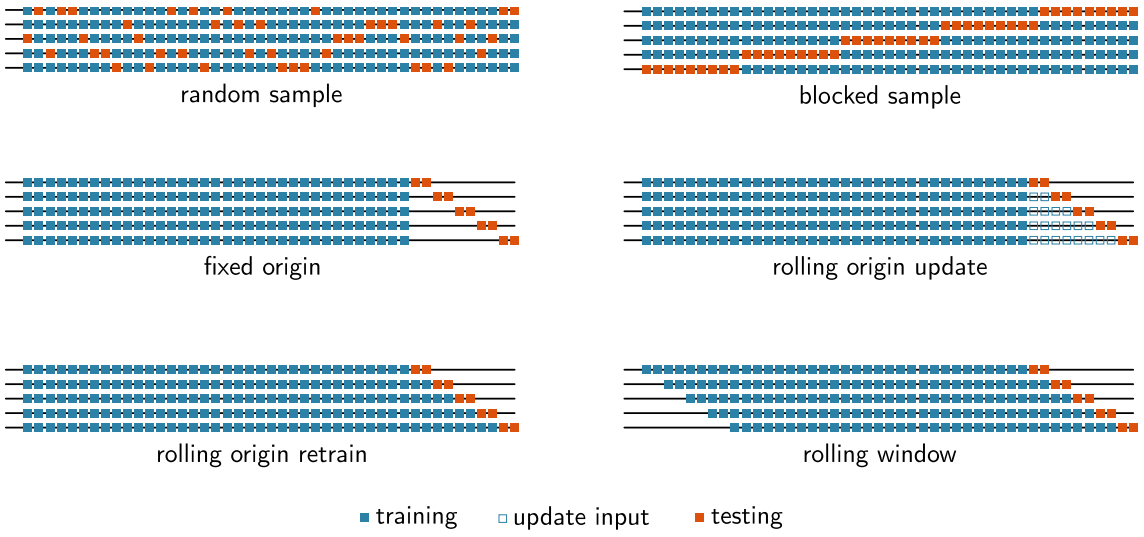

FIGURE 4 Common cross-validation variants for time series forecasting based on Bergmeir (2013)

6.2 | Optimization methods

Given validation data and metrics, the hyperparameter configuration of forecasting methods can be optimized using different optimization methods. Table 6 shows the summary of HPO methods for time series forecasting. In the following subsections, different automated HPO methods are introduced, and their application in forecasting pipelines is exemplified using literature references.

6.2.1 | Grid search and the random search

The most elementary method for HPO is the exhaustive *grid search*, where the data scientist defines a finite set of $\pi = 1, ..., \Pi$ hyperparameter values to be evaluated, resulting in a full factorial configuration space $\mathbf{\Lambda} \in \mathbb{R}^{\Pi}$. As the grid search evaluates the Cartesian product of these sets, the number of computations B grows exponentially with the dimensionality of $\mathbb{R}^{\Pi}$. Hence, increasing the discretization resolution increases the computing effort substantially (Feurer & Hutter, 2019). The *random search* (Bergstra & Bengio, 2012) is an alternative to the grid search. It irregularly samples the hyperparameter set until a certain number of computations B is exhausted. The random search may perform better than the grid search if some hyperparameters are much more important than others.[28]

Figure 5 shows a comparison of both methods with two hyperparameters and an equal number of computations B.

Hyndman et al. (2002) generalize the formulation of ES forecasting methods with the error trend seasonality (ETS) method and suggest a grid search that automatically selects the hyperparameter configuration with the lowest in-sample AIC. Utilizing grid search for HPO is also applied to statistical forecasting methods based on AR and moving averages (MAs). Sekma et al. (2016) optimize the hyperparameters of an AR (p,s) and a vector autoregression (VAR) (p,s) method, respectively, using grid search. The hyperparameter configuration resulting in the minimal BIC, calculated in-sample, is automatically selected. A similar HPO method for MA (q) methods is proposed by Svetunkov and Petropoulos (2018), where the data scientist needs to define the in-sample optimization criterion. An advanced method, combining preprocessing and HPO of ARIMA (p,d,q) methods is proposed by Alzyout et al. (2019). First, a stationarity test determines the differencing order d. Second, the maximal AR-lag order $p_{\max}$ and the maximal MA-lag order $q_{\max}$ are determined by automatically evaluating the PACF and the ACF, respectively. Finally, a grid search from 0 to $p_{\max}$ and 0 to $q_{\max}$ selects the optimal hyperparameter configuration based on in-sample AIC or BIC. Combining preprocessing and HPO is also proposed by Hyndman and Khandakar (2008). First, the authors propose to automatically determine the differencing and seasonal differencing order d and D and the seasonal period s with a stationarity test. Second, instead of an exhaustive grid search over the hyperparameters p,q,P, and Q of the sARIMA $(p,d,q)(P,D,Q)_s$, they propose a two-stage grid search, reducing the number of evaluations. In the first stage, four candidate hyperparameter configurations are evaluated, whose hyperparameters depend on the previously determined s. The

TABLE 6 Summary of hyperparameter optimization (HPO) methods in automated time series forecasting pipelines

Reference	Optimized forecasting method(s)	Optimized hyper-parameters	Evaluation criterion	Validation sample			Optimization method
				In-sample	Out-of-sample	Cross-validation	
Tran & Reed, 2004	sARIMA	p,q		–	–	–	ACF&PACF
Amin et al., 2012	ARIMA, SETARMA	p,q		–	–	–	Testing, ACF&PACF
Sekma et al., 2016	AR, VAR	p,s	BIC	X	–	–	Grid search, checking
Svetunkov & Petropoulos, 2018	MA	q	User-defined	X	–	–	Grid search
Alzyout et al., 2019	ARIMA	p,d,q	User-defined	X	–	–	Testing, ACF&PACF
Hwang et al., 2012	ARIMA	p,d,q	Multiple (5)	X	–	–	Grid search, checking
Hyndman & Khandakar, 2008	sARIMA	p,d,q,P,D,Q,s	AIC	X	–	–	Testing, two-stage grid search
Arlt & Trcka, 2021	sARIMA	p,d,q,P,D,Q	User-defined	X	X	None or rolling origin update	Testing, grid search, checking
Pedregal, 2019	sARIMA	p,d,q,P,D,Q,s	AIC	X	–	–	var. minimization, two-stage grid search
Lowther et al., 2020	sARIMA	p,d,q,P,D,Q,s	User-defined	X	X	Used but not described	Two-stage MIQP, grid search
Dellino et al., 2018b	sARIMA	p,d,q,P,D,Q	MAE	–	X	Rolling window	BO GP
Hyndman et al., 2002	ETS	E,T,S	AIC	X	–	–	Grid search
Bermúdez et al., 2012	TES	α,β,γ,φ,p	Multiple (3)	X	–	–	Multi-objective NLP
Spiliotis et al., 2020	Theta	φ,T,S	MAE	X	–	–	Brent, testing, grid search
Villegas & Pedregal, 2019	UC	T,S,I	BIC	X	–	–	Grid search
Chakrabarti & Faloutsos, 2002	kNN	k		–	–	–	Heuristic

TABLE 6 (Continued)

Reference	Optimized forecasting method(s)	Optimized hyper-parameters	Evaluation criterion	Validation sample			Optimization method
				In-sample	Out-of-sample	Cross-validation	
Sagaert et al., 2018	LASSO	λ	MAPE	–	X	Used but not described	Grid search
Son & Kim, 2015	SVR	C,γ	RMSE	–	X	–	Grid search
Maldonado et al., 2019; Valente & Maldonado, 2020	SVR	C,γ	MAPE	–	X	Rolling origin retrain	Grid search
Widodo & Budi, 2013; Widodo et al., 2016	MKL	C,ε, kernel,γ	sMAPE	–	X	Used but not described	Grid search, embedded
Gao et al., 2021	FTS	l,m,n, Gaussian kernel radius	Multiple (2)	–	X	Used but not described	Grid search
Eğrioğlu & Bas, 2022	hybrid ANN	N_i,N_h	RMSE	–	X	–	Grid search
Yan, 2012	GRNN	Spread		–	–	–	Heuristic
Panigrahi & Behera, 2020	MLP	N_i,N_h	RMSE	X	X	–	DEA
Donate et al., 2013	MLP	N_i,N_h,α,s	MSE	–	X	–	GA, DEA, EDA
Donate & Cortez, 2014	MLP	N_i,N_h,α,Δ_{max}	MSE	–	X	–	EDA
Silva et al., 2020	FTS	k,α,μ	Multiple (2)		X	Rolling window	GA
Fan et al., 2019	ELM	N_i,N_h	sMAPE	–	X	–	Random search
Bandara et al., 2020	LSTM	N_h, epoch-size, batch-size,α, epochs, noise, L2	MASE	–	X	–	BO GP
Sergio et al., 2016	MLP, DBN, SVR	Method-dependent	MSE	–	X	Used but not described	PSO
Rätz et al., 2019	GBM, LASSO, MLP, RF, SVR	Method-dependent	MAE	–	X	Blocked sample	BO TPE

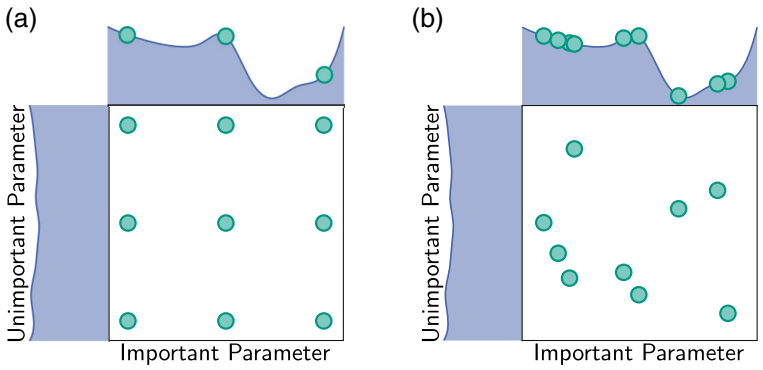

FIGURE 5 Comparison of a grid search and a random search for minimizing a function with one important and one unimportant parameter (Bergstra & Bengio, 2012; Feurer & Hutter, 2019)

hyperparameter configuration with the smallest in-sample AIC value proceeds to the second stage, where p,q,P, and Q are varied ± 1. If a configuration with a lower AIC value is found, it becomes the active configuration of stage two and the variation is repeated until the AIC stops improving. Pedregal (2019) adopt this method except for the preprocessing step. Instead of stationarity tests, the variance of the time series is minimized to identify the differencing orders d and D.

Grid search is also applied for HPO of forecasting methods based on machine learning. Sagaert et al. (2018) determine the optimal shrinkage factor $\boldsymbol{\lambda}$ of the LASSO method for embedded feature selection in terms of the mean MAPE of a 10-fold cross-validation. Several authors apply a grid search to determine the optimal hyperparameter configuration of an SVR. Son and Kim (2015) optimize the hyperparameters C and γ based on the RMSE without cross-validation, Maldonado et al. (2019) and Valente and Maldonado (2020) based on the MAPE using the rolling origin retrain cross-validation. The grid search used by Widodo & Budi, 2013, Widodo et al., 2016 is based on a 5-fold cross-validation, using the mean sMAPE value across folds as the forecast error measure. They optimize the hyperparameters C and ε of the SVRs used in a MKL approach with embedded feature selection. Combining automated feature selection and HPO is also suggested by Fan et al. (2019). The authors identify the optimal lag features using random search and link the number of hidden neurons N_h to the number of input neurons N_i. Grid search is also applied to forecasting methods that are inspired by biology. Gao et al. (2021) optimize the FTS order l, the number of fuzzy sets for the input n and target m, and the Gaussian kernel radius. As evaluation criterion, they use the MASE combined with a penalty measure for high FTS complexities. Eğrioğlu and Fildes (2022) determine the best architecture of a hybrid ANN, incorporating linear and nonlinear components, in terms of the input and hidden neurons N_i and N_h.

