Information Sciences 484 (2019) 302–337

Contents lists available at ScienceDirect

Information Sciences

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

# Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model

Antonio Rafael Sabino Parmezan[a,*], Vinicius M.A. Souza[a], Gustavo E.A.P.A. Batista[a]

*Laboratory of Computational Intelligence, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Avenue Trabalhador São-carlense, 400, São Carlos 13566-590, SP, Brazil*

ARTICLE INFO

*Article history:*
Received 2 June 2018
Revised 28 January 2019
Accepted 30 January 2019
Available online 30 January 2019

*Keywords:*
Univariate analysis
Automatic parameter tuning
Multi-step-ahead prediction
Time series forecasting
Data mining

ABSTRACT

The choice of the most promising algorithm to model and predict a particular phenomenon is one of the most prominent activities of the temporal data forecasting. Forecasting (or prediction), similarly to other data mining tasks, uses empirical evidence to select the most suitable model for a problem at hand since no modeling method can be considered as the best. However, according to our systematic literature review of the last decade, few scientific publications rigorously expose the benefits and limitations of the most popular algorithms for time series prediction. At the same time, there is a limited performance record of these models when applied to complex and highly nonlinear data. In this paper, we present one of the most extensive, impartial and comprehensible experimental evaluations ever done in the time series prediction field. From 95 datasets, we evaluate eleven predictors, seven parametric and four non-parametric, employing two multi-step-ahead projection strategies and four performance evaluation measures. We report many lessons learned and recommendations concerning the advantages, drawbacks, and the best conditions for the use of each model. The results show that SARIMA is the only statistical method able to outperform, but without a statistical difference, the following machine learning algorithms: ANN, SVM, and *k*NN-TSPI. However, such forecasting accuracy comes at the expense of a larger number of parameters. The evaluated datasets, as well detailed results achieved by different indexes as MSE, Theil's U coefficient, POCID, and a recently-proposed multi-criteria performance measure are available online in our repository. Such repository is another contribution of this paper since other researchers can replicate our results and evaluate their methods more rigorously. The findings of this study will impact further research on this topic since they provide a broad insight into models selection, parameters setting, evaluation measures, and experimental setup.



## 1. Introduction

The technological advances in computing area, including databases systems, machine learning, and cloud computing have leveraged off the conversion of data into useful information and knowledge to support decision making. These advances

* Corresponding author.
*E-mail addresses:* parmezan@usp.br (A.R.S. Parmezan), vmasouza@icmc.usp.br (V.M.A. Souza), gbatista@icmc.usp.br (G.E.A.P.A. Batista).

https://doi.org/10.1016/j.ins.2019.01.076
0020-0255/

contributed to the development of computer systems capable of storing, analyzing and managing an increasing amount of data.

Data can be represented in different formats, from the most basic, such as numeric and nominal, to more complex ones, such as audio and video. However, the storage of temporal information, which allows the chronological organization of the collected data, is one of the data representations that has attracted most attention of researchers and driven the creation of large databases [14].

Temporal data mining is a process to extract useful knowledge from time series. Prediction[1] is one of the tasks contemplated by temporal data mining, which is motivated by the challenge of reducing future uncertainty, especially due to the volatility of some phenomena [5]. Some examples are the prediction of the closing price of a company's stock each day, prediction of unemployment for a state each quarter, and energy load forecasting. Thus, predictors constitute an essential tool in many areas for decision making as commercial and financial strategies or politics, and their adequate use can cause social and economic impacts.

The methods for time series prediction rely on the idea that historical data include intrinsic patterns which convey useful information for the future description of the phenomenon investigated. These patterns are usually non-trivial to identify, and their discovery is one of the primary goals of the time series processing: the circumstances the patterns found will repeat and what types of changes they may suffer over time [31].

The design of a model for time series prediction focuses on the application of algorithms. Under certain assumptions about the data, the model captures the variables involved and represents the existing dynamic relations, summarizing them in a robust and potentially flexible mathematical structure. The structure can be used to predict future data, as well as to help to understand the process that originated the data.

The application of statistical methods based on autoregression and Moving Averages (MA) are considered the state-of-the-art for time series modeling and prediction for over a half-century [16]. The algorithms that implement these methods assume the data follow a known distribution. Based on that information, function parameters are defined to fit a model to the data. However, the usage of these parametric methods requires sophisticated mathematical concepts as well as vast technical expertise for the establishment of the model's parameters.

The use of fundamentals of descriptive statistics can guide the parameters definition [5]. In several cases, autocorrelation-based functions automatize this task. The results can be interpreted via correlograms and by the application of techniques for obtaining input arguments from the minimization of information criteria, which penalizes the model for the number of parameters required for its adjustment.

More recently, several studies have employed non-parametric modeling based on machine learning algorithms for time series prediction [33,40]. One of the main advantages of this approach is that it does not presuppose the data distribution nature.

Empirical research has demonstrated that machine learning algorithms for time series prediction provide very competitive results, frequently outperforming statistical models [9,25,37,50]. Nevertheless, to the best of our knowledge and according to the systematic review conducted in this work, there are no articles in the literature containing robust empirical studies focused on the comparison between parametric and non-parametric methods. Thorough research regarding the application of these algorithms on benchmark datasets and in the presence of properties that challenge time series modeling, such as trend, non-stationarity and distribution shifts, could consolidate the efficiency and effectiveness of each method.

The present study fills this gap by providing an objective comparison of the most popular statistical and machine learning models for univariate time series prediction. The primary purpose of this research is to empirically identify the behavior of these predictors concerning datasets with specific characteristics. With this in mind, we provide a comprehensive guide for users to choose the best predictive models for their applications. The three major contributions of this paper can be summarized as follows:

- Planning and conduction of a systematic review and meta-analysis to position this research in the corresponding state-of-the-art. We have selected 117 publications, between the years 2009 and 2018, whose content helped to answer ten scientific questions;
- A critical analysis and algorithms recommendation based on the execution of one of the most extensive, impartial and comprehensible experimental evaluations ever done for time series prediction. Using a set of 95 time series from synthetic and real domains, we face eleven predictive algorithms, seven parametric and four non-parametric, employing two multi-step-ahead projection strategies and four performance evaluation measures. The findings of our experimental comparison will facilitate further research on this topic since they provide a better insight into the predictive performance of the methods currently available in the literature. In addition to discussing which algorithms should be used as baseline and topline when investigating the proposition of novel models, we highlight the advantages and drawbacks of each method. We performed this analysis according to characteristics of the data, and we hope that it can guide practitioners in statistics and machine learning;
- Building an online archive, namely of ICMC-USP Time Series Prediction Repository [32], which grants access to all materials produced in this work. Thereby, other researchers can reuse the available datasets to replicate our results and compare their methods more rigorously against different predictors.

[1] The terms "prediction" and "forecast" are used interchangeably in this paper.

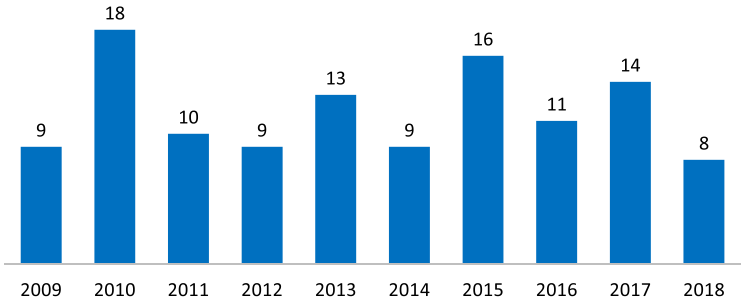



**Fig. 1.** Number of papers published by year.

We emphasize that the outlined experimental protocol and our discussion on its usage are a guideline for model selection, parameters setting, and employment of statistical and machine learning algorithms for time series prediction.

The remaining of this paper is structured as follows: Section 2 presents the related work and the statistics derived from a meta-analysis of the literature, which was guided by the results of a systematic review. Section 3 reports the main definitions and notations about time series. Section 4 describes the temporal data prediction approaches with their usual methods and some techniques that help in its parameters estimation. Section 5 specifies the configuration of the experiments, which includes datasets, algorithms and performance measures. Section 6 presents results and discussion, while Section 7 punctuates the limitations, recommendations, and practical implications of the outcomes. Finally, Section 8 reports the conclusions and directions for future work.

## 2. Systematic review and meta-analysis of the literature

For over a half-century, statistical methods based on autoregression and MA have influenced the temporal data processing and analysis fields. Although some studies have stated, between the seventies and eighties, that the parametric models could not be readily adapted to many real applications, they resisted over the years [16]. The preference for these methods made them reached the condition of state-of-art for time series modeling and prediction.

In the last two decades, with the rise of the data mining process, there is an increasing interest in the adaptation of machine learning methods, especially those for regression tasks, to support analysis with time dependence. Due to their simplicity and comprehensibility, the non-parametric techniques have established themselves as serious candidates to the classical models, so that scientific competitions have been undertaken to encourage both the improvement of these algorithms as the development of new solutions [1].

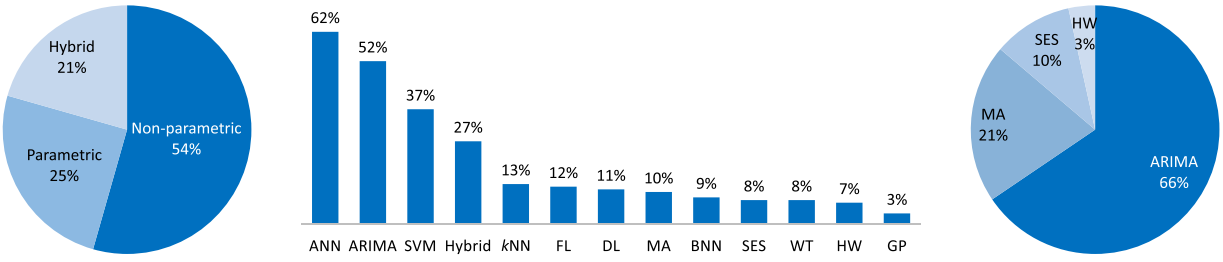
The researchers from statistics and machine learning communities have contributed to several aspects of the prediction process, such as the assistance in selecting the most promising model [26], the study of the deseasonalization effects on the projection of future values [3], and the construction of hybrid models by a combination of statistical and machine learning methods [2]. In this sense, the quality of parametric and non-parametric models has been explored mainly in annual or biennial competitions aimed at assessing the performance of prediction algorithms on a considerable amount of time series data [3].