6.2.2 | Bayesian optimization

Rather than evaluating a finite search grid, BO explores and exploits the configuration space $\boldsymbol{\Lambda} = \Lambda_1 \times \Lambda_2, ..., \Lambda_H$ of H hyperparameters Λ. BO uses a probabilistic surrogate model to approximate the objective function Q that maps the forecast accuracy Q on $\boldsymbol{\Lambda}$. More specifically, an observation $Q(\boldsymbol{\lambda})$ of the objective function reflects the forecast accuracy with a particular hyperparameter configuration $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ of the forecasting methods used. In each iteration, the optimization updates the surrogate model with the new observation and uses an acquisition function to decide on the next hyperparameter configuration $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ to be explored (Figure 6). The acquisition function trades off the exploration against the exploitation of $\boldsymbol{\Lambda}$ by determining the expected benefit of different hyperparameter configurations using the probabilistic distribution of the surrogate model (Feurer & Hutter, 2019).

For surrogate modeling, various approaches exist, ranging from Gaussian Processes (GPs) and their modifications to machine learning approaches, for example, RFs or tree-structured Parzen estimators (TPEs). Feurer and Hutter et al. (2019) recommend using a GP-based BO for configuration spaces with real-valued hyperparameters and computationally expensive training, and an RF or TPE-based BO for configuration spaces with categorical hyperparameters and conditions, for example, the choice of a forecasting method and its conditional (sub-)configuration space.

Dellino et al. (2018a) apply a BO based on GP surrogate modeling to optimize the hyperparameters of the sARIMA $(p,d,q)(P,D,Q)$ using the rolling window cross-validation and the MAE. They compare the BO to an exhaustive grid

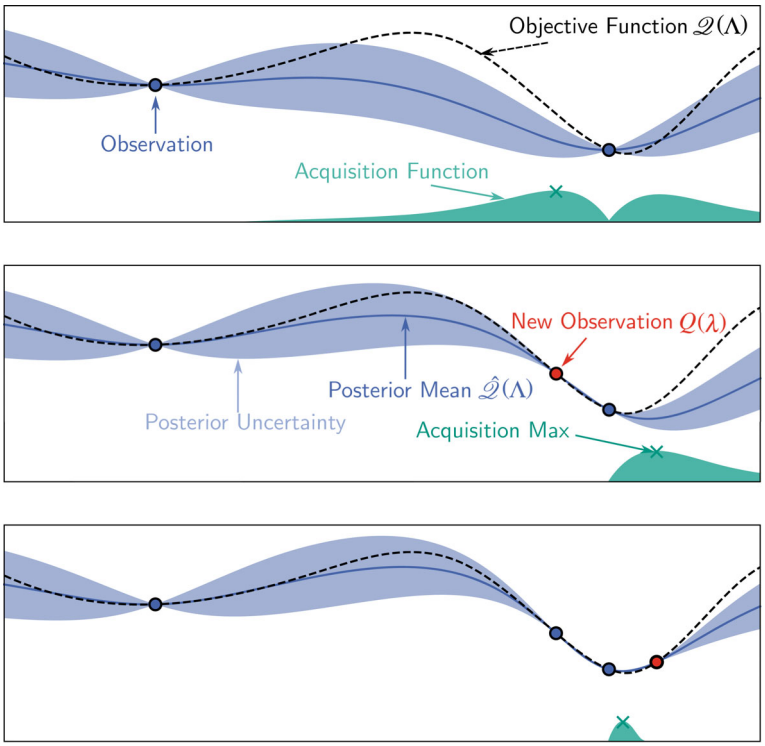

FIGURE 6 Two exemplary iterations of a Bayesian optimization (BO) on a 1D function. The BO minimizes the predicted objective function (blue line) by maximizing the acquisition function (green surface). The acquisition value is high where the value of the predicted objective function is low, and its predictive uncertainty (light blue interval) is high (Feurer & Hutter, 2019). The true objective function (dashed line) might lie outside of the predicted uncertainty interval

search, where the BO achieves lower forecast errors but requires more computing time. In their experiment, however, the comparison is unequal as the configuration space of the BO is larger, and the best-performing model of the BO results in a hyperparameter configuration that the grid search does not evaluate.

Bandara et al. (2020) use a BO with a GP surrogate model to optimize the recurrent neural architecture of a long short-term memory (LSTM) and the hyperparameters of the training method. The forecast accuracy of each hyperparameter configuration is evaluated out-of-sample based on the MASE. Rätz et al. (2019) optimize multiple forecasting methods, combining feature selection, HPO, and forecasting method selection using a BO based on a TPE surrogate model. The forecast accuracy of hyperparameter configurations is estimated using the mean MAE of a 5-fold blocked sample cross-validation.

6.2.3 | Nonlinear programming

Mathematical programming can be applied for HPO if calculating the evaluation criterion is solvable in closed form. Nonlinear programming (NLP) solves an optimization problem, where at least one of the objective functions or constraints is a nonlinear function of the decision variables. The objective function may be convex or nonconvex, where nonconvex NLP incorporates multiple feasible regions and multiple locally optimal solutions within them (Bazaraa et al., 2006). Depending on the formulation of the objective function and its constraints, different solving methods are appropriate.

Bermúdez et al. (2012) apply the generalized reduced gradient method to solve a multi-objective NLP. They jointly minimize the in-sample RMSE, MAPE, and MAD to determine the smoothing parameters $\alpha,\beta,\gamma,\varphi$ and the number of periods of the seasonal cycle p of the TES method. Lowther et al. (2020) use MIQP to select suitable exogenous features for ARIMA. They combine this selection with a grid search over the sparsity parameter k of the MIQP formulation and the sARIMA hyperparameters p,d,q,P,D,Q.

6.2.4 | Heuristics

A heuristic is an informed search technique that systematically explores a configuration space $\mathbf{\Lambda}$ subject to a constant search rule (Pearl, 1984).

Tran and Reed (2004) propose heuristics based on the ACF and PACF to determine the AR and MA lag order p and q. A similar approach is published by Amin et al. (2012). To determine the transformation parameter θ of the Theta forecasting method, Spiliotis et al. (2020) apply the Brent–Dekker method—a root-finding method. They determine the optimal θ for eight different trend T and season S configurations, and select the configuration that minimizes the in-sample MAE.

Chakrabarti and Faloutsos (2002) propose a heuristic to specify the number of nearest neighbors k of the k-nearest neighbors (kNN) forecasting method after selecting the optimal lag features. A training sample-related heuristic for determining the spread factor of the general regression neural network (GRNN) is introduced by Yan (2012).

6.2.5 | Metaheuristics

Metaheuristics are strategies for guiding a search according to feedback from the objective function, previous decisions, and prior performance (Stützle, 1999), that is, the searching behavior changes while exploring the configuration space $\mathbf{\Lambda}$. Metaheuristics do not require assumptions about the objective function and can solve optimization problems where gradient-based methods fail.

Evolutionary optimization

Evolutionary algorithms (EAs) comprise a wide range of population-based metaheuristics inspired by biological evolution (Bäck, 1996). A population of candidate hyperparameter configurations is evaluated using a fitness function to determine the performance of solutions. Weak solutions drop out, while well-performing solutions evolve. The mechanisms of selection and evolution differ between algorithms.

GAs evolve a population of candidate hyperparameter configurations to explore and exploit the configuration space $\mathbf{\Lambda}$. The hyperparameters of a candidate solution are encoded as genes in a chromosome. In each generation, the fitness of the population is evaluated, and the chromosomes of individual candidates are modified to create a new generation—the offspring. The modification includes recombination and mutation and depends on an individual candidate's fitness. A part of the population is retained and forms with the offspring the next generation. DEAs differ from GAs in the mechanism of generating the offspring. While in GAs an individual acts as parent to generate an offspring, the DEA adds the weighted difference between two chromosomes to create a new individual. In this way, no separate probability distribution is required, making the algorithm self-organizing. In EDAs, the population is replaced by a probability distribution over the choices available at each position in the chromosome of the individuals. A new generation is obtained by sampling this distribution, avoiding premature convergence and making the representation of the population more compact.

Donate et al. (2013) evaluate a GA, a DEA, and an EDA for optimizing the hyperparameter configuration of an MLP, including the number of input and hidden neurons N_i and N_h, as well as the training method's hyperparameters learning rate α and the weight initialization seed s. The results of the experiments show that the DEA and the EDA require more than 100 generations to improve significantly over GA. After 200 generations, the EDA achieves the lowest forecast error, followed by the DEA and the GA. In a later publication (Donate & Cortez, 2014), the authors adapt the chromosome encoding and replace s with the parameter Δ_max of the used training algorithm. In both publications, the fitness of each individual is evaluated by calculating the MSE on an out-of-sample validation data set. Panigrahi and Behera (2020) apply a DEA to optimize N_i and N_h of an MLP, combining in-sample and out-of-sample validations. The fitness of each individual (RMSE) is calculated in-sample, and the DEA is terminated when the RMSE on the validation data set increases,[29] indicating the beginning of overfitting. Instead of optimizing a single fitness function using GA, Silva et al. (2020) optimize two fitness functions with the rolling window cross-validation. They simultaneously minimize the forecast error and model complexity of the FTS forecasting method to find the trade-off between these conflicting criteria. Thereby, the authors use the parallelizability of the GA to evaluate the number of partitions k, the partitioning method α, and the selection of the membership function μ on multiple computing cluster nodes.

Particle swarm optimization

In PSO, a population of hyperparameter candidate configurations—the swarm—is evaluated. The candidates move through the configuration space $\mathbf{\Lambda}$, where the movement of the swarm is guided by the best-performing candidates so far.

Sergio et al. (2016) apply PSO to optimize the hyperparameter configuration of multiple forecasting methods, combining the best hyperparameter configurations afterward to an ensemble.

6.3 | Diagnostic checking

Diagnostic checking evaluates the fitted forecasting model against criteria that indicate an adequate forecasting method configuration. An adequate forecasting method configuration yields residuals $r[k] = y[k] - \widehat{y}[k]$ with properties of white noise, that is, the residuals are not autocorrelated having zero mean and finite variance (Hyndman & Athanasopoulos, 2021). For statistical forecasting methods, the relationship between dependent and independent variables should be statistically significant. That is, the parameters estimated in the fitting process describing this relationship are significantly different from zero. If the forecasting method makes assumptions about the time series characteristics (e.g., stationarity), it is advisable to check whether the assumptions hold (Hyndman & Athanasopoulos, 2021).

Regarding statistical forecasting methods, Amin et al. (2012) check the randomness of the residuals with a Box–Pierce test and the significance of the parameters of the ARIMA or self-exciting threshold autoregressive moving average (SETARMA) models with a *t*-test. Furthermore, they test the invertibility and the stationarity, that is, both the sum of the AR parameters and the sum of the MA parameters have to be smaller than one. Analogously, Hwang et al. (2012) verify stationarity and invertibility. For residual diagnostics, they additionally check the residual's correlations with the Ljung–Box test. Similar to the above authors, Sekma et al. (2016) test the parameters' significance and check if the residuals are white noise with the Ljung–Box test (Ljung & Box, 1978).

In terms of machine learning-based forecasting methods, Eğrioğlu and Bas (2022) use the *F*-test to check if the SMNM is adequate. The null hypothesis to be discarded is that each parameter of the LR applied during input significance testing (described in Section 5.2) equals zero.

6.4 | Discussion

We discuss HPO as the third section of the automated forecasting pipeline, identify a possible problem, give recommendations, and suggest future research.

For HPO, most automated forecasting pipelines apply grid search as shown in Table 6. Directed search methods, for example, evolutionary optimization and BO, are, however, used for HPO of forecasting methods with high computational training complexity, primarily including machine learning methods. For evaluating the forecast accuracy during HPO, only 10 of the 31 references apply cross-validation to increase the robustness of the HPO, and often, the applied cross-validation procedure is insufficiently described. Also, diagnostic checking of the forecasting method configuration after the HPO is only applied sporadically and mostly for statistical forecasting methods.

With regard to automation, several authors link HPO and the preceding automated steps of preprocessing (Section 4) and feature engineering (Section 5). One potential problem is the optimization of the differencing orders d and D of the ARIMA (p,d,q) and sARIMA $(p,d,q)(P,D,Q)_s$ methods using IC metrics. The differencing transformation affects the likelihood in IC metrics, making the metrics between different values of d and D not comparable (Hyndman & Athanasopoulos, 2021). Therefore, d and D should be determined in the preprocessing section. For the subsequent HPO, we assume a straightforward grid search to be sufficient since the computational training complexity of ARIMA and sARIMA methods is rather low, and the configuration space is small. In contrast, for HPO of forecasting methods with high computational training complexity, for example, ANNs, and large categorical and conditional configuration spaces, we recommend a BO based on TPE.

Given these recommendations, for automated forecasting pipelines in future work, we suggest that the automated analysis of residuals is also integrated into the pipeline using machine learning-based forecasting methods.

TABLE 7 Summary of method selection and ensembling methods in automated time series forecasting pipelines

Reference	Forecasting method(s)	Forecasting method selection	Selection basis		Forecast Ensembling	
			Evaluation criteria	Meta-features	Pool	Ensemble
Martínez, Charte, et al., 2019	kNN		–	–	Use all	avg.
Tak et al., 2021	DES, TES, FTS-N, TS-FIS, MLP	Empirical	X	–	Use all	wgt.
Shcherbakov et al., 2013	ANN, MA, sRW	Heuristic	–	–		
Pawlikowski & Chorowska, 2020	autoARIMA, ETS, LR, RW, sRW, Theta	Heuristic	–	–	Use all	wgt.
Balkin & Ord, 2000	MLP, AR, RW	Empirical	X	–		
Amin et al., 2012	ARIMA, SETARMA	Empirical	X	–		
Sekma et al., 2016	AR, VAR	Empirical	X	–		
Pereira et al., 2018	ARIMA DES, LR, dRW, SES, TES	Empirical	X	–		
Züfle & Kounev, 2020	GBM, RF, Telescope	Empirical	X	–		
Rätz et al., 2019	GBM, LASSO, MLP, RF, SVR	Empirical	X	–		
Fildes & Petropoulos, 2015	autoARIMA, Damped, RW, sRW, SES, DES, TES, Theta	Empirical	X	–		
Crone & Kourentzes, 2010	MLP	Empirical	X	–	k-best	avg.
Kourentzes et al., 2019	ETS	Heuristic, empirical	X	–	k-best	avg.
Shetty & Shobha, 2016	autoARIMA, ETS, NNetAR, TBATS	Empirical	X	–	k-best	wgt.
Wu et al., 2020	ELM, ENN, FNN, GRNN, MLP, RBFNN	Empirical	X	–	k-best	wgt.
Sergio et al., 2016	DBN, MLP, SVR	Heuristic, empirical	X	–	dyn. Best	avg., wgt.
Taghiyeh et al., 2020	autoARIMA, MA, SES, DES, TES, Theta	Decision model	X	-		
Kück et al., 2016	SES, DES, TES	Decision model	X	X		
Widodo & Budi, 2013	autoARIMA, MKL, PR, SVR	Decision model	–	X		
Scholz-Reiter et al., 2014	MLP, autoARIMA, ETS, LC, LL, RW	Decision model	–	X		
Shahoud et al., 2020	DT, GBM, LR, RF	Decision model	–	X		
Cui et al., 2016	MLP, GP, MARS, PR, RBFNN, SVR	Decision model	–	X		
Baddour et al., 2018	autoARIMA, BSTS, ETS, GBM, Prophet, TBATS, Theta	Decision model	–	X		
Bauer, Züfle, Grohmann, et al., 2020	Telescope: Cubist, Evtree, GBM, NNetAR, RPaRT, SVR	Decision model	–	X		
Bandara et al., 2020	LSTM	Decision model	–	X		

TABLE 7 (Continued)

Reference	Forecasting method(s)	Forecasting method selection	Selection basis		Forecast Ensembling	
			Evaluation criteria	Meta-features	Pool	Ensemble
Montero-Manso et al., 2020	AR, autoARIMA, ETS, NNetAR, RW, dRW, sRW, TBATS, Theta	Decision model	–	X	Use all	wgt.
Li et al., 2020	autoARIMA, ETS, NNetAR, RW, dRW, sRW, STL-AR, TBATS, Theta	Decision model	–	X	Use all	wgt.
Ma & Fildes, 2021	autoARIMA, ELM, ETS, GBM, LR, RF, SVR	Decision model	–	X	Use all	wgt.
Züfle et al., 2019	autoARIMA, ETS, NNetAR, RW	Decision model	–	X	k-best	wgt.
Villegas et al., 2018	MA, IMA, White Noise	Decision model	X	X		

Abbreviations: avg., averaging; dyn., dynamic; wgt., weighting.

7 | FORECASTING METHOD SELECTION AND ENSEMBLING

Not only optimizing hyperparameters of a forecasting method but also selecting the appropriate method is crucial for the forecast accuracy. Consequently, the forecasting method selection is often combined with an individual HPO, that is, selecting the best forecasting method from a pool of candidates whose hyperparameters, in turn, are already optimized.[30] Forecast ensembling aims to bundle the forecasts of several methods, thereby reducing the impact of occasional poor forecasts—which can even occur with the best-selected and optimally configured forecasting method.

7.1 | Forecasting method selection

For automatically selecting the best-performing forecasting method, there are several approaches that we divide into heuristic, empirical, and decision model-based selection. The selection is based on experience, a determined evaluation criteria, or meta-features. The determined evaluation criteria reflect ICs or error measures. Meta-features describe properties of the time series to be forecast and provide meta-information, including statistical characteristics of the target time series, such as the skewness, kurtosis, and self-similarity; and domain information, such as physical properties of the system and environmental characteristics. In the following, methods for automated forecasting method selection are presented, and their application in forecasting pipelines is examined based on the reviewed literature, guided by the summary in Table 7.

7.1.1 | Heuristic forecasting method selection

The heuristic selection of the forecasting method relies on fixed rules. The basis of these rules is experience and statistical tests that examine the time series for certain characteristics. Therefore, heuristic selection requires neither determining evaluation criteria nor meta-features.

Shcherbakov et al. (2013) propose decision rules that consider the amount of available training data. For small amounts of training data (i.e., less than 672 observations), a naïve method is selected that uses the previous day's value as the forecast. For moderate amounts of training data (i.e., 672–2688 observations), a MA method is applied, whereas an ANN is selected for greater amounts of training data (i.e., more than 2688 observations). Besides the amount of training data, the availability of calendar information and exogenous time series also determines the chosen forecasting method.

7.1.2 | Empirical forecasting method selection

The empirical forecasting method selection determines the performance of several forecasting methods during training (in-sample) or on a validation data set (out-of-sample) given an evaluation criterion and automatically selects the best-performing forecasting method.[31] The determined performance scores, however, may be subject to statistical fluctuations, for example, in the case of cross-validation with random sampling. To address this problem, statistical significance tests can be applied, for example, the one proposed by Harvey et al. (1997). The test assumes that the performance scores originate from the same distribution and quantifies the likelihood that the scores are equal (null hypothesis). If the null hypothesis is rejected, the test suggests that the difference in the performance scores is statistically significant.

Balkin and Ord (2000) train an AR, an MLP, and a naïve RW method, and select the forecasting method with the smallest in-sample BIC. Similarly, Sekma et al. (2016) decide between AR and VAR forecasting methods based on the smallest in-sample BIC; Amin et al. (2012) use the AIC to choose between ARIMA and SETARMA. An out-of-sample validation is used by Pereira et al. (2018). They calculate the MAE of six forecasting methods on a validation data set and select the method with the lowest MAE. A similar selection strategy is used by Züfle and Kounev (2020). They select the best-performing candidate forecasting method in terms of the R^2 score on validation data. Robustness can be increased by cross-validation. Rätz et al. (2019) combine feature selection, HPO, and the selection of the optimal forecasting method using BO with the mean MAE over five folds.

The effectiveness of in-sample and out-of-sample empirical forecasting method selection is evaluated by Fildes and Petropoulos (2015). They assess the following four empirical selection methods: (1) minimal one-point-ahead in-sample MSE, (2) minimal one-point-ahead out-of-sample MAPE, (3) minimal h-point-ahead out-of-sample MAPE, and (4) minimal 1–18 points-ahead out-of-sample MAPE. The latter method proves to be better than the 1-point-ahead validation and even than adjusting the validation to the corresponding forecast horizon H.

7.1.3 | Decision model-based forecasting method selection

Instead of a heuristic or empirical forecasting method selection, one may train a decision model to automatically select the optimal forecasting method. As decision models, regression, classification, and clustering methods can be used to establish a relationship between evaluation criteria or meta-features and the optimal forecasting method.

In a decision model, the selection and aggregation of meta-features can improve decision accuracy. The principle of feature selection and aggregation methods corresponds to the descriptions in Section 5.2. Similar to the optimization of forecasts, the decision model can also be improved by HPO. Table 8 gives an overview of meta-feature engineering and decision models applied in the literature.

Taghiyeh et al. (2020) propose a decision model-based selection method that relies on a classification method. It selects the optimal forecasting method from a pool of candidates based on their in-sample and out-of-sample MSE. None of the three classification methods, including Logistic Regression (LogR), DT, and Support Vector Machine (SVM), outperforms the others. Kück et al. (2016) propose a decision model-based selection based on the out-of-sample sMAPE and meta-features. They apply an MLP classifier as decision model and select its inputs using a grid search over 127 feature sets that include error measures and meta-features.