To position this study in the corresponding state-of-the-art, we performed a meta-analysis of the literature. We conducted this meta-analysis using the results of a systematic review of papers published in the last ten years. Our supplementary material details the systematic review protocol, which includes ten research questions, a search key, and five search sources as well as the selected publications and a detailed results interpretation. Throughout the remainder of this section, we highlight the most relevant points found and describe the most related papers to the present work.

Fig. 1 displays a distribution bar chart, by publication year, regarding the number of selected papers at the end of the systematic review. In particular, the graph shows that from January 2009 to December 2018, were published annually about twelve articles, with two peaks in 2010 and 2015.

The inspection of the 117 papers discovered by the systematic review allowed us to elaborate a broad and in-depth meta-analysis, of which Fig. 2 illustrates some results. Fig. 2(a) shows the percentage of use of each prediction approach in 68 papers with real applications. Fig. 2(b) portrays the frequency in which the most popular methods appeared in the 117 publications. Fig. 2(c) graphically summarizes the algorithms most used as baselines in 29 empirical studies involving both statistical and machine learning methods.

Table 1 compares our study with other 11 experimental papers previously published. This comparison uses the following criteria: the number of parametric and non-parametric predictors compared; the amount of synthetic and real time series selected; the number of measures chosen to evaluate the algorithms' performance; and if any statistical significance test was applied to support the comparison of results. Most of these articles assess predictors on datasets from specific domains. Although [1] considered a significant number of datasets, the lengths of the series are short with values between 81 and 126 observations. In general, such studies have concluded that no particular traditional model can provide the best predictions. The supplementary material presents a description of these papers.




(a) Approaches (b) Methods (c) Baseline methods

**Fig. 2.** Summary of the meta-analysis results. The acronyms are: Artificial Neural Networks (ANN), ARIMA models – Autoregressive Integrated Moving Average (ARIMA) or Seasonal ARIMA (SARIMA) –, Support Vector Machines (SVM), *k*-Nearest Neighbors (*k*NN), Fuzzy Logic (FL), Deep Learning (DL), Bayesian Neural Networks (BNN), Simple Exponential Smoothing (SES), Wavelet Transform (WT), Holt-Winters (HW) models, and Gaussian Process (GP).

**Table 1**
Some properties of empirical studies reported in related work.

| Paper | #Predictor(s) | | #Synthetic | #Real | #Performance | Statistical |
|---|---|---|---|---|---|---|
| | Parametric | Non-parametric | Dataset(s) | Dataset(s) | Measure(s) | Test(s) |
| [33] | – | 3 | – | 55 | 3 | ✓ |
| [50] | 2 | 2 | – | 9 | 3 | |
| [37] | 1 | 7 | 5 | 35 | 1 | ✓ |
| [25] | 1 | 2 | – | 6 | 1 | |
| [9] | 1 | 2 | 1 | 7 | 2 | ✓ |
| [1] | – | 8 | – | 1045 | 2 | ✓ |
| [6] | 5 | 3 | 1 | 1 | 1 | ✓ |
| [44] | 5 | – | – | 5 | 3 | |
| [8] | 2 | 1 | – | 10 | 1 | ✓ |
| [49] | 3 | 1 | – | 1 | 4 | |
| [18] | 6 | 3 | – | 10 | 2 | ✓ |
| This paper | 7 | 4 | 40 | 55 | 4 | ✓ |

As summarized in the last row of Table 1, our proposal differs from the literature not only by comparing eleven popular predictive algorithms using two multi-step-ahead strategies on 95 time series, but also includes a more rigorous experimental design supported by a new multi-criteria performance measure. Besides discussing which algorithms should be used as baseline and topline when investigating the proposition of novel models, we highlighted the advantages and drawbacks of each method for certain types of datasets. We compare the models' performances with the support of statistical significance tests.

## 3. Fundamentals of time series

A time series $Z$ of size $m$ can be formulated as an ordered sequence of observations, *i.e.*, $Z = (z_1, z_2, \ldots, z_m)$ where $z_t \in \Re$ is an observation at time $t$. In this work, we assume the time series are discrete and uniformly sampled over time.

When the time series values are synthesized by a mathematical function $y = f(time)$, the series is said deterministic. When the time series comprises, in addition to a mathematical time function, a random term $\epsilon$, $y = f(time, \epsilon)$, the series is stochastic or nondeterministic.

Another relevant feature of a time series is the stationarity. A stationary series develops randomly around a constant average, reflecting some stable equilibrium [31]. Attention to this property is essential since some methods assume the stationarity condition. A typical transformation takes successive differences of the original time series. The first difference, defined as $\Delta z_t = z_t - z_{t-1}$ is usually enough to make the series stationary.

Fig. 3(a) displays a real time series that expounds, in tons, the monthly chocolate production in Australia from January 1958 to December 1990. These measurements, provided by the Australian Bureau of Statistics, as well as all datasets adopted in this paper, are available at the ICMC-USP Time Series Prediction Repository [32].

Fig. 3(a) shows that the time series values do not oscillate around a fixed level. Instead, there exists an increasing behavior whose the period of variation remains constant as the level increases. We can avail of temporal data decomposition techniques to explore those properties. The decomposition techniques, beyond allowing the identification of components that act in a series, enable to obtain patterns (via indexes and equations) that may be coupled to a computational model for prediction of future values. The three major components of a time series are:

**Trend** ($T$) is a long-term increase or decrease in the data which can assume a great variety of patterns (*e.g.*, linear, exponential, damped, and polynomial) [31]. Real time series with an increasing trend can be found in phenomena related to the demographic development, gradual change of consumption habits, and demand for technologies in

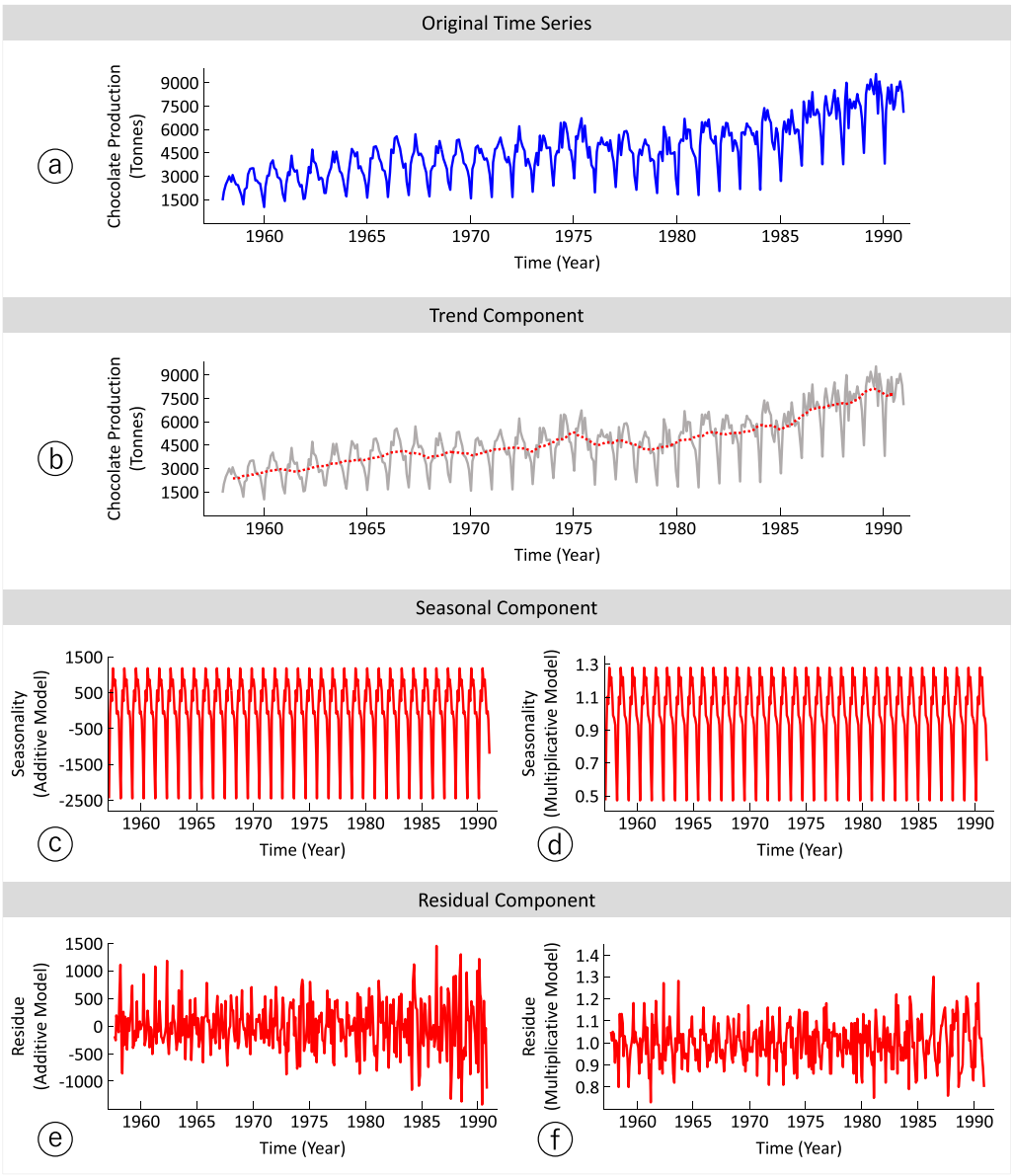



**Fig. 3.** Components extracted from the monthly chocolate production time series.

the social sectors. The decreasing trend, in turn, can be found in series concerning the mortality rates, epidemics, and unemployment. We can use regression models and the MA method to obtain this component [31,41]. We use the trend to estimate the level, *i.e.*, the value or the typical range of values that the variable assumes if there is no increasing or decreasing trend in the long term. Fig. 3(b) shows the trend, estimated using MA with 12 periods (monthly data), of the chocolate production time series;

**Seasonality** ($S$) is the occurrence of cyclic patterns of variation that repeat, at relatively constant time intervals, along with the trend component. Examples of seasonal patterns are the increase in sales of air conditioners in summer and warm clothing in winter. Average percentage, percentage relation, relation between MA, and relative links are some of the algorithms that allow computing the seasonality from the extraction of seasonal indexes [41];

**Residue** ($R$) is the short-term fluctuations that are neither systematic nor predictable. In the real world, unforeseen events cause such instabilities, such as natural disasters, terrorist attacks, and strikes. In practical terms, the residual component is what remains after the estimation of $T$ and $S$ components, and their removals from the time series [31]. We can define the residue $R$ of a period $t$ of a series $Z$, in agreement with the additive and multiplicative models, as $R_t = Z_t - (T_t + S_t)$ and $R_t = Z_t \div (T_t \times S_t)$, respectively.

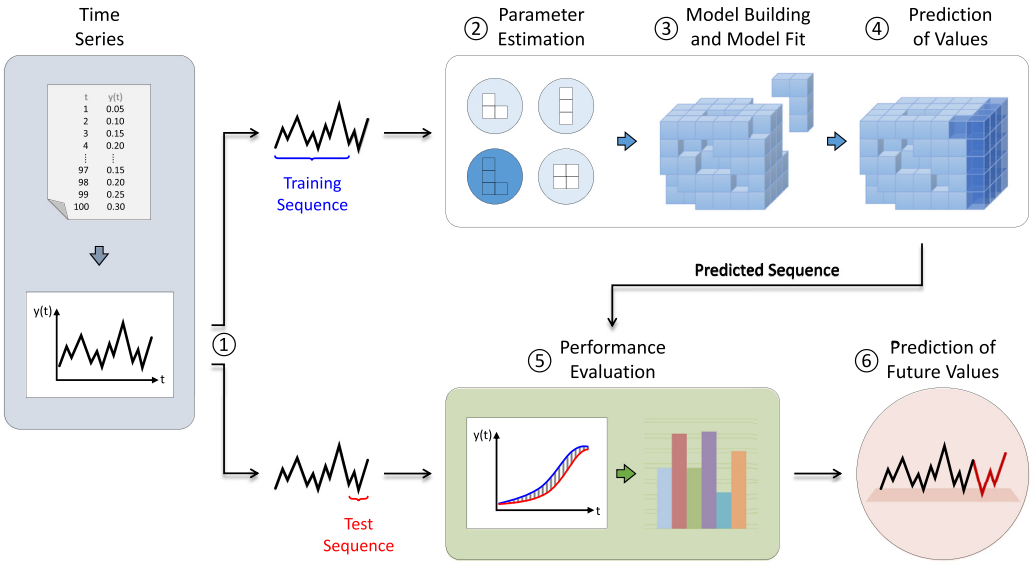



**Fig. 4.** Time series prediction process.

For a better analysis and understanding, we can reformulate the time series $Z$, according to Eqs. (1) and (2), by an additive or multiplicative decomposition of its components.

$$Z_t = T_t + S_t + R_t \tag{1}$$

$$Z_t = T_t \times S_t \times R_t \tag{2}$$

In the additive model (Eq. (1)), the interest variable value is the sum of the components values, which contemplate the same unit of the observation $Z_t$. Considering the additive decomposition, Fig. 3 – (c) and (e) – shows the seasonality provided by the relation between MA, and the residue of the chocolate production time series.

Differently, in the multiplicative model (Eq. (2)), only the trend has the same unit of the investigated variable. The other components exhibit values that can modify the trend, *i.e.*, they assume values larger, smaller or exactly equal to 1. Adopting the multiplicative decomposition, Fig. 3 – (d) and (f) – outlines the seasonality, obtained via relation between MA, and the residue of the chocolate production time series.

We note that not every data sequence will have all the three mentioned components, even when the classical decomposition is considered.

## 4. Time series prediction

The time series prediction process covers six steps, as illustrated in Fig. 4.

The first step partitions the time series in two sequences: one before the prediction horizon, which is intended to the model training (building and fit); and another after that period, which is used to test (evaluate) the quality of the fitted model.

The second step chooses the predictive model structure based on data characteristics and estimates the parameters using some search technique. Usually, the algorithm that implements this technique receives as input the training sequence, which is subdivided into subsequences (samples) for training and validation, and a set of predefined parameters. At each iteration, the algorithm seeks for parameters values that minimize the predictive error of the model.

The third step builds the model with the previously found parameters values and fits the training sequence data. This model is then extrapolated, in the fourth step, for the periods of the test sequence. Evidently, the prediction error of the model reflects the chosen values for the parameters. Such error can be amplified for long time horizons.

The fourth step also chooses the strategy to predict the values of a time series several periods ahead (prediction horizon $h > 1$). The most intuitive strategy is known as multi-step (or recursive), where the prediction of $h > 1$ is conducted $h$ successive times considering a predictive model with $h = 1$ [3]. After the model's extrapolation, the predicted value or the respective actual value can be employed to calculate the next prediction. In this paper, when the predicted values are used, we called the strategy of multi-step-ahead with approximate iteration. Otherwise, when the actual values are adopted, the strategy is called multi-step-ahead with updated iteration.

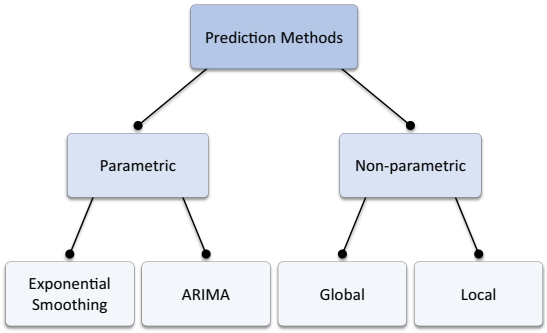



**Fig. 5.** Hierarchy of approaches for time series prediction.

The fifth step compares the predicted values to the test sequence to measure the model's accuracy. The performance analysis is essential given that distinct models may have similar adjustments, but result in significantly different predictive values.

The sixth step makes predictions for future periods of the time series. This step should monitor the prediction error as soon as the actual values of the series arrive. This monitoring aims to indicate when it is necessary to update the model with new data or readjust its parameters since the distribution of most recent data is distinct from old data.

The time series prediction methods have evolved over the years passing from simple regression techniques to robust statistical and artificial intelligence algorithms. We can group the methods into two approaches according to the prior knowledge about the data distribution (Fig. 5): (i) parametric (exponential smoothing or based on autoregression and MA), and (ii) non-parametric (global or local). The following subsections discuss these approaches along with the most renowned algorithms for time series prediction.

### *4.1. Parametric methods*

The statistical methods require *a priori* knowledge about the data distribution to build predictive models. This assumption makes the model depends on a set of parameters, which must be determined to optimize the prediction results. We can divide the models into two groups according to their mathematical complexity [31]: (i) exponential smoothing models and (ii) ARIMA models.

The exponential smoothing models decompose the time series into components whose values are smoothed by weights that exponentially decay over time. In the end, an additive or multiplicative structure recomposes the smoothed components to predict future values [15].

In contrast, ARIMA models involve three statistical procedures [5]: (i) autoregression, (ii) integration, and (iii) MA. Autoregression expresses the correlation between observations, *i.e.*, how much the current values influences the next ones. The integration procedure indicates the number of differences required to guarantee the stationarity of the series. Lastly, the MA part comprises unknown factors that cannot be explained by the time series past values. The SARIMA generalization also allows to model temporal data with seasonal variations.

#### *4.1.1. Moving averages*

The MA model of order $r$, MA($r$), is a simple technique that performs an arithmetic average of the last $r$ values of time series to predict the next value. MA considers a constant number of observations to exploit the autocorrelation structure of the prediction residues at the current time with those occurred in previous periods. Eq. (3) defines the MA model, where $r$ is the number of observations included in the average $z_{t+1}$.

$$z_{t+1} = \frac{z_t + z_{t-1} + z_{t-2} + \ldots + z_{t-r+1}}{r} \tag{3}$$

The higher the value of $r$, the more uniform (smoothed) will be the predicted data behavior. Thus, when the series shows small distortions in their patterns or random fluctuations, it is recommended to employ a substantial $r$ value to avoid the influence of noise in the predictions. Otherwise, if the series is nearly devoid of randomness and presents a significant shift in the curves inflection points, it is indicated to use a smaller $r$ value to allow the model to react quickly to the data changes.

The drawbacks of the MA model are their low accuracy to deal with trend and seasonality. Since the prediction of the next value always involves the addition of new data and the discard of the previous one. Furthermore, the weights assigned to the $r$ observations are typically all equal. Therefore, there is no emphasis on the most recent observations [29].

#### *4.1.2. Simple exponential smoothing*

The SES method is equivalent to MA with $r = m$, except by the fact that each series value receives a different weight. The weights increase exponentially over time so that the most recent observations exert more influence on the calculation of future predictions [23].

Eq. (4) expresses the SES model structure [15], where $L$ denotes the level at time $t$, $\alpha$ $(0 < \alpha < 1)$ is the weight (or constant smoothing) assigned to historical values, and $z_t$ corresponds to the last observed value.

$$L_t = \alpha z_t + \alpha(1-\alpha)z_{t-1} + \ldots + \alpha(1-\alpha)^{m-1}z_1 \tag{4}$$

To avoid an expensive computation that involves all observations for each new estimate $L$, we can reduce Eq. (4) in function of the current value of the time series and the level computed in the previous time. Eq. (5) formalizes the result of this simplification [15].

$$L_t = \alpha z_t + (1-\alpha)L_{t-1} \tag{5}$$

Usually, at the beginning of the SES prediction process, it is supposed that the first fitted value is equal to the first series value, *i.e.*, $L_1 = z_1$. In this case, the adjustment procedure starts from the second observation of the time series. The exponential smoothing of the last observed value ($z_{m+1} = L_m$) gives the prediction at time $m+1$. We call this strategy one-step-ahead. This method does not support an extension to larger horizons. In this case, the fitted value $L_m$ gives the prediction of all future values for multiple horizons.