The approaches above have the disadvantage that all candidate forecasting methods must be trained on the target data set to determine the applied error measures. Computing meta-features, in contrast, does not require training. Hence, relying only on meta-features for decision model-based selection saves computing time. Widodo and Budi (2013) propose a kNN classifier to select the forecasting method for a target time series based on the meta-features introduced in reference (Wang et al., 2009). In the design of the classifier, the authors apply FS for meta-feature selection. Another approach based on the same meta-features is introduced by Scholz-Reiter et al. (2014). They use a meta-feature aggregation instead of meta-feature selection, and a linear discriminant analysis (LDA) as a classification method to select the optimal forecasting method. Shahoud et al. (2020) introduce statistical meta-features for different aggregation levels of the time series to extract characteristics at different time scales. They select suitable meta-features by a BE and aggregate them using an autoencoder. RF and MLP classifiers are compared for decision-making, both optimized with a grid search, where the MLP classifier achieves better performance in terms of selecting the best forecasting method. In addition to statistical and time series meta-features, Cui et al. (2016) include domain information. The domain-based meta-features describe the physical properties of buildings for which energy consumption is to be forecast. Bauer, Züfle, Herbst, et al. (2020) evaluate three types of decision models, that is, classification, regression, and

TABLE 8 Summary of meta-feature selection and aggregation methods for decision models to select forecasting methods

		Meta-feature		Decision model		
Reference	**Forecasting method(s)**	**Selection**	**Aggregation**	**Method**	**Type**	**HPO**
Taghiyeh et al., 2020	autoARIMA, MA, SES, DES, TES, Theta			LogR, SVM, DT	class.	
Kück et al., 2016	SES, DES, TES	Grid search		MLP	class.	Grid search
Widodo & Budi, 2013	autoARIMA, MKL, PR, SVR	FS		kNN	class.	
Scholz-Reiter et al., 2014	MLP, autoARIMA, ETS, LC, LL, RW		PCA	LDA	class.	
Shahoud et al., 2020	DT, GBM, LR, RF	BE	Autoencoder	MLP, RF	class.	Grid search
Cui et al., 2016	MLP, GP, MARS, PR, RBFNN, SVR	PCA, Pearson	SVD	MLP	class.	Grid search
Baddour et al., 2018	autoARIMA, BSTS, ETS, GBM, Prophet, TBATS, Theta	Use all		RF	class.	Grid search
Bauer, Züfle, Grohmann, et al., 2020	Telescope: Cubist, Evtree, GBM, NNetAR, RPaRT, SVR	Use all		RF	reg., class.	
Bandara et al., 2020	LSTM	Use all		k-means, DBSCAN, Snob	clust.	Embedded
Montero-Manso et al., 2020	AR, autoARIMA, ETS, NNetAR, RW, dRW, sRW, TBATS, Theta	Use all		GBM	class.	Bo
Li et al., 2020	autoARIMA, ETS, NNetAR, RW, dRW, sRW, STL-AR, TBATS, Theta	Learning		GBM	class.	Not described
Ma & Fildes, 2021	autoARIMA, ELM, ETS, GBM, LR, RF, SVR	Learning		CNN, FCNN	class.	Grid search
Züfle et al., 2019	autoARIMA, ETS, NNetAR, RW	Use all		LR	reg.	
Villegas et al., 2018	MA, IMA, White Noise	Grid search		SVM	class.	Grid search

Abbreviations: class., classification; clust., clustering; reg., regression.

hybrid. In the classification decision model, an RF is trained to map meta-features to the forecasting method with the lowest forecast error. In the regression decision model, an RF learns how much worse each forecasting method is compared to the best method (i.e., forecast accuracy degradation). The hybrid decision model combines this RF regression with an RF classifier that maps the RF regression output to the best method. In the evaluation, the hybrid approach achieves the best performance in terms of forecast accuracy degradation. Instead of training an individual model for each new time series, Bandara et al. (2020) suggest clustering time series using meta-features and training only one model for each cluster, which is applied to all-time series in the cluster.

7.2 | Forecast ensembling

Forecast ensembling aims to improve the forecast robustness by bundling multiple forecasts of different models. We differentiate forecast ensembling from ensemble learning methods that build an ensemble of weak models,[32] such as RF and GBM. Ensembling the forecasts from a pool of different forecasting models aims to avoid occasional poor forecasts, rather than outperforming the best individual forecasting model (Shaub, 2020). In the following, methods for

ensembling in automated forecasting pipelines are introduced and exemplified using the reviewed literature, guided by the summary in Table 7.

The benefit of forecast ensembling is empirically demonstrated in many cases. For example, in the analysis of the M3 forecasting competition (Makridakis & Hibon, 2000), averaging the model output of all submitted forecasting methods performs better than each individual method itself. To improve averaging (also called voting ensembling), one may weigh the forecasting models' outputs according to the expected individual forecast accuracy based on the forecasting models' errors, either calculated in-sample (training data) or out-of-sample (validation data). One weighting method is to use the multiplicative inverse of the forecasting models' errors. Another weighting method is to rank the forecasting models in descending order of their errors and use their rank as weight. A more sophisticated ensembling method is to train a meta-model to combine the forecasting models' outputs (also called stacking), for example, using the output of the different forecasting models as inputs of a regression meta-model and training it to estimate the target variable. Another sophisticated ensembling method is to find the weights using an optimization method, that is, considering the weights as decision variables and varying them to minimize the error of the ensemble.

An ensemble can also be improved by only considering the k-best candidate methods, ranked by a forecasting method selection beforehand (i.e., heuristic, empirical, and decision model-based methods). The number of considered candidates k is either specified manually by the data scientist or determined automatically. For this, one can use statistical tests to drop out candidates that significantly perform worse than the best method (McDonald & Thompson, 1967), as well as use FS or BE, similar to their application in feature selection, described in Section 5.2.

Simple forecast ensembling through averaging is used by Martínez, Charte, et al. (2019). After the identification of optimal lag features with FS, they average the output of three kNN models with $k \in \{3,5,7\}$. Improving the ensemble by weighted averaging is applied by Tak et al. (2021). The authors use fuzzy c-means clustering to determine the optimal weights of six different forecasting models. First, the outputs of these six models are clustered into six clusters. Second, the membership degrees of each model in each cluster are determined and are considered as their weights, so each cluster becomes a potential ensemble. Third, the ensemble that achieves the best forecast accuracy (RMSE or MAPE) is empirically selected.

Selecting the candidate methods based on a heuristic forecasting method selection is proposed by Pawlikowski and Chorowska (2020). They categorize the time series data of the M4 forecasting competition (Makridakis et al., 2020) in terms of their frequency, the existence of a trend and seasonality. Depending on the category, they select a distinct pool of candidate forecasting methods. The hyperparameters of the candidate methods are optimized and the weight for each candidate is determined based on the sMAPE error, validated out-of-sample with a rolling origin cross-validation.

The following authors use empirical forecasting method selection to consider only the k-best candidate methods in the ensemble. After automatically selecting the input features with the INF, Crone and Kourentzes (2010) empirically determine the forecast accuracy of candidate MLP architectures in a grid search (N_h, activation) with a rolling origin cross-validation and average the outputs of the 10 best candidates to reduce the impact of overfitting. Kourentzes et al. (2019) propose an FS heuristic to decide on the number of ranked candidates to be considered for averaging. They calculate the forecast accuracy metric's rate of increase C' assigned to each forecast and include all candidates until the first steep increase $C' > T$. To detect the increase, they use the same approach used for detecting outliers in boxplots, that is, $T = \mathrm{Q3} + 1.51\mathrm{QR}$, where Q3 is the third quartile.

Instead of averaging the k-best candidates, Shetty and Shobha (2016) assign weights to the filtered candidates before averaging. Candidates with low forecast errors Q receive more weight, that is,

$$w_i = \frac{v_i}{\sum_{j=1}^{k} v_j}, \qquad v_i = \frac{\prod_{j=1}^{k} Q_j}{Q_i \sum_{j=1}^{k} Q_j}. \tag{8}$$

Wu et al. (2020) propose a multi-objective optimization to determine the optimal weights for averaging candidates. They apply the flower pollination metaheuristic to minimize

$$\min \sum_{i=1}^{4} w_i Q_i, \qquad \text{subject to} \sum_{i=1}^{4} w_i = 1, \tag{9}$$

with the error measures Q_i including the MAE, RMSE, and their relative formulations.

In the decision model-based method selection, one can also include weighted averaging directly in the decision model. Montero-Manso et al. (2020) and Li et al. (2020) train a GBM classifier with softmax-transformed outputs corresponding to the weights of the candidates for averaging. Similarly, Ma and Fildes (2021) compare a convolutional neural network (CNN) and a fully connected neural network (FCNN) classifier using output neurons with softmax activation to predict the weights for averaging. Instead of softmax outputs, Züfle et al. (2019) use an LR as a decision model. Its output reflects the probability of how well the forecasting method fits the time series and determines the weight for averaging. For filtering the k-best candidate methods, they postprocess the LR output.

Above-mentioned forecast ensembling methods with weighted average determine the weights statically, that is, the weights do not change as the time series evolves. Sergio et al. (2016) propose a dynamic weighting method for the ensembling of forecasts. For every single forecast, the method searches for the *k*-nearest patterns in the training data similar to the given input data. Three ensemble functions—average, median, and softmax—are evaluated on the found similar patterns, and the method with the best forecast accuracy is chosen for the forecast. A dynamic decision model-based approach is also introduced by Villegas et al. (2018). In their work, a binary SVM classifier is trained on forecast accuracy metrics and meta-features to predict the best forecasting method from a pool of candidates for each forecast origin. In their experiment, the dynamic selection achieves the best forecast accuracy compared to the mean and the median ensembling of all candidate forecasting methods.

7.3 | Discussion

We discuss the automated selection of the optimal forecasting methods as the fourth and forecast ensembling as the fifth section of the automated forecasting pipeline. For both pipeline sections, we highlight potential problems, give recommendations, and suggest future work.

The automated selection includes heuristic, empirical, and decision model-based methods. As shown in Table 7, most empirical selection methods for the forecasting method are based on evaluation criteria whereas decision models base their selection mainly on meta-features. None of the reviewed papers using empirical selection methods applies statistical tests to check whether the differences in the empirically determined performance scores are significant.[33] Considering automation, the selection of the forecasting method is often combined with an individual HPO of the candidate forecasting methods. If multiple forecasting methods are evaluated in the selection, about half of the reviewed literature combines the selection with forecast ensembling.

In the automated selection, we notice different potential problems. The heuristic forecasting method selection is based on straightforward decision rules. To strengthen the evidence of the selection, however, our impression is that evaluation criteria or meta-features are needed. The empirical selection based on evaluation criteria requires a high computing effort as every candidate forecasting method needs to be fitted to the data, and the determined performance scores might be subject to statistical fluctuations. The selection with a decision model based on meta-features reduces the computing effort but requires a sufficiently large and diverse data set for training the decision model. Based on our analysis, a comprehensive benchmark comparing the computing effort and the forecast accuracy of heuristic, empirical, and decision model-based forecasting method selection is missing, and therefore no recommendation can be made. Empirical evidence, however, exists for the forecast ensembling. We recommend combining forecast ensembling with the forecasting method selection by determining the pool of candidate methods and ensemble weights. In the reviewed literature that applies ensembling, we recognize untapped potential for automation that can be leveraged by applying more sophisticated methods, for example, automatically reducing the candidate pool using statistical tests to drop out candidates significantly underperforming or by using FS or BE.

Based on the challenges of automated forecasting method selection, future work could tailor decision models for specific domains using HPO. Furthermore, the research could analyze if the selection of pretrained forecasting methods is beneficial—either for application to the new time series without adaptation or after re-training on the new data.

8 | AUTOMATED FORECASTING PIPELINE

The forecasting pipeline consists of the sections preprocessing, feature engineering, HPO, and forecasting method selection and ensembling (Table 9).

TABLE 9 Overview of the forecasting pipeline sections automated in the reviewed literature

Reference	Preprocessing	Feature engineering	Hyper-parameter optimization	Forecasting method selection	Forecast Ensembling	Covered sections	Cluster
Liu et al., 2017; Lu & AbouRizk, 2009; Anvari et al., 2016; Cerqueira et al., 2021	X	–	–	–	–	●	A
Hyndman et al., 2002; Svetunkov & Petropoulos, 2018; Pedregal, 2019; Bermúdez et al., 2012; Hwang et al., 2012; Villegas & Pedregal, 2019; Arlt & Trcka, 2021; Silva et al., 2020	–	–	X	–	–	●	B
Pereira et al., 2018; Shcherbakov et al., 2013; Shahoud et al., 2020; Taghiyeh et al., 2020; Villegas et al., 2018; Kück et al., 2016; Cui et al., 2016; Scholz-Reiter et al., 2014; Baddour et al., 2018	–	–	–	X	–	●	C
Kourentzes & Crone, 2010; Martínez, Charte, et al., 2019; Bauer, Züfle, Grohmann, et al., 2020	X	X	–	–	–	●●	D
Hyndman & Khandakar, 2008; Maravall et al., 2015; Dellino et al., 2018a; Alzyout et al., 2019; Tran & Reed, 2004	X	–	X	–	–	●●	E
Fildes & Petropoulos, 2015; Violos et al., 2020	X	–	–	X	–	●●	F
Lowther et al., 2020; Chakrabarti & Faloutsos, 2002; Valente & Maldonado, 2020; Son & Kim, 2015; Aladag et al. (2014); Eğrioğlu & Fildes, 2022	–	X	X	–	–	●●	G
Balkin & Ord, 2000	–	X	–	X	–	●●	H
Spiliotis et al., 2020; Bandara et al., 2020	–	–	X	X	–	●●	I
Li et al., 2020; Montero-Manso et al., 2020; Shetty & Shobha, 2016; Kourentzes et al., 2019; Wu et al., 2020; Pawlikowski & Chorowska, 2020; Tak et al., 2021	–	–	–	X	X	●●	J

TABLE 9 (Continued)

Reference	Preprocessing	Feature engineering	Hyper-parameter optimization	Forecasting method selection	Forecast Ensembling	Covered sections	Cluster
Maldonado et al., 2019; Yan, 2012; Panigrahi & Behera, 2020; Sagaert et al., 2018; Widodo et al., 2016; Donate & Cortez, 2014; Donate et al., 2013; Fan et al., 2019; Dellino et al., 2018b; Gao et al., 2021; Eğrioğlu & Bas, 2022	X	X	X	–	–	●●●	K
Amin et al., 2012; Sekma et al., 2016; Widodo & Budi, 2013	X	–	X	X	–	●●●	L
Ma & Fildes, 2021; Crone & Kourentzes, 2010; Züfle et al., 2019	X	–	–	X	X	●●●	M
Martínez, Frías, et al., 2019	X	X	–	–	X	●●●	N
Züfle & Kounev, 2020; Bauer et al., 2020	X	X	–	X	–	●●●	O
Rätz et al., 2019	X	X	X	X	–	●●●●	P
Sergio et al., 2016	X	–	X	X	X	●●●●	Q

8.1 | Status quo

Table 9 shows that the analyzed literature covers different sections of the forecasting pipeline.[34] Clustering the analyzed papers that cover the same sections of the forecasting pipeline yields 17 clusters. Most of the 69 papers reviewed cover only two or three sections of the forecasting pipeline. Only two papers cover four sections, and none of the papers reviewed covers all five sections of the forecasting pipeline.

Twenty-seven papers consider only statistical forecasting methods, 24 papers only machine learning methods, and 16 papers both. A majority of the papers that include both families of forecasting methods consider forecasting method selection in their pipeline (13 of 16 papers).

Of all the papers analyzed, most papers focus on preprocessing (36) and HPO (35), followed by the forecasting method selection (30). Since statistical methods consider lag features implicitly in the model structure, only 23 papers explicitly deal with feature engineering. The least attention is paid to the ensembling of forecasting methods, that is, in only 13 papers.

8.2 | Discussion

Based on the analysis of the status quo, we give the following recommendations to holistically automate the complete forecasting pipeline. The optimization of the hyperparameters should be combined with the automated feature selection (clusters G, K, and P in Table 9) because it is expected that each feature set candidate requires an individual hyperparameter configuration. Moreover, combining an HPO and the automated selection of forecasting methods (i.e., CASH) is advantageous as the best method is selected from a pool of candidates whose hyperparameters, in turn, are already optimized by an HPO (clusters I, L, P, and Q in Table 9). Also, the forecast ensembling should be combined with the automated selection of appropriate forecasting methods to eliminate poor candidates (clusters J, M, and Q in Table 9). Hereby, both statistical and machine learning-based forecasting methods should be considered, as the diversity of forecasting methods has the potential to increase the robustness of the results.

Regarding hybrid forecasting methods, that is, combining the advantages of multiple forecasting methods, we already see methods to automate the design process of the forecasting pipeline with respect to the hybridization approaches. Most references use the ensembling hybridization approach (Li et al., 2020; Ma & Fildes, 2021; Montero-Manso et al., 2020; Pawlikowski & Chorowska, 2020; Shetty & Shobha, 2016; Züfle et al., 2019), followed by the data preprocessing-based hybridization approach (Bauer, Züfle, Grohmann, et al., 2020), and the sequential hybridization approach (Kourentzes & Crone, 2010). However, recent hybrid forecasting methods that use deep neural networks to improve the forecast of well-studied statistical methods (Lim & Zohren, 2021) are still insufficiently automated and require expert knowledge for the pipeline design, which limits the large-scale application of these hybrid methods. In addition, it is worth considering the related research direction is end-to-end (E2E) learning. While deep learning-based hybrid modeling only considers specific sections of the design process using deep neural networks, E2E learning uses one deep neural network to address several sections with appropriate layers. More specifically, E2E learning aims to include all five sections of the forecasting model's design process into the training using gradient-based learning, for example, embedded feature learning in a deep neural network (Ma & Fildes, 2021). As in automated forecasting pipelines, both deep learning-based hybrid modeling and E2E learning need to consider all five sections of the design process to obtain high-performing and robust forecasts.

Regarding automating the design process, emerging approaches such as neural architecture search and meta-learning (Hutter et al., 2019) are promising for both hybrid forecasting methods and E2E learning in the future.

Finally, we note that manual tailoring of forecasting models is still a common practice in time series forecasting competitions despite the first successful applications of automated forecasting pipelines. For example, in the most recent M5 forecasting competition (Makridakis et al., 2021), Liang and Lu (2020) provide evidence that their AutoML pipeline achieves competitive forecast accuracy with moderate computing effort. Therefore, we assume that the potential of the automated design is not fully utilized yet in time series competitions.

9 | CONCLUSION

Time series are collected in many domains, and forecasting their progression over a certain future period is becoming increasingly important for many use cases. This rapidly growing demand requires making the design process of time series forecasts more efficient by automation; that is, automating design decisions within each section of the forecasting pipeline or automatically combining methods across pipeline sections. Although the manual design process of forecasting models and the first aspect of design automation has already been considered by various researchers, understanding how various automation methods interact within the pipeline and how they can be combined is a critical open question. Therefore, the article reviews existing literature on automated time series forecasting pipelines and analyzes how each of the five sections in these pipelines are automated. It also investigates the reviewed literature in terms of the interaction and combination of automation methods within the five pipeline sections, incorporating both AutoML and automated statistical forecasting methods.

Besides various specific insights on automation related to each pipeline section that we discuss throughout the corresponding sections, we find on a general level that the majority of the 69 reviewed papers only cover two or three of the five sections of the forecasting pipeline. Therefore, we conclude that there is a research gap regarding approaches that holistically consider the automation of the forecasting pipeline, enabling the large-scale application to use cases without manual and time-consuming tailoring.

Besides the holistic automation, future work should research the adaption of the identified automation methods for probabilistic time series forecasts. Concurrently, the automated forecasting pipelines should be validated and tailored to particular use cases before generalizing them to universal automated forecasting pipelines. The resulting forecasting models can either be applied directly or serve as a starting point for advanced manual tailoring with expert knowledge. In any case, future work should examine their performance in terms of forecast accuracy and computing effort, for example, in forecasting competitions. This performance evaluation would benefit if future work on automated forecasting pipelines would be open source—both the implementation of the evaluated methods and the data sets used for the evaluation. Moreover, open-source publishing should promote the adoption of automated pipelines for time series forecasting, thus leveraging the great potential of improving the design efficiency through automation, achieving a high forecast accuracy and a robust operation.

Overall, the article focuses on the automated design of forecasting pipelines. Therefore, to fully automate the entire forecasting process, future work should also consider the automated application of the resulting forecasting models,

including performance monitoring (e.g., forecast accuracy and computing effort) during operation and model adaption.[35]

AUTHOR CONTRIBUTIONS

Stefan Meisenbacher: Conceptualization (lead); formal analysis (lead); investigation (lead); methodology (lead); validation (lead); visualization (lead); writing – original draft (lead). **Marian Turowski:** Conceptualization (equal); methodology (equal); visualization (equal); writing – review and editing (lead). **Kaleb Phipps:** Conceptualization (equal); methodology (equal); visualization (equal); writing – review and editing (lead). **Martin Rätz:** Conceptualization (equal); methodology (equal); visualization (equal); writing – review and editing (equal). **Dirk Müller:** Funding acquisition (equal); supervision (equal); writing – review and editing (supporting). **Veit Hagenmeyer:** Funding acquisition (equal); supervision (equal); writing – review and editing (supporting). **Ralf Mikut:** Conceptualization (lead); funding acquisition (equal); methodology (equal); supervision (equal); writing – review and editing (lead).

ACKNOWLEDGMENT

Open Access funding enabled and organized by Projekt DEAL.

FUNDING INFORMATION

The Helmholtz Association's Initiative and Networking Fund through Helmholtz AI; the Helmholtz Association under the Program "Energy System Design"; the German Research Foundation (DFG) as part of the Research Training Group 2153 "Energy Status Data: Informatics Methods for its Collection, Analysis and Exploitation"; Federal Ministry for Economic Affairs and Energy (BMWi), promotional reference 03ET1568.

CONFLICT OF INTEREST

The authors have declared no conflicts of interest for this article.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study. Search terms are available in Supporting Information.

ORCID

Stefan Meisenbacher https://orcid.org/0000-0002-9320-5341
Marian Turowski https://orcid.org/0000-0002-3776-2215
Kaleb Phipps https://orcid.org/0000-0002-9197-1739
Martin Rätz https://orcid.org/0000-0002-3573-2872
Dirk Müller https://orcid.org/0000-0002-6106-6607
Veit Hagenmeyer https://orcid.org/0000-0002-3572-9083
Ralf Mikut https://orcid.org/0000-0001-9100-5496

ENDNOTES

[1] In addition to automating the design process, automating the operation of forecasts is proposed by Meisenbacher et al. (2021), which includes self-monitoring and automatic model adaption as forecast accuracy decreases.