The flexibility, mathematical simplicity, and reasonable accuracy explain the popularity conferred to the SES method. For making a new prediction, the algorithm needs the most recent observation, the last predicted value, and the parameter $\alpha$. Among the drawbacks of the method, stands the difficulty in finding the most appropriate value for the smoothing constant [23].

#### 4.1.3. Holt's exponential smoothing

The SES model when applied to temporal data that present increasing (or decreasing) linear behavior, provides predictions which underestimate (or overestimate) the actual values. To avoid this systematic error, we can make use of methods as Holt's Exponential Smoothing (HES) [15].

The HES model structure, defined by Eqs. (6) and (7) [22], is similar to the SES method. However, besides to use the parameter $\alpha$ to soften the level component, the algorithm uses a second smoothing constant ($\beta$) for modeling the time series trend.

$$L_t = \alpha z_t + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{6}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{7}$$

$$z_{t+h} = L_t + hT_t \tag{8}$$

The smoothing constants values $\alpha$ and $\beta$ lie in the range [0, 1], and Eqs. (6) and (7) estimate the level and trend components, respectively. These equations, as well any exponential smoothing method, modify previous estimates when a new observation is computed. Moreover, in Eq. (8), $z_{t+h}$ indicates the prediction value of $z$ at time $t+h$, where $h$ represents the prediction horizon.

To implement the HES algorithm recurrence relation, we need to provide its initial values. A widely accepted rule in the literature is to assume $L_1 = z_1$ and $T_1 = z_2 - z_1$. As the method is based on the self-learning concept, the initial values do not affect the predictions. However, this fact does not apply to the smoothing constants, which are difficult to set and bad choices may degrade the predictive performance of the algorithm.

#### 4.1.4. Holt-Winters' seasonal exponential smoothing

The HW models structure comprises three equations with distinct smoothing constants, which are related to the time series primary components. Given this design outline, such models are divided into two groups [47]: (i) Multiplicative and (ii) Additive. The choice of a structure depends on the seasonal pattern of the investigated series.

We can use the Multiplicative model of HW (MHW) to adjust time series in which the amplitude of the seasonal variation rises with the increase of the series average level. The algorithm employs the following equations [47]:

$$L_t = \alpha\frac{z_t}{S_{t-s}} + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{9}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{10}$$

$$S_t = \gamma\frac{z_t}{L_t} + (1-\gamma)S_{t-s} \tag{11}$$

$$z_{t+h} = (L_t + hT_t)S_{t-s+h} \tag{12}$$

In these equations, $\alpha$, $\beta$ and $\gamma$ are smoothing constants whose values lie in the range [0, 1], $s$ indicates the number of observations that make up a seasonal variation, and $z_{t+h}$ represents the prediction value $z$ for the period $t+h$.

Analogously to SES and HES methods, the MHW algorithm receives as input a time series and recursively applies the three described equations. Such application should be started at some time in the past, where the values of $L$, $T$ and $S$ was previously estimated. A simple way to obtain this approximation is through the initialization of level and trend in the same period $s$. Thus, the level can be determined from the average of the first station (Eq. (13)).

$$L_s = \frac{1}{s}(z_1 + z_2 + \ldots + z_s) \tag{13}$$

The trend can be initialized using two complete stations, as defined by Eq. (14).

$$T_s = \frac{1}{s}\left(\frac{z_{s+1} - z_1}{s} + \frac{z_{s+2} - z_2}{s} + \ldots + \frac{z_{s+s} - z_s}{s}\right) \tag{14}$$

The initial seasonal indexes can be computed by the ratio between the first observations and the average of the first period, as shown in Eq. (15).

$$S_1 = \frac{z_1}{L_s}, S_2 = \frac{z_2}{L_s}, \ldots, S_s = \frac{z_s}{L_s} \tag{15}$$

The Additive model of HW (AHW) has greater explanatory power in series where the difference between the highest and the lowest demand value within the stations remains constant over time. The algorithm that implements this model uses the following equations:

$$L_t = \alpha(z_t - S_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{16}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{17}$$

$$S_t = \gamma(z_t - L_t) + (1-\gamma)S_{t-s} \tag{18}$$

$$z_{t+h} = L_t + hT_t + S_{t-s+h} \tag{19}$$

Eq. (10) of MHW is identical to Eq. (17) of AHW. The difference lies in the use of the other equations, in which the seasonal indexes are summed and subtracted, rather than multiplied and divided as in the multiplicative model.

The variables $L$ and $T$ are commonly initialized by applying the same equations of MHW. The seasonal indexes are calculated according to Eq. (20).

$$S_1 = z_1 - L_s, S_2 = z_2 - L_s, \ldots, S_s = z_s - L_s \tag{20}$$

The correct choice of HW models is associated with the morphology of the seasonal variations in the time series, regardless of the existence of the trend component. In these models, when $\gamma = 0$ does not mean that the series is devoid of seasonality, but that seasonal rates have been initialized with values that do not need to be fixed along the prediction.

#### 4.1.5. ARIMA and SARIMA models

The ARIMA models of order (*p, d, q*), *i.e.* ARIMA(*p, d, q*), result from the combination of three procedures [5]: (i) Autoregression (AR($p$)), (ii) integration[2], and (iii) Moving Averages (MA($q$)). The simultaneous use of these three components is not a rule to model time series with absence of seasonal patterns, once they can be executed in a conjugated way, *i.e.*, one complementing the other. By this perspective and as the integration procedure can be performed in a preprocessing step, the ARIMA nomenclature is also used to refer to the following structures: $ARIMA(p,0,0) = AR(p)$; $ARIMA(0,0,q) = MA(q)$; and $ARIMA(p,0,q) = ARMA(p,q)$.

The main benefit of ARMA is that, to adjust its structure to complex stationary time series, it often uses a number of terms lower than required by models purely AR or purely MA. When a time series is non-stationary, it can be transformed using a data differentiation procedure which ensures such property. This procedure when added to the ARMA structure results in the ARIMA model with order (*p, d, q*), ARIMA(*p, d, q*), defined by Eq. (21).

$$I'_t = \delta + \sum_{i=1}^{p} \phi_i I'_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i} + e_t \tag{21}$$

In Eq. (21), $I'_t = \Delta^d z_t = \Delta(\Delta^{d-1} z_t)$ and $d$ indicates the difference operator degree; $\phi_p$ and $\theta_q$ are, in this order, the parameters of the procedures: autoregressive, with lag length $p$, and MA, with lag length $q$; $\delta$ reflects the initial level of the

[2] An operation which consists of taking successive differences from the original series $Z$. The first difference is denoted by $\Delta z_t = z_t - z_{t-1}$; the second difference is defined as $\Delta^2 z_t = \Delta(\Delta z_t) = \Delta(z_t - z_{t-1})$; finally, the $d$th difference equals $\Delta^d z_t = \Delta(\Delta^{d-1} z_t)$.

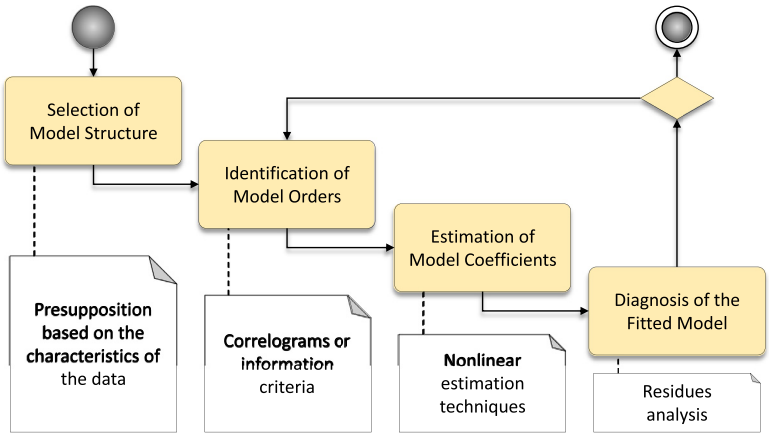



**Fig. 6.** Activity flow diagram for building an ARIMA or SARIMA model.

model (performs the same function as the intercept in linear regression) and is calculated according to Eq. (22), where $\mu$ represents the stationary process average; and $e_t$ is the white noise in a distribution with zero average and constant variance $\sigma_e^2$. Besides, for each time instant $t$, it is assumed that $e_t$ is independent of the time series past values $(z_{t-1}, z_{t-2}, \ldots, z_{t-m+1})$.

$$\delta = \mu(1 - \phi_1 - \phi_2 - \cdots - \phi_p) \tag{22}$$

The constant $\delta$ may be omitted in the ARIMA model when the time series under study is non-stationary by nature and, consequently, it was differentiated for obtaining stationarity ($d > 1$). If the series is stationary in its original form ($d = 0$), but not with zero average and unit standard deviation, the constant is required. In addition, when the model is devoid of the autoregressive part (AR($p$)), it is assumed that the constant is equal to the time series average ($\delta = \mu$) [31].

ARIMA can model homogeneous non-stationary series, *i.e.*, time series with a non-explosive trend, as well as stationary series. This model exploits the autocorrelation between the time series values at successive instants, but when the data are observed in periods of less than one year, the series may also have autocorrelation for a seasonal station $s$. In this context, the seasonal ARIMA models, also known as SARIMA, have in its structure a non-seasonal part (Eq. (21)), with parameters ($p$, $d$, $q$), and a seasonal part (Eq. (23)), with parameters $(P, D, Q)_s$.

$$I_t'' = \delta + \sum_{i=1}^{P} \Phi_{is} I_{t-is}'' + \sum_{i=1}^{Q} \Theta_{is} e_{t-is} + e_t \tag{23}$$

In Eq. (23), $I_t'' = \Delta^D z_t = \Delta(\Delta^{D-1} z_t)$ and $D$ indicates the degree of the seasonal difference operator; the constant $\delta$ is computed according to Eq. (24) and its use follows the same rules as those imposed on the ARIMA structure, but now considering $D$; $\Phi_P$ e $\Theta_Q$ are, in this order, the parameters of the procedures seasonal autoregressive, with lag length $P$, and of the seasonal MA, with lag length $Q$; and $e_t$ is the white noise that cannot be explained by the model.

$$\delta = \mu(1 - \Phi_1 - \Phi_2 - \cdots - \Phi_p) \tag{24}$$

The SARIMA$(p, d, q) \times (P, D, Q)_s$ is denoted by Eq. (25), where the non-seasonal and seasonal parts are summed.

$$I_t = \delta + \sum_{i=1}^{p} \phi_i I_{t-i}' + \sum_{i=1}^{q} \theta_i e_{t-i} + \sum_{i=1}^{P} \Phi_{is} I_{t-is}'' + \sum_{i=1}^{Q} \Theta_{is} e_{t-is} + e_t \tag{25}$$

In SARIMA models, $\delta$ is calculated by Eq. (26) and it can be omitted when $d + D > 1$. Otherwise, if $d + D \leq 1$, the use of $\delta$ is required. When the model is devoid of the autoregressive and seasonal autoregressive parts, we must assume that the constant is equal to the time series average ($\delta = \mu$) [31].

$$\delta = \mu(1 - \phi_1 - \phi_2 - \cdots - \phi_p)(1 - \Phi_1 - \Phi_2 - \cdots - \Phi_p) \tag{26}$$

The application of SARIMA is especially appropriate when the data have seasonal variations that are not adequately addressed by the first difference ($\Delta z_t = z_t - z_{t-1}$). A typical example is a time series that describes monthly data. In this series, a dependency between the observations $z_t$ and $z_{t-12}$ is likely to be found.

The design of an ARIMA or SARIMA model is guided based on the iterative cycle of Box-Jenkins [5]. This method enables the identification of the stochastic process of data generation and related parameters. The four steps of the iterative cycle are outlined in Fig. 6 and detailed as follow.

1. Selection of model structure: The model structure is chosen based on the data characteristics. Therefore, when the series comprises trend, it is recommended to use an ARIMA structure. On the other hand, if the series presents both components of trend and seasonally, it is suggested to adopt a SARIMA structure;

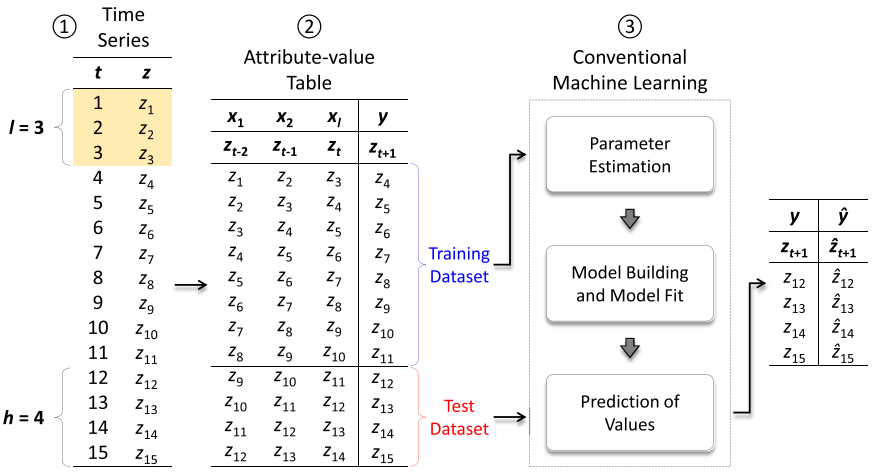



**Fig. 7.** Exemplification of global approach for time series prediction.

2. Identification of model orders: The values of $p$, $d$, and $q$ of ARIMA($p$, $d$, $q$), or the values of $p$, $d$, $q$, $P$, $D$, and $Q$ of SARIMA$(p, d, q) \times (P, D, Q)_s$, are set with the aid of correlograms or information criteria. Initially, iterative mechanism counts the number of integrations ($d$ or $D$), in which the data sequence is differentiated as many times as necessary until its variance is less than the variance computed for its original version (without differentiation). Afterward, the analyst must inspect the functions of sample autocorrelation and sample partial autocorrelation of the differentiated time series in the lags $1, 2, 3, \ldots$ to obtain the value of $p$ and $q$, and in the lags $s, s \times 2, s \times 3, \ldots$ to obtain the value of $P$ and $Q$. An alternative way to find these parameters values lies in the application of information criteria. The Akaike Information Criterion (AIC), expressed by Eq. (27) [31], penalizes the adjustment quality of models with many parameters.

$$AIC = -2 \times LL + (\log(n) + 1) \times NP \tag{27}$$

where $LL$ is the likelihood function logarithm, $n$ refers to the number of observations of the training series, and $NP$ comprises the number of parameters. The idea is to select the model that achieves the lower AIC. Obviously, a model with more parameters may have a better fit, but not necessarily will be preferred regarding AIC;
3. Estimation of model coefficients: The preliminary estimates of $\phi_p$ and $\Phi_P$, of the autoregressive component, and $\theta_q$ and $\Theta_Q$, of the MA component, can be obtained using the autocorrelations of the time series integrated into the model identification step. After the assignment of initial values, the coefficients are estimated by maximizing the likelihood function. As the least squares estimators can approximate the maximum likelihood estimators, the said function is typically maximized by nonlinear least squares using the Levenberg-Marquardt algorithm;
4. Diagnosis of the fitted model: The model identified as the most promising is examined to ensure that the data dynamic was satisfactorily represented. In practice, the estimates of errors (residues) are analyzed by residual autocorrelation tests. These tests are intended to verify if the residues present white noise behavior, *i.e.* if their autocorrelations behave randomly and are not significant. In the affirmative case, we can extrapolate the model to future times. In the negative case, it will be necessary to select another model and repeat the identification, estimation, and diagnosis steps.

The employment of ARIMA models requires expertise both in the application domain and in the computational mathematics. Moreover, the analyst perception and experience are essential for the modeling process becomes more practical and less expensive.

### *4.2. Non-parametric methods*

Machine learning prediction methods, as opposed to statistical models, describe the data properties without the prior knowledge of their distribution. Because they do not depend explicitly on parameters to model the phenomenon's behavior, these methods are simpler to adjust and show reliable performance even when applied to complex and highly nonlinear series. We can divide the machine learning predictors into two approaches [24]: (i) global and (ii) local.

In the global approach, the machine learning methods consider all observations of the training series to build a model. It normally involves the transposition of the data sequence into an attribute-value table which is used as input to machine learning regression algorithms. Fig. 7 shows how to employ this approach.

In Fig. 7, the transposition procedure of the sequence $Z$ with size $m = 15$ into the attribute-value format is obtained sweeping a sliding window of length $l = 3$. This window is iteratively shifted on the time series in order to collect all the subsequences formed by $l$ consecutive observations. Each extracted subsequence refers to a pair $(X_i, y_i)$ where: $X_i = (x_{i1}, x_{i2}, x_{il})$ and corresponds to the temporal pattern of length $l$; and $y_i$ indicates the subsequent value to $X_i$, observed at

| $x_1$ | $x_2$ | $x_l$ | $y$ | | $\hat{y}$ |
|---|---|---|---|---|---|
| $z_{t-2}$ | $z_{t-1}$ | $z_t$ | $z_{t+1}$ | | $\hat{z}_{t+1}$ |
| $z_9$ | $z_{10}$ | $z_{11}$ | $z_{12}$ | → | $\hat{z}_{12}$ |
| $z_{10}$ | $z_{11}$ | $\hat{z}_{12}$ | $z_{13}$ | → | $\hat{z}_{13}$ |
| $z_{11}$ | $\hat{z}_{12}$ | $\hat{z}_{13}$ | $z_{14}$ | → | $\hat{z}_{14}$ |
| $\hat{z}_{12}$ | $\hat{z}_{13}$ | $\hat{z}_{14}$ | $z_{15}$ | → | $\hat{z}_{15}$ |

(a) Approximate iteration

| $x_1$ | $x_2$ | $x_l$ | $y$ | | $\hat{y}$ |
|---|---|---|---|---|---|
| $z_{t-2}$ | $z_{t-1}$ | $z_t$ | $z_{t+1}$ | | $\hat{z}_{t+1}$ |
| $z_9$ | $z_{10}$ | $z_{11}$ | $z_{12}$ | → | $\hat{z}_{12}$ |
| $z_{10}$ | $z_{11}$ | $z_{12}$ | $z_{13}$ | → | $\hat{z}_{13}$ |
| $z_{11}$ | $z_{12}$ | $z_{13}$ | $z_{14}$ | → | $\hat{z}_{14}$ |
| $z_{12}$ | $z_{13}$ | $z_{14}$ | $z_{15}$ | → | $\hat{z}_{15}$ |

(b) Updated iteration

**Fig. 8.** Representation of the multi-step-ahead prediction strategy for the global approach.

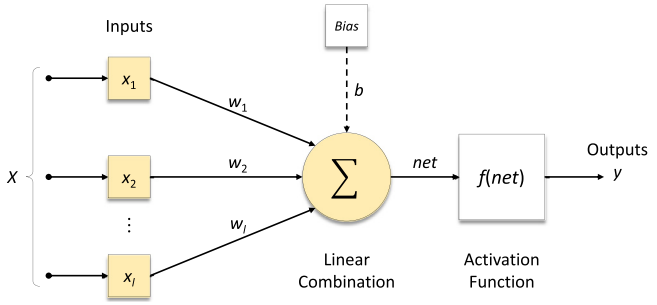



**Fig. 9.** Structure of perceptron.

the instant $l+1$. The set of pairs $(X_{ij}, y_{ij})$, where $j \in [1, m-l]$, constitutes the attribute-value table. The idea behind this conversion is to use observations from the past to predict an observation in the future.