[2] In the following, we consider deep learning methods as part of the broader family of machine learning.

[3] https://www.scopus.com/.

[4] The applied search strings and keywords are documented in the Supporting Information.

[5] https://beallslist.net/.

[6] After publishing the preprint of this article, we have reviewed further literature suggestions from the community and later published publications accordingly to keep the paper up to date for publication, resulting in six additional reviewed articles.

[7] The vectors that include past values can be sparse, that is, only certain time points from k to H_1 are included.

[8] Brockwell and Davis (2016) provide mathematical details on the maximum likelihood estimation.

[9] More details on training algorithms for the SVR are given in reference (Smola & Schölkopf, 2004).

[10] Natekin and Knoll (2013) provide further details on GBMs, and Lee et al. (2020) detail the concept of bagging and RFs.

[11] Bose and Mali (2019) review methods for automated partitioning and learning fuzzy logical relationships in FTS forecasting.

[12] Further details on the theory behind the backpropagation algorithm can be found in reference (Wythoff, 1993).

[13] The authors disaggregate the time series into the seasonal, trend, and residual components and apply anomaly detection and handling on the stationary residual.

[14] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html.

[15] More recent methods like the Multiple Seasonal-Trend decomposition using Loess (MSTL) are not yet applied in the reviewed literature.

[16] If the features are originally in a different resolution or length, they must be transformed accordingly.

[17] Two time series are necessary because, otherwise, the encoding would be ambiguous for several days.

[18] We classify the data domain according to the following categories: economics and finance, energy, nature and demographics, human access, and other or unknown. The categories are adapted from the references (Bauer et al., 2021; Makridakis & Hibon, 2000) and extended to cover the literature reviewed.

[19] Authors that apply forecasting methods to data from several domains mainly use univariate time series from forecasting competitions, where no exogenous time series are provided.

[20] Since the embedded feature selection is specifically designed for the training algorithm of the respective forecasting method, the transfer to other forecasting methods is not straightforward.

[21] As the dimensionality of the feature space increases, the available training data becomes sparse.

[22] Since the PCA assumes a normal distribution, anomalies must be identified and handled beforehand, and appropriate transformations need to be performed to obtain a stationary time series (Yang & Shahabi, 2005).

[23] A well-known exception is RNNs that feedback input values within the neural structure to capture sequential relationships.

[24] For multiple-point-ahead forecasts, future values of the exogenous time series are required if lag, cyclical, and transformation features are extracted from them; either through simultaneous forecasting or the integration of exogenous forecast results.

[25] Initial work on the automated extraction of endogenous features exists (Barandas et al., 2020; Cerqueira et al., 2021).

[26] Also, the computing effort for training or executing the forecasting method, which is often a contrarian evaluation criterion to the forecasting accuracy, can be taken into account, for example, if one faces limited computing resources in applying it to a use case.

[27] Since ICs try to prevent overfitting implicitly with the penalty term, they are computed in-sample.

[28] The greater importance of a few hyperparameters over others applies in many cases (Bergstra & Bengio, 2012; Rätz et al., 2019).

[29] This regularization is also called *early stopping*.

[30] Selecting the optimal forecasting method and finding the optimal hyperparameter configuration is also called the combined algorithm selection and hyperparameter optimization (CASH) problem.

[31] For a detailed description of evaluation criteria, as well as in-sample and out-of-sample validation, refer to Section 6.1.

[32] The forecast of a weak model, for example, a DT, is only slightly superior to a random estimate. Ensemble learning aims to combine many weak models to achieve a good estimate.

[33] Using statistical tests to automatically select the most suitable forecasting method can lead to ambiguous results if there is no significant difference in performance scores. Therefore, we rather consider them for ensembling to reduce the candidate pool.

[34] The scope and complexity with which the analyzed literature addresses the sections of the forecasting pipeline are analyzed in the previous Sections 4–7 of this article.

[35] For the fusion of automated design and automated application, automation levels are defined by Meisenbacher et al. (2021).

RELATED WIREs ARTICLES

Energy forecasting tools and services
Data mining tools

FURTHER READING

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, *6*(1), 3–73.

REFERENCES

Ahmad, T., Zhang, H., & Yan, B. (2020). A review on renewable energy and electricity requirement forecasting models for smart grid and buildings. *Sustainable Cities and Society*, *55*, 102052. https://doi.org/10.1016/j.scs.2020.102052

Aladag, C. H., Yolcu, U., Eğrioğlu, E., & Bas, E. (2014). Fuzzy lagged variable selection in fuzzy time series with genetic algorithms. *Applied Soft Computing*, *22*, 465–473. https://doi.org/10.1016/.asoc.2014.03.028

Alzyout, M., Alsmirat, M., & Al-Saleh, M. I. (2019). Automated ARIMA model construction for dynamic vehicle GPS location prediction. In *Proceedings of the 2019 Sixth International Conference on Internet of Things*: *Systems*, *Management and Security (IOTSMS)*, pp. 380–386. https://doi.org/10.1109/IOTSMS48152.2019.8939197

Amin, A., Grunske, L., & Colman, A. (2012). An automated approach to forecasting QoS attributes based on linear and non-linear time series modeling. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pp. 130–139. https://doi.org/10.1145/2351676.2351695

Anvari, S., Tuna, S., Canci, M., & Turkay, M. (2016). Automated Box-Jenkins forecasting tool with an application for passenger demand in urban rail systems. *Journal of Advanced Transportation*, *50*(1), 25–49. https://doi.org/10.1002/atr.1332

Arlt, J., & Trcka, P. (2021). Automatic SARIMA modeling and forecast accuracy. *Communications in Statistics: Simulation and Computation*, *50*(10), 2949–2970. https://doi.org/10.1080/03610918.2019.1618471

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms (first)*. Oxford University Press.

Baddour, K. E., Ghasemi, A., & Rutagemwa, H. (2018). Spectrum occupancy prediction for land mobile radio bands using a recommender system. In *Proceedings of the 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–6. https://doi.org/10.1109/VTCFall.2018.8690654

Balkin, S. D., & Ord, J. K. (2000). Automatic neural network modeling for univariate time series. *International Journal of Forecasting*, *16*(4), 509–515. https://doi.org/10.1016/S0169-2070(00)00072-8

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, *140*, 112896. https://doi.org/10.1016/j.eswa.2019.112896

Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., Liu, H., Schultz, T., & Gamboa, H. (2020). TSFEL: Time series feature extraction library. *SoftwareX*, *11*, 100456. https://doi.org/10.1016/j.softx.2020.100456

Bauer, A., Züfle, M., Eismann, S., Grohmann, J., Herbst, N., & Kounev, S. (2021). Libra: A benchmark for time series forecasting methods. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pp. 189–200. https://doi.org/10.1145/3427921.3450241

Bauer, A., Züfle, M., Grohmann, J., Schmitt, N., Herbst, N., & Kounev, S. (2020). An automated forecasting framework based on method recommendation for seasonal time series. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pp. 48–55. https://doi.org/10.1145/3358960.3379123

Bauer, A., Züfle, M., Herbst, N., Kounev, S., & Curtef, V. (2020). Telescope: An automatic feature extraction and transformation approach for time series forecasting on a level-playing field. In *Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 1902–1905. https://doi.org/10.1109/ICDE48307.2020.00199

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2006). *Nonlinear programming: Theory and algorithms (3rd ed.)*. John Wiley & Sons. https://doi.org/10.1002/0471787779

Bergmeir, C. (2013). *New approaches in time series forecasting: Methods, software and evaluation procedures [Doctoral dissertation, Universidad de Granada]*. Repositorio Institucional de la Universidad de Granada.

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213. https://doi.org/10.1016/j.ins.2011.12.028

Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, *120*, 70–83. https://doi.org/10.1016/j.csda.2017.11.003

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305.

Bermúdez, J. D., Segura, J. V., & Vercher, E. (2012). SIOPRED performance in a forecasting blind competition. In *Proceedings of the 2012 IEEE Conference on Evolving and Adaptive Intelligent Systems*, pp. 192–197. https://doi.org/10.1109/EAIS.2012.6232828

Bertsimas, D., King, A., & Mazumder, R. (2016). Best subset selection via a modern optimization lens. *The Annals of Statistics*, *44*(2), 813–852. https://doi.org/10.1214/15-AOS1388

Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*, *54*(3), 1–33. https://doi.org/10.1145/3444690

Boone, T., Ganeshan, R., Jain, A., & Sanders, N. R. (2019). Forecasting sales in the supply chain: Consumer analytics in the big data era. *International Journal of Forecasting*, *35*(1), 170–180. https://doi.org/10.1016/j.ijforecast.2018.09.003

Bose, M., & Mali, K. (2019). Designing fuzzy time series forecasting models: A survey. *International Journal of Approximate Reasoning*, *111*, 78–99. https://doi.org/10.1016/j.ijar.2019.05.002

Box, G. E. P., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B: Methodological*, *26*(2), 211–243.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). *Time series analysis: Forecasting and control* (5th ed.). Wiley.

Brockwell, P. J., & Davis, R. A. (2016). *Introduction to time series and forecasting* (3rd ed.). Springer International Publishing.

Canova, F., & Hansen, B. E. (1995). Are seasonal patterns constant over time? A test for seasonal stability. *Journal of Business & Economic Statistics*, *13*(3), 237–252.

Cerqueira, V., Moniz, N., & Soares, C. (2021). VEST: Automatic feature engineering for forecasting. *Machine Learning*. 1–23. https://doi.org/10.1007/s10994-021-05959-y

Chakrabarti, D., & Faloutsos, C. (2002). F4: Large-scale automated forecasting using fractals. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pp. 2–9. https://doi.org/10.1145/584792.584797

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 1–58. https://doi.org/10.1145/1541880.1541882

Chen, X., Yu, R., Ullah, S., Wu, D., Li, Z., Li, Q., Qi, H., Liu, J., Liu, M., & Zhang, Y. (2022). A novel loss function of deep learning in wind speed forecasting. *Energy*, *238*, 121808. https://doi.org/10.1016/j.energy.2021.121808

Cheng, C., Sa-Ngasoongsong, A., Beyca, O., Le, T., Yang, H., Kong, Z., & Bukkapatnam, S. T. S. (2015). Time series forecasting for nonlinear and non-stationary processes: A review and comparative study. *IIE Transactions*, *47*(10), 1053–1071. https://doi.org/10.1080/0740817X.2014.999180

Cheung, Y.-W., & Lai, K. S. (1995). Lag order and critical values of the augmented dickey-fuller test. *Journal of Business & Economic Statistics*, *13*(3), 277–280.

Cox, D. R., & Stuart, A. (1955). Some quick sign tests for trend in location and dispersion. *Biometrika*, *42*(1/2), 80–95.