Assuming a prediction horizon $h = 4$, the data of the table are partitioned into two sets: (i) training set, used to build the model; and (ii) test set, used for model performance evaluation. The prediction accuracy is estimated by comparing the predicted values $\hat{y}_k$ with their actual values $y_k$, where $k \in [1, h]$. Fig. 8 exemplifies an application of the multi-step-ahead projection strategy, with approximate and updated iterations, to calculate the estimates $\hat{y}_k$.

Despite its simplicity, the global approach is not exempt from limitations. The most obvious of them is the fact that the pairs $(X_{ij}, y_{ij})$ are considered independent and identically distributed by traditional machine learning algorithms. This assumption leads to a loss of temporal information, which implies in the performance degradation of the resulting regression model. Among the methods that apply this approach, we can cite those that consider polynomial and rational functions. Moreover, there are also those grounded in ANN [46] and SVM [37,40].

The local approach comprises machine learning algorithms that have been adapted to include temporal information in the prediction process. Such methods partition the time series into subsequences whose the closest or most important values related to the current value are combined to produce the future value. These combinations are undertaken by approximation functions, such as simple local average and weighted. Examples of methods that use the local approach are variations of the *k*NN algorithm [33,48].

In the following subsections, we present three state-of-the-art machine learning algorithms that we consider in our experimental evaluation.

#### *4.2.1. Artificial neural networks*

ANN are computational models inspired by the information processing performed by the human brain [19]. The Perceptron, exhibited in Fig. 9, is the simplest form of an ANN used for, besides other tasks, the classification of linearly separable classes [30,38].

In Fig. 9, the single neuron comprises $l$ data inputs $x_i \in X$. The $i^{\text{th}}$ element of $X$, eventually provided by adjacent neurons, is associated with a synaptic weight $w_i$. This weight can assume a negative or positive value that reflects the importance of the input. The linear combination of inputs with weights, added a threshold (bias) $b \in \Re$, results in the *net* value (Eq. 28). This value is sent to an activation function $f$ that sets the output $y$ of the neuron.

$$net = \sum_{i=1}^{l} w_i x_i + b \tag{28}$$

The bias aims to correct, increase or decrease the *net* value. This correction contributes to achieving a result of $f(net)$ closest to the expected. In the model presented in [30], $f$ corresponds to a staircase function. Regarding $y$, it can be binary with $y \in \{0, 1\}$ or $y \in \{-1, 1\}$, as well continuous where $y \in \Re$. Furthermore, we may apply other types of activation function.

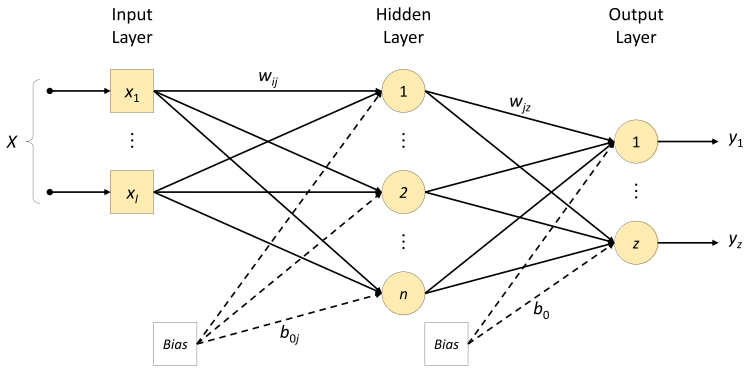



**Fig. 10.** Structure of MLP with a single hidden layer.

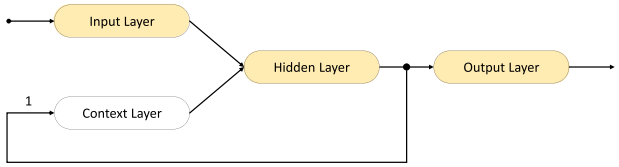



**Fig. 11.** Structure of SRN. Recurrent link between hidden and context layer.

A learning process with a finite number of iterations adjusts the synaptic weights of the Perceptron. The learning is conducted by the error correction rule known as the Perceptron convergence algorithm [27]. This algorithm searches for a weight vector $w$, such that the two equalities of the step function are satisfied.

The proposition of the Backpropagation learning algorithm [39] allowed ANN with more than two layers. One type of ANN with multiple layers, usually trained by the backpropagation algorithm, is the MLP. Fig. 10 shows the structure of a three-layered MLP.

The output of the model outlined in Fig. 10, but considering a single neuron in the output layer, can be represented as follows:

$$y = f\left(\sum_{j=1}^{n} w_j f\left(\sum_{i=1}^{l} w_{ij} x_i + b_{0j}\right) + b_0\right)$$

An MLP can have one or more layers of neurons between the input and output layers. These intermediate layers are units that do not interact directly with the environment and work as combiners of characteristics. If there are appropriate connections between the input units and a considerable set of intermediate units, one can always find the representation that will produce the correct mapping between the data input and the results output. Although MLP models are more difficult to interpret, their main advantages are the capacity to deal with large volumes of data and proper generalization.

The MLP accuracy is associated with three topological aspects: (i) determining the number of hidden layers; (ii) definition of the number of neurons in each layer; (iii) specification of synaptic weights that interconnect the neurons in different layers of the network. As reported in [10], to produce any mapping, at most two intermediate layers are required with a sufficient number of units per layer [11,35]. On another hand, with only a single intermediate layer is possible to approximate any continuous function.

We can categorize the Perceptron and MLP models as feed-forward neural networks because in them the neuron-to-neuron signals flow only in one direction: from input to output. Differently, in Recurrent Neural Networks (RNN), connections between neurons form a cycle, and the signals are able to move in different directions. For example, in the Simple Recurrent Network (SRN) [13], the state of the hidden layer at a given time is conditioned on its previous state by a context layer, as illustrated in Fig. 11. This recursion implies a short-term memory, which allows the network to store complex signals for arbitrarily long time periods.

The ability to model temporal dependencies makes RNN architectures especially suitable for tasks as speech classification and prediction, where input and/or output covers data sequences that are dependent. Among the several improved RNN, we can cite Bidirectional RNN (BRNN) [42], and Nonlinear Autoregressive RNN with Exogenous Inputs (NARX) [43].

In recent years, the concept of fuzzy logic and wavelets has become very attractive in the field of RNN [28]. Fuzzy logic enables us to reduce the complexity of the data and to deal with uncertainty [7]; wavelet transform allows us to analyze non-stationary signals to discover their local details [36]; RNN has self-learning characteristic that increases the accuracy of the model. Their combination contributes to the development of models with fast learning capability that can describe nonlinear systems characterized by uncertainty.

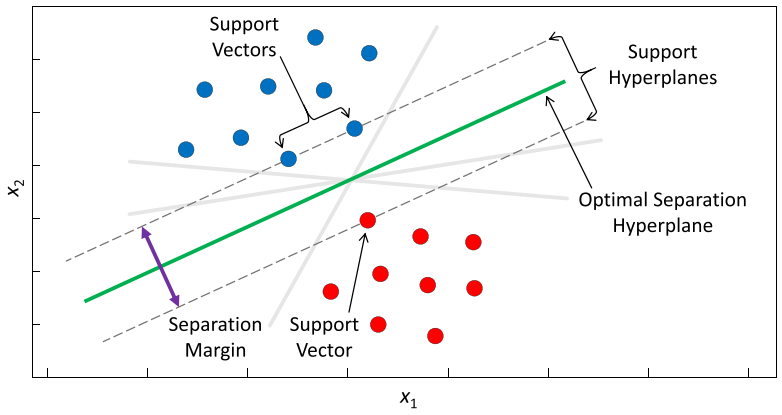



**Fig. 12.** Optimal separation hyperplane and its supporting hyperplanes. The ordered axes $x_1$ and $x_2$ represent the dimensions of the samples in a 2D space.

While theoretically powerful, the recurrent models aforementioned were widely considered to be hard to train due to the so-called vanishing and exploding gradient problems [4,20]. A popular solution to deal with this issue is adopting gated architectures, like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which were specifically designed to allow the network to learn much longer-range dependencies [21].

Although LSTM is responsible for several models considered state-of-the-art in the literature, its performance depends heavily on the amount of data available. Besides, the parameterization of LSTM-based networks is still very complex and expensive.

*4.2.2. Support vector machines*

SVM constitutes a machine learning technique based on the statistical learning theory [45] with the ability to solve distinct problems as classification and regression [17].

Although SVM models have a similar structure to ANN, they differ in how the learning is conducted. While ANN work by minimizing the empirical risk, *i.e.*, the error minimization of the induced model on the training data, SVM are grounded on the principle of structural risk minimization, which seeks the lowest training error while minimizing an upper bound on the generalization error of the model (model error when applied to test data) [45].

The generalization concept is best understood in the case of binary classification. Thus, given a set of points belonging two classes, an SVM determines the hyperplane that separates them placing the largest possible number of points of a class on the same side, while the distance from each class to the decision surface is maximized. Fig. 12 shows a set of straight lines that discriminate the data into two classes. Between these straight lines, only one maximizes the separation margin (distance between the hyperplane and the nearest sample of each class). The straight line with maximum margin, called optimal separation hyperplane, is the object to be searched during the model training.

The technique indicated in Fig. 12 is restricted to linearly separable problems. However, in situations where the samples are not linearly separable, the solution focuses on mapping the input data to a higher-dimensional space (feature space). We can achieve this mapping using a kernel function.

Linear, polynomial, and Radial Basis Function (RBF) kernels are the most applied in practice. Each one of them covers parameters that need to be set. For example, the SVM model with RBF kernel requires two parameters [17]: (i) $\mathbb{C}$, which is a regularization term that imposes a weight on the training set errors minimization regarding the model complexity minimization; and (ii) $\sigma$, that reflects the Gaussian's width of the kernel function. The number of radial functions and their respective centers is determined by the support vectors found.