Crone, S. F., & Kourentzes, N. (2010). Feature selection for time series prediction: A combined filter and wrapper approach for neural networks. *Neurocomputing*, *73*(10), 1923–1936. https://doi.org/10.1016/j.neucom.2010.01.017

Cui, C., Wu, T., Hu, M., Weir, J. D., & Li, X. (2016). Short-term building energy model recommendation system: A meta-learning approach. *Applied Energy*, *172*, 251–263. https://doi.org/10.1016/j.apenergy.2016.03.112

De Gooijer, J. G., & Hyndman, R. J. (2006). 25 Years of time series forecasting. *International Journal of Forecasting*, *22*(3), 443–473. https://doi.org/10.1016/j.ijforecast.2006.01.001

Dellino, G., Laudadio, T., Mari, R., Mastronardi, N., & Meloni, C. (2018a). Microforecasting methods for fresh food supply chain management: A computational study. *Mathematics and Computers in Simulation*, *147*, 100–120. https://doi.org/10.1016/j.matcom.2017.12.006

Dellino, G., Laudadio, T., Mari, R., Mastronardi, N., & Meloni, C. (2018b). A reliable decision support system for fresh food supply chain management. *International Journal of Production Research*, *56*(4), 1458–1485. https://doi.org/10.1080/00207543.2017.1367106

Donate, J. P., & Cortez, P. (2014). Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting. *Applied Soft Computing*, *23*, 432–443. https://doi.org/10.1016/j.asoc.2014.06.041

Donate, J. P., Li, X., Sánchez, G. G., & de Miguel, A. S. (2013). Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm. *Neural Computing and Applications*, *22*(1), 11–20. https://doi.org/10.1007/s00521-011-0741-0

Eğrioğlu, E., & Bas, E. (2022). A new automatic forecasting method based on a new input significancy test of a single multiplicative neuron model artificial neural network. *Network*, *33*, 1–16. https://doi.org/10.1080/0954898X.2022.2042609

Eğrioğlu, E., & Fildes, R. (2022). A new bootstrapped hybrid artificial neural network approach for time series forecasting. *Computational Economics*, *59*(4), 1355–1383. https://doi.org/10.1007/s10614-020-10073-7

Fan, S., Qin, X., Jia, Z., Qi, X., & Lin, M. (2019). ELM-based improved layered ensemble architecture for time series forecasting. *IEEE Access*, *7*, 97827–97837. https://doi.org/10.1109/ACCESS.2019.2927047

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automated machine learning* (pp. 3–33). Springer International Publishing.

Fildes, R., & Petropoulos, F. (2015). Simple versus complex selection rules for forecasting many time series. *Journal of Business Research*, *68*(8), 1692–1701. https://doi.org/10.1016/j.jbusres.2015.03.028

Gao, R., Duru, O., & Yuen, K. F. (2021). High-dimensional lag structure optimization of fuzzy time series. *Expert Systems with Applications*, *173*, 114698. https://doi.org/10.1016/j.eswa.2021.114698

González Ordiano, J. Á., Waczowicz, S., Hagenmeyer, V., & Mikut, R. (2018). Energy forecasting tools and services. *WIREs Data Mining and Knowledge Discovery*, *8*(2), e1235. https://doi.org/10.1002/widm.1235

Hajirahimi, Z., & Khashei, M. (2019). Hybrid structures in time series modeling and forecasting: A review. *Engineering Applications of Artificial Intelligence*, *86*, 83–106. https://doi.org/10.1016/j.engappai.2019.08.018

Han, Z., Zhao, J., Leung, H., Ma, K. F., & Wang, W. (2019). A review of deep learning models for time series prediction. *IEEE Sensors Journal*, *21*(6), 7833–7848. https://doi.org/10.1109/JSEN.2019.2923982

Harvey, D., Leybourne, S., & Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, *13*(2), 281–291. https://doi.org/10.1016/S0169-2070(96)00719-4

Heidrich, B., Bartschat, A., Turowski, M., Neumann, O., Phipps, K., Meisenbacher, S., Schmieder, K., Ludwig, N., Mikut, R., & Hagenmeyer, V. (2021). pyWATTS: Python workflow automation tool for time series. Retrieved from https://github.com/KIT-IAI/pyWATTS.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning: Methods, systems, challenges*. Springer International Publishing.

Hwang, S., Park, M., Lee, H.-S., & Kim, H. (2012). Automated time-series cost forecasting system for construction materials. *Journal of Construction Engineering and Management*, *138*(11), 1259–1269. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000536

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTexts.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, *27*(3), 1–22. https://doi.org/10.18637/jss.v027.i03

Hyndman, R. J., Koehler, A. B., Snyder, R. D., & Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, *18*(3), 439–454. https://doi.org/10.1016/S0169-2070(01)00110-8

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688.

Jović, A., Brkić, K., & Bogunović, N. (2015). A review of feature selection methods with applications. In *Proceedings of the 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1200–1205. https://doi.org/10.1109/MIPRO.2015.7160458

Kourentzes, N., Barrow, D., & Petropoulos, F. (2019). Another look at forecast selection and combination: Evidence from forecast pooling. *International Journal of Production Economics*, *209*, 226–235. https://doi.org/10.1016/j.ijpe.2018.05.019

Kourentzes, N., & Crone, S. F. (2010). Frequency independent automatic input variable selection for neural networks for forecasting. In *Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. https://doi.org/10.1109/IJCNN.2010.5596637

Koutsandreas, D., Spiliotis, E., Petropoulos, F., & Assimakopoulos, V. (2021). On the selection of forecasting accuracy measures. *Journal of the Operational Research Society*, *73*, 937–954. https://doi.org/10.1080/01605682.2021.1892464

Kück, M., Crone, S. F., & Freitag, M. (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1499–1506. https://doi.org/10.1109/IJCNN.2016.7727376

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, *54*(1), 159–178. https://doi.org/10.1016/0304-4076(92)90104-Y

Lee, T.-H., Ullah, A., & Wang, R. (2020). Bootstrap aggregating and random forest. In P. Fuleky (Ed.), *Macroeconomic forecasting in the era of big data: Theory and practice* (pp. 389–429). Springer International Publishing. https://doi.org/10.1007/978-3-030-31150-6_13

Li, X., Kang, Y., & Li, F. (2020). Forecasting with time series imaging. *Expert Systems with Applications*, *160*, 113680. https://doi.org/10.1016/j.eswa.2020.113680

Liang, C., & Lu, Y. (2020). *Using AutoML for time series forecasting*. Google AI Blog, Google Research. Retrieved from. http://ai.googleblog.com/2020/12/using-automl-for-time-series-forecasting.html

Lilliefors, H. W. (1967). On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, *62*(318), 399–402.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194), 20200209. https://doi.org/10.1098/rsta.2020.0209

Liu, S., Gu, S., & Bao, T. (2017). An automatic forecasting method for time series. *Chinese Journal of Electronics*, *26*(3), 445–452. https://doi.org/10.1049/cje.2017.01.011

Ljung, G. M., & Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, *65*(2), 297–303.

Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., & Király, F. J. (2019). sktime: A unified interface for machine learning with Time Series. In *2019 Workshop on Systems for ML at NeurIPS*. Retrieved from https://www.sktime.org/en/stable/api_reference.html.

Lowther, A. P., Fearnhead, P., Nunes, M. A., & Jensen, K. (2020). Semi-automated simultaneous predictor selection for regression-SARIMA models. *Statistics and Computing*, *30*(6), 1759–1778. https://doi.org/10.1007/s11222-020-09970-6

Lu, Y., & AbouRizk, S. M. (2009). Automated Box-Jenkins forecasting modelling. *Automation in Construction*, *18*(5), 547–558. https://doi.org/10.1016/j.autcon.2008.11.007

Ma, S., & Fildes, R. (2021). Retail sales forecasting with meta-learning. *European Journal of Operational Research*, *288*(1), 111–128. https://doi.org/10.1016/j.ejor.2020.05.038

Makridakis, S., & Hibon, M. (2000). The M3 competition: Results, conclusions and implications. *International Journal of Forecasting*, *16*(4), 451–476. https://doi.org/10.1016/S0169-2070(00)00057-1

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*(1), 54–74. https://doi.org/10.1016/j.ijforecast.2019.04.014

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2021). The M5 competition: Background, organization, and implementation. *International Journal of Forecasting*. https://doi.org/10.1016/j.ijforecast.2021.07.007

Maldonado, S., González, A., & Crone, S. (2019). Automatic time series analysis for electric load forecasting via support vector regression. *Applied Soft Computing*, *83*, 105616. https://doi.org/10.1016/j.asoc.2019.105616

Maravall, A., López-Pavón, R., & Pérez-Cañete, D. (2015). Reliability of the automatic identification of ARIMA models in program TRAMO. In J. Beran, Y. Feng, & H. Hebbel (Eds.), *Empirical economic and financial research* (Vol. 48, pp. 105–122). Springer International Publishing.

Martínez, F., Charte, F., Rivera, A. J., & Frías, M. P. (2019). Automatic time series forecasting with GRNN: A comparison with other models. In I. Rojas, G. Joya, & A. Catala (Eds.), *Advances in computational intelligence* (Vol. 11506, pp. 198–209). Springer International Publishing. https://doi.org/10.1007/978-3-030-20521-8_17

Martínez, F., Frías, M. P., Pérez, M. D., & Rivera, A. J. (2019). A methodology for applying k-nearest neighbor to time series forecasting. *Artificial Intelligence Review*, *52*(3), 2019–2037. https://doi.org/10.1007/s10462-017-9593-z

Masini, R. P., Medeiros, M. C., & Mendes, E. F. (2021). Machine learning advances for time series forecasting. *Journal of Economic Surveys*, 1–36. https://doi.org/10.1111/joes.12429

McDonald, B. J., & Thompson, W. A. (1967). Rank sum multiple comparisons in one- and two-way classifications. *Biometrika*, *54*(3/4), 487–497. https://doi.org/10.2307/2335040

Meisenbacher, S., Pinter, J., Martin, T., Hagenmeyer, V., & Mikut, R. (2021). Concepts for automated machine learning in smart grid applications. Proceedings of the 31st Workshop on Computational Intelligence, Berlin, 25–26 November 2021, pp. 11–35. https://doi.org/10.5445/KSP/1000138532

Mohammadi, S. (2018). A new test for the significance of neural network inputs. *Neurocomputing*, *273*, 304–322. https://doi.org/10.1016/j.neucom.2017.08.007

Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting*, *36*(1), 86–92. https://doi.org/10.1016/j.ijforecast.2019.02.011

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *7*, 21. https://doi.org/10.3389/fnbot.2013.00021

Osborn, D. R., Chui, A. P. L., Smith, J. P., & Birchenhall, C. R. (1988). Seasonality and the order of integration for consumption. *Oxford Bulletin of Economics and Statistics*, *50*(4), 361–377. https://doi.org/10.1111/j.1468-0084.1988.mp50004002.x

Panigrahi, S., & Behera, H. S. (2020). Time series forecasting using differential evolution-based ANN modelling scheme. *Arabian Journal for Science and Engineering*, *45*(12), 11129–11146. https://doi.org/10.1007/s13369-020-05004-5

Pawlikowski, M., & Chorowska, A. (2020). Weighted ensemble of statistical models. *International Journal of Forecasting*, *36*(1), 93–97. https://doi.org/10.1016/j.ijforecast.2019.03.019

Pearl, J. (1984). *Heuristics: Intelligent search strategies for computer problem solving* (1st ed.). Addison Wesley Publishing Company.

Pedregal, D. J. (2019). Time series analysis and forecasting with ECOTOOL. *PLoS One*, *14*(10), 1–23. https://doi.org/10.1371/journal.pone.0221238

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(85), 2825–2830.

Pereira, P., Araujo, J., Matos, R., Preguica, N., & Maciel, P. (2018). Software rejuvenation in computer systems: An automatic forecasting approach based on time series. In *Proceedings of the 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–8. https://doi.org/10.1109/PCCC.2018.8711347

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Ben Taieb, S., Bergmeir, C., Bessa, R. J., Bijak, J., Boylan, J. E., Browell, J., Carnevale, C., Castle, J. L., Cirillo, P., Clements, M. P., Cordeiro, C., Cyrino Oliveira, F. L., De Baets, S., Dokumentov, A., ... Ziel, F. (2022). Forecasting: Theory and practice. *International Journal of Forecasting*, *38*, 705–871. https://doi.org/10.1016/j.ijforecast.2021.11.001

Rahimi, I., Chen, F., & Gandomi, A. H. (2021). A review on COVID-19 forecasting models. *Neural Computing and Applications*, 1–11. https://doi.org/10.1007/s00521-020-05626-8

Rätz, M., Javadi, A. P., Baranski, M., Finkbeiner, K., & Müller, D. (2019). Automated data-driven modeling of building energy systems via machine learning algorithms. *Energy and Buildings*, *202*, 109384. https://doi.org/10.1016/j.enbuild.2019.109384

Rios, R. A., & de Mello, R. F. (2012). A systematic literature review on decomposition approaches to estimate time series components. *INFOCOMP Journal of Computer Science*, *11*(3–4), 31–46.

Sagaert, Y. R., Aghezzaf, E.-H., Kourentzes, N., & Desmet, B. (2018). Tactical sales forecasting using a very large set of macroeconomic indicators. *European Journal of Operational Research*, *264*(2), 558–569. https://doi.org/10.1016/j.ejor.2017.06.054

Scholz-Reiter, B., Kück, M., & Lappe, D. (2014). Prediction of customer demands for production planning: Automated selection and configuration of suitable prediction methods. *CIRP Annals*, *63*(1), 417–420. https://doi.org/10.1016/j.cirp.2014.03.106

Sekma, N. C., Elleuch, A., & Dridi, N. (2016). Automated forecasting approach minimizing prediction errors of CPU availability in distributed computing systems. *International Journal of Intelligent Systems and Applications*, *8*(9), 8–21. https://doi.org/10.5815/ijisa.2016.09.02

Sergio, A. T., de Lima, T. P. F., & Ludermir, T. B. (2016). Dynamic selection of forecast combiners. *Neurocomputing*, *218*(C), 37–50. https://doi.org/10.1016/j.neucom.2016.08.072

Shahoud, S., Khalloof, H., Duepmeier, C., & Hagenmeyer, V. (2020). Incorporating unsupervised deep learning into meta learning for energy time series forecasting. In *Proceedings of the Future Technologies Conference (FTC2020)*, Vol. 1(1288), pp. 326–345. https://doi.org/10.1007/978-3-030-63128-4_25

Shaub, D. (2020). Fast and accurate yearly time series forecasting with forecast combinations. *International Journal of Forecasting*, *36*(1), 116–120. https://doi.org/10.1016/j.ijforecast.2019.03.032

Shaukat, K., Alam, T. M., Luo, S., Shabbir, S., Hameed, I. A., Li, J., Abbas, S. K., & Javed, U. (2021). A review of time-series anomaly detection techniques: A step to future perspectives. In K. Arai (Ed.), *Advances in Information and Communication, FICC 2021. Advances in Intelligent Systems and Computing* (Vol. 1363, pp. 865–877). Springer. https://doi.org/10.1007/978-3-030-73100-7_60

Shcherbakov, M., Kamaev, V., & Shcherbakova, N. (2013). Automated electric energy consumption forecasting system based on decision tree approach. *IFAC Proceedings Volumes*, *46*(9), 1027–1032. https://doi.org/10.3182/20130619-3-RU-3018.00486

Shetty, J., & Shobha, G. (2016). An ensemble of automatic algorithms for forecasting resource utilization in cloud. *2016 Future Technologies Conference (FTC)*, 301–306. https://doi.org/10.1109/FTC.2016.7821626

Silva, P. C. L., de Oliveira e Lucas, P., Sadaei, H. J., & Guimarães, F. G. (2020). Distributed evolutionary hyperparameter optimization for fuzzy time series. *IEEE Transactions on Network and Service Management*, *17*(3), 1309–1321. https://doi.org/10.1109/TNSM.2020.2980289

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222. https://doi.org/10.1023/B:STCO.0000035301.49549.88

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, *36*(1), 75–85. https://doi.org/10.1016/j.ijforecast.2019.03.017

Son, H., & Kim, C. (2015). Forecasting short-term electricity demand in residential sector based on support vector regression and fuzzy-rough feature selection with particle swarm optimization. *Procedia Engineering*, *118*, 1162–1168. https://doi.org/10.1016/j.proeng.2015.08.459

Spiliotis, E., Assimakopoulos, V., & Makridakis, S. (2020). Generalizing the theta method for automatic forecasting. *European Journal of Operational Research*, *284*(2), 550–558. https://doi.org/10.1016/j.ejor.2020.01.007

Stützle, T. (1999). *Local search algorithms for combinatorial problems: Analysis, improvements, and new applications [PhD Thesis]*. Darmstadt University of Technology, Germany.

Svetunkov, I., & Petropoulos, F. (2018). Old dog, new tricks: A modelling view of simple moving averages. *International Journal of Production Research*, *56*(18), 6034–6047. https://doi.org/10.1080/00207543.2017.1380326

Taghiyeh, S., Lengacher, D. C., & Handfield, R. B. (2020). Forecasting model selection using intermediate classification: Application to MonarchFx corporation. *Expert Systems with Applications*, *151*, 113371. https://doi.org/10.1016/j.eswa.2020.113371

Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, *39*(8), 7067–7083. https://doi.org/10.1016/j.eswa.2012.01.039

Tak, N., Eğrioğlu, E., Bas, E., & Yolcu, U. (2021). An adaptive forecast combination approach based on meta intuitionistic fuzzy functions. *Journal of Intelligent Fuzzy Systems*, *40*, 9567–9581. https://doi.org/10.3233/JIFS-202021

Tran, N., & Reed, D. A. (2004). Automatic ARIMA time series modeling for adaptive I/O prefetching. *IEEE Transactions on Parallel and Distributed Systems*, *15*(4), 362–377. https://doi.org/10.1109/TPDS.2004.1271185

Tuggener, L., Amirian, M., Rombach, K., Lorwald, S., Varlet, A., Westermann, C., & Stadelmann, T. (2019). Automated machine learning in practice: State of the art and recent results. In *Proceedings of the 2019 6th Swiss Conference on Data Science (SDS)*, pp. 31–36. https://doi.org/10.1109/SDS.2019.00-11

Valente, J. M., & Maldonado, S. (2020). SVR-FFS: A novel forward feature selection approach for high-frequency time series forecasting using support vector regression. *Expert Systems with Applications*, *160*, 113729. https://doi.org/10.1016/j.eswa.2020.113729

Villegas, M. A., & Pedregal, D. J. (2019). Automatic selection of unobserved components models for supply chain forecasting. *International Journal of Forecasting*, *35*(1), 157–169. https://doi.org/10.1016/j.ijforecast.2017.11.001

Villegas, M. A., Pedregal, D. J., & Trapero, J. R. (2018). A support vector machine for model selection in demand forecasting applications. *Computers & Industrial Engineering*, *121*, 1–7. https://doi.org/10.1016/j.cie.2018.04.042

Violos, J., Tsanakas, S., Androutsopoulou, M., Palaiokrassas, G., & Varvarigou, T. (2020). Next position prediction using LSTM neural networks. In *Proceedings of the 11th Hellenic Conference on Artificial Intelligence*, pp. 232–240. https://doi.org/10.1145/3411408.3411426

Wang, X., & Wang, C. (2020). Time series data cleaning: A survey. *IEEE Access*, *8*, 1866–1881. https://doi.org/10.1109/ACCESS.2019.2962152

Wang, X., Smith-Miles, K., & Hyndman, R. J. (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing*, *72*(10), 2581–2594. https://doi.org/10.1016/j.neucom.2008.10.017

Weber, M., Turowski, M., Çakmak, H. K., Mikut, R., Kühnapfel, U., & Hagenmeyer, V. (2021). Data-driven copy-paste imputation for energy time series. *IEEE Transactions on Smart Grid*, *12*(6), 5409–5419. https://doi.org/10.1109/TSG.2021.3101831

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, *26*(2), xiii–xxiii.

Widodo, A., & Budi, I. (2013). Feature enhancement for model selection in time series forecasting. In *Proceedings of the 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 367–373. https://doi.org/10.1109/ICACSIS.2013.6761603

Widodo, A., Budi, I., & Widjaja, B. (2016). Automatic lag selection in time series forecasting using multiple kernel learning. *International Journal of Machine Learning and Cybernetics*, *7*(1), 95–110. https://doi.org/10.1007/s13042-015-0409-7

Wu, Z., Xia, X., Xiao, L., & Liu, Y. (2020). Combined model with secondary decomposition-model selection and sample selection for multi-step wind power forecasting. *Applied Energy*, *261*, 114345. https://doi.org/10.1016/j.apenergy.2019.114345

Wythoff, B. J. (1993). Backpropagation neural networks: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, *18*(2), 115–155. https://doi.org/10.1016/0169-7439(93)80052-J

Yan, W. (2012). Toward automatic time-series forecasting using neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(7), 1028–1039. https://doi.org/10.1109/TNNLS.2012.2198074

Yang, K., & Shahabi, C. (2005). On the stationarity of multivariate time series for correlation-based data analysis. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 805–808. https://doi.org/10.1109/ICDM.2005.109

Yolcu, U., Egrioglu, E., Bas, E., Yolcu, O. C., & Dalar, A. Z. (2021). Probabilistic forecasting, linearity and nonlinearity hypothesis tests with bootstrapped linear and nonlinear artificial neural network. *Journal of Experimental & Theoretical Artificial Intelligence*, *33*(3), 383–404. https://doi.org/10.1080/0952813X.2019.1595167

Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., & Saeed, J. (2020). A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, *1*(2), 56–70. https://doi.org/10.38094/jastt1224

Zöller, M.-A., & Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. *Journal of Artificial Intelligence Research*, *70*, 409–472. https://doi.org/10.1613/jair.1.11854

Züfle, M., Bauer, A., Lesch, V., Krupitzer, C., Herbst, N., Kounev, S., & Curtef, V. (2019). Autonomic forecasting method selection: Examination and ways ahead. In *Proceedings of the 2019 IEEE International Conference on Autonomic Computing (ICAC)*, pp. 167–176. https://doi.org/10.1109/ICAC.2019.00028

Züfle, M., & Kounev, S. (2020). A framework for time series preprocessing and history-based forecasting method recommendation. In *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems*, pp. 141–144. https://doi.org/10.15439/2020F101

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Meisenbacher, S., Turowski, M., Phipps, K., Rätz, M., Müller, D., Hagenmeyer, V., & Mikut, R. (2022). Review of automated time series forecasting pipelines. *WIREs Data Mining and Knowledge Discovery*, e1475. https://doi.org/10.1002/widm.1475