The construction of an SVM implies in solving a quadratic problem with linear constraints which depends on the set of input data, parameters, and of the separation margin. During the training phase, the Lagrange multipliers that characterize the support vectors are obtained. These support vectors define the edges of the optimal separation hyperplane.

*4.2.3.* Nearest neighbors

Similarity-based methods, like the *k*NN classifier, are characterized by not constructing a model that explicitly describes the training dataset behavior. The model is built by simply storing the data sample. The generalization on the training set is performed every time we asked the algorithm for a new classification.

The general idea behind the adaptation of *k*NN for time series prediction is very intuitive. Given a series $Z = (z_1, z_2, \ldots, z_m)$ in which $z_t \in \Re$, the problem is to predict the value $z_{m+h}$, where $h$ is the prediction horizon. For simplicity, but without loss of generality, the idea will be discussed in a unitary horizon ($h = 1$), *i.e.*, considering only the prediction of the next value ($z_{m+1}$).

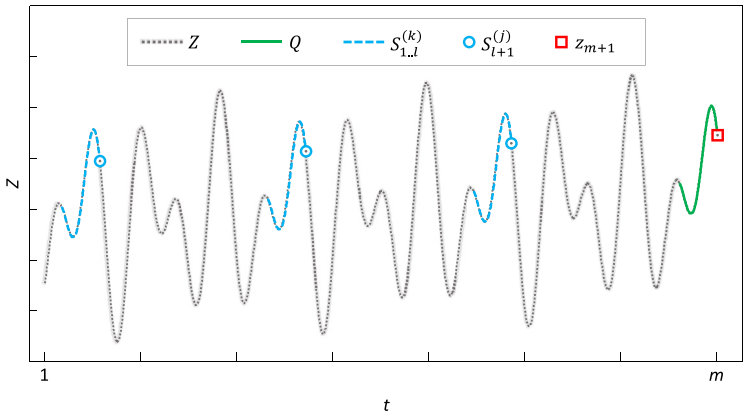



**Fig. 13.** An example that illustrates the *k*NN algorithm for time series prediction with $l = 25$ and $k = 3$.

The modified method uses the last $l$ observations as query $Q$, and searches for the $k$ most similar subsequences to $Q$, using a sliding window of size $l$. Given $S^{(1)}_{1..l}, \ldots, S^{(k)}_{1..l}$ as the $k$ most similar subsequences of $Q$, the algorithm uses the next observations of each subsequence $S^{(j)}_{l+1}$ with $1 \leq j \leq k$ to predict $z_{m+1}$. Thereby, the values of $S^{(j)}_{l+1}$ are provided as input to a prediction function $f$ to approximate the value of $z_{m+1}$. Eq. (29) is an example of ensemble function.

$$f(S) = \frac{1}{k} \sum_{j=1}^{k} S^{(j)}_{l+1} \tag{29}$$

In Eq. (29), the prediction function $f$ has the argument $S$ that denotes the set of $k$ most similar subsequences, and $S^{(j)}$ refers to $j$th nearest neighbor. This is the simplest way of combining the projections since the predictions average considers that all projected values are equally probable to occur in the future.

Fig. 13 displays an application of the described method with $l = 25$ and $k = 3$. The dotted line in gray represents the observations that belong to the time series; the green line indicates the subsequence of length 25 taken as reference query; the blue dotted lines express the most similar subsequences found using some similarity measure, in this case, the Euclidean distance; the blue circles correspond to the observations used for making the prediction; and the red square reflects the value to be predicted.

The prediction made by similarity-based methods considers only the previous $l$ observations. Thus, the temporal dependence is restricted to a limited number of previous observations, since usually a certain value is not influenced by observations that happened a long time ago.

Several surveys were conducted to analyze the performance of *k*NN with different ensemble functions and distance measures. Moreover, a few papers showed that these methods are useful to predict highly nonlinear and complex time series patterns [33,48]. A recent study proposed a novel and promising modification of the *k*NN algorithm for time series prediction, namely *k*NN - Time Series Prediction with Invariances (*k*NN-TSPI) [33]. This proposal differs from the literature by incorporating techniques for amplitude and offset invariance, complexity invariance, and treatment of trivial matches. According to the authors, these three invariances when combined allow more meaningful matching between the reference queries and temporal data subsequences.

### *4.3. Techniques for parameter estimation*

One of the main difficulties faced by researchers in the time series prediction is the search for the best parameter setting to fit a model according to a dataset.

In theoretical terms, the establishment of all parameters of a model could need the full exploitation of the state space. As this procedure is impractical for most real-world datasets, search algorithms are used to find a suboptimal solution with satisfactory performance and acceptable computational cost. Next, we present the main parameter estimation methods for time series prediction.

#### *4.3.1. Holdout validation*

The holdout validation technique performs the search for parameters values that minimize the predictive model error over a different number of intervals in the training data. Algorithm 1 describes the logic of this sampling technique to find adequate values for the parameters $l$ and $k$ of the *k*NN-TSPI method.

Algorithm 1 iteratively evaluates a set of previously defined parameters. At each iteration and according to the current parameters combination, a new model is built and adjusted on the $m - h$ observations of the subsequence $S$, *i.e.*, $z_1, \ldots, z_{m-h}$

**Algorithm 1:** Holdout validation.

```
/* S represents a training subsequence of length n extracted from a time series Z of size m          */
/* max_p is an upper bound for number of observations constituting a seasonal station in the historical
   series                                                                                             */
/* h indicates the amount of values to be predicted by the best model identified                      */
/* ℙ comprises the parameters list which resulted in the least prediction error                       */
Input: S, max_p, h
Output: ℙ
1  begin
2      min_error = ∞;
3      h' = (max_p + h) ÷ 2;
4      for l ← 3 : 2 : max_p do
5          for k ← 1 : 2 : 9 do
6              error = knn_tspi(S, l, k, h');
7              if error < min_error then
8                  min_error = error;
9                  l_best = l;
10                 k_best = k;
11             end
12         end
13     end
14     ℙ ← {l_best, k_best};
15     return ℙ;
16 end
```

($S \in Z$). Afterward, the model is extrapolated to a horizon prediction $h'$ whose length is equivalent to the validation subsequence $(s_{n-h'+1}, \ldots, s_n)$. At the end of this search, the most promising parameters ($\mathbb{P}$) are those which minimize the error between the predicted and validation subsequences. Many measures can measure this error, but the most common is the Mean Square Error (MSE).

There are different ways to choose the number of observations covered by $h'$. We have adopted $h' = (max\_p + h) \div 2$, where $h$ indicates the number of values to be projected in the test step by the prediction method using the best set of parameters found.

*4.3.2. Cross-validation*

Cross-validation is a sampling technique broadly used for evaluating models in machine learning. For time series prediction, the technique searches for the most adequate parameter values for global approach methods. Algorithm 2 details this sampling technique to search values for the parameters $l$, $\mathbb{C}$, and $\sigma$ of the SVM regression algorithm.

In Algorithm 2, the idea behind the search is similar to that observed in Algorithm 1, except by the content of the 4th and 7th lines. In the 4th line occurs the transposition of the training subsequence $S$ for the attribute-value format using a sliding window of length $l$, such as illustrated by the training set in Fig. 7. In the 7th line, the attribute-value table $T$ is randomly divided into $k$ samples (*kFolds*) mutually exclusive, all of approximately the same size. The $k$th sample is used as a validation set and the $k-1$ remaining samples are the training set. For each combination of $k-1$ samples, a model is constructed and adjusted according to the current parameters combination. The prediction error is computed on the validation set $k$ by a loss function as MSE. Evidently, when we use the MSE, the parameterization error is given by the average of the MSE of the $k$ generated models and is considered an estimative of the true error expected on independent data.

*4.3.3. Box-Jenkins method*

We can determine the parameters of an ARIMA model via a search mechanism guided by an information criterion that penalizes the models' adjustment with many parameters. This method is called Box-Jenkins. Algorithm 3 exemplifies the use of this method to identify all SARIMA parameters of order $(p, d, q) \times (P, D, Q)_s$.

In Algorithm 3, $p$, $d$, $q$, $P$, $D$, and $Q$ are relevant time delays of the time series within a search space pre-determined by the user. The value of *max_p* corresponds, in number of observations, a seasonal period in the series. In situations where this information is not clearly visible, we can apply the technique of scatter plot to obtain *max_p*. Given that the constant $\delta$ represents the initial level of the model, a condition for inclusion or omission of $\delta$ was inserted respecting the differentiation rules for SARIMA in the 10th line. In the 16th line, the most promising parameters are chosen so to minimize the AIC (Eq. (27)).

The values of $d$ and $D$ can be established in a preprocessing step integrating the time series until its variance becomes smaller than its original version (undifferentiated). The prior identification of these values is important to reduce the processing time of Algorithm 3.

**Algorithm 2:** Cross-validation.

```
// S represents a training subsequence
/* max_p specifies an upper bound for the number of observations constituting a seasonal station in the
   historical series                                                                                    */
/* kFolds is the number of partitions on which the training data sample will be split                  */
/* ℙ comprises the parameters list which resulted in the least prediction error                        */
Input: S, max_p, h
Output: ℙ
begin
    min_error = ∞;
    for l ← 3 : 2 : max_p do
        T ← generate_data_table(S, l);
        for ℂ ← 0 : 0.25 : 1 do
            for σ ← 0.005 : 0.05 : 0.25 do
                error = cross_validation(T, kFolds, ℂ, σ, model = "SVM");
                if error < min_error then
                    min_error = error;
                    l_best = l;
                    ℂ_best = ℂ;
                    σ_best = σ;
                end
            end
        end
    end
    ℙ ← {l_best, ℂ_best, σ_best};
    return ℙ;
end
```

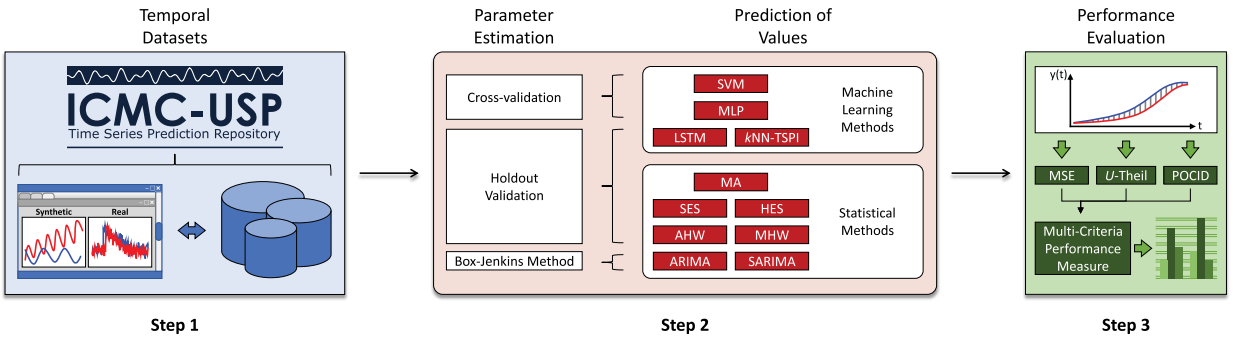



**Fig. 14.** Experimental setup.

## 5. Empirical evaluation

The protocol of our performance evaluation was organized in three steps, as outlined in Fig. 14.

In Step 1, we have selected 95 datasets, which 40 of them are synthetic and 55 are from real domains. These data pose some problems that one usually encounters in a typical one or multi-step-ahead prediction tasks such as the growing trend, non-stationarity, outliers and multiple overlying seasonalities. The datasets and their full descriptions are available online at the ICMC-USP Time Series Prediction Repository [32]. Such archive is an original contribution of this article and most of the datasets maintained by it are often reported in the literature, as demonstrated by our systematic review. However, before we created the repository, these datasets were scattered among multiple portfolios (online archives, web pages, supplementary materials, and books). Our idea was to centralize this data in a single repository to facilitate its recovery and usage. In what follows, we present a summary of their properties.

We designed the 40 synthetic datasets to analyze and understand the performance of algorithms over peculiar characteristics of data. We grouped these data into three categories according to its originating process: (i) deterministic; (ii) stochastic; and (iii) chaotic.

**Algorithm 3:** Box-Jenkins method.

```
// S represents a training subsequence
/* max_ord1 e max_ord2 are three-position vectors whose values indicate the maximum lag lengths of the
   non-seasonal and seasonal parts, respectively                                                    */
/* max_p is an upper bound for the number of observations constituting a seasonal station in the
   historical series                                                                                */
/* ℙ comprises the parameters list which resulted in the least prediction error                     */
Input: S, max_ord1, max_ord2, max_p
Output: ℙ
1  begin
2      best_aic = ∞;
3      n = length(S);
4      for p ← 0 : max_ord1[1] do
5          for d ← 0 : max_ord1[2] do
6              for q ← 0 : max_ord1[3] do
7                  for P ← 0 : max_ord2[1] do
8                      for D ← 0 : max_ord2[2] do
9                          for Q ← 0 : max_ord2[3] do
10                             if d + D ≤ 1 then
11                                 fit = sarima(S, [p, d, q],[P, D, Q], max_p, δ = TRUE);
12                             else
13                                 fit = sarima(S, [p, d, q],[P, D, Q], max_p, δ = FALSE);
14                             end
15                             fit_aic = −2 ∗ fit.loglik + (log(n) + 1) ∗ length(fit.coef);
16                             if fit_aic < best_aic then
17                                 best_aic ← fit_aic;
18                                 best_fit ← fit;
19                                 best_model ← [p, d, q, P, D, Q, max_p];
20                             end
21                         end
22                     end
23                 end
24             end
25         end
26     end
27     ℙ ← {best_aic, best_fit, best_model};
28     return ℙ;
29 end
```

Table 2 shows the main characteristics of each synthetic dataset such as size ($m$), the maximum number of observations in a seasonal variation ($max\_p$), and the prediction horizon ($h$). In our experimental evaluation, we set the value $h$ as 5% of the series size.

The 55 real datasets are from distinct domains, including agriculture (ID = 27.M, 35.M, and 47.M–52.M in Table 2(b)), climatology (ID = 01.A, 03.D, 04.D, 10.D–14.D, 29.M, 32.M, 36.M, and 43.M–45.M in Table 2(b)), engineering (ID = 42.M in Table 2(b)), finance (ID = 05.D–07.D, 09.D, 16.D–22.D, 30.M, and 38.M in Table 2(b)), medicine (ID = 08.D, 53.I, and 54.I in Table 2(b)), physics (ID = 02.A, 15.D, 40.M, and 55.I in Table 2(b)), and tourism (ID = 39.M in Table 2(b)). Table 2(b) describes these time series. Besides the previously presented characteristics for the synthetic datasets, Table 2(b) also exhibits the type of data acquisition and the time interval of acquisition.

In Step 2, we estimate the parameters according to the particularities of each method. In this context, the non-parametric algorithms SVM and MLP, which are applied according to the global approach, had their parameters determined through cross-validation in ten partitions and the minimization of the MSE (Algorithm 2). On the other hand, the parameters of the parametric models ARIMA and SARIMA were defined using the Box-Jenkins method and the minimization of the AIC (Eq. (27)) and maximum likelihood (Algorithm 3). The remaining methods had their parameters estimated by holdout validation with MSE minimization (Algorithm 1).

Table 3 shows the eleven predictors evaluated, as well as a description of their parameters and the range of values considered in the estimation step.

We previously evaluated, in addition to the single-layer MLP, two other candidate algorithms: SRN and RNN with two layers. RNN obtained a lower predictive performance than the SRN and MLP models. In contrast, SRN and MLP achieved

**Table 2**
Summary of characteristics and settings of the benchmark datasets.

(a) Synthetic data

| ID | Dataset | *m* | *max_p* | *h* (*m* × 5%) |
|---|---|---|---|---|
| | Fourier A: | 790 | | |
| 01.D | • Constant level | | 25 | 40 |
| 02.D | • Increasing trend | | 25 | 40 |
| 03.D | • Decreasing trend | | 25 | 40 |
| | Fourier B: | 790 | | |
| 04.D | • Constant level | | 38 | 40 |
| 05.D | • Increasing trend | | 38 | 40 |
| 06.D | • Decreasing trend | | 38 | 40 |
| | Fourier C: | 790 | | |
| 07.D | • Constant level | | 34 | 40 |
| 08.D | • Increasing trend | | 34 | 40 |
| 09.D | • Decreasing trend | | 34 | 40 |
| | Fourier D: | 790 | | |
| 10.D | • Constant level | | 38 | 40 |
| 11.D | • Increasing trend | | 38 | 40 |
| 12.D | • Decreasing trend | | 38 | 40 |
| | Seasonal dependence: | 2200 | | |
| 13.D | • Constant level | | 25 | 110 |
| 14.D | • Increasing trend | | 25 | 110 |
| 15.D | • Decreasing trend | | 25 | 110 |
| 16.D | Multiplicative seasonality | 590 | 14 | 30 |
| 17.D | High frequency | 550 | 63 | 28 |
| | CCA: | 1000 | | |

(b) Real data

| ID | Dataset | Acquisition | Start | Finish | *m* | *max_p* | *h* |
|---|---|---|---|---|---|---|---|
| 01.A | Fortaleza | Annual | 1849 | 1997 | 149 | 6 | 7 |
| 02.A | *Manchas* | Annual | 1749 | 1942 | 176 | 11 | 12 |
| | *Atmosfera*: | Daily | Jan-01-1997 | Dec-31-1997 | 365 | | |
| 03.D | • *Temperatura* | | | | | 7 | 31 |
| 04.D | • *Umidade relativa do Ar* | | | | | 7 | 31 |
| 05.D | Banespa | Daily | Jan-03-1995 | Dec-27-2000 | 1499 | 7 | 88 |
| 06.D | CEMIG | Daily | Jan-03-1995 | Dec-27-2000 | 1499 | 7 | 88 |
| 07.D | *IBV* | Daily | Jan-03-1995 | Dec-27-2000 | 1499 | 7 | 88 |
| 08.D | Patient demand | Daily | Jan-01-2007 | Mar-31-2009 | 821 | 7 | 90 |
| 09.D | Petrobras | Daily | Jan-03-1995 | Dec-27-2000 | 1499 | 7 | 88 |
| | *Poluição*: | Daily | Jan-01-1997 | Dec-31-1997 | 365 | | |
| 10.D | • PM10 | | | | | 7 | 31 |
| 11.D | • SO2 | | | | | 7 | 31 |
| 12.D | • CO | | | | | 7 | 31 |
| 13.D | • O3 | | | | | 7 | 31 |
| 14.D | • NO2 | | | | | 7 | 31 |
| 15.D | Star | Daily | 1922 | 1924 | 600 | 7 | 25 |
| | Stock market: | Daily | Jan-06-1986 | Dec-31-1997 | 3128 | | |
| 16.D | • Amsterdam | | | | | 7 | 92 |
| 17.D | • Frankfurt | | | | | 7 | 92 |
| 18.D | • London | | | | | 7 | 92 |
| 19.D | • Hong Kong | | | | | 7 | 92 |
| 20.D | • Japan | | | | | 7 | 92 |
| 21.D | • Singapore | | | | | 7 | 92 |
| 22.D | • New York | | | | | 7 | 92 |
| 23.M | *Bebida* | Monthly | Jan-1985 | July-2000 | 187 | 12 | 7 |
| | CBE: | Monthly | Jan-1958 | Dec-1990 | 396 | | |
| 24.M | • Chocolate | | | | | 12 | 24 |
| 25.M | • Beer | | | | | 12 | 24 |

(*continued on next page*)

**Table 2** (*continued*)

(a) Synthetic data

| ID | Dataset | *m* | *max_p* | *h* (*m* × 5%) |
|---|---|---|---|---|
| 18.S | • Constant level | | 12 | 50 |
| 19.S | • Seasonal patterns | | 30 | 50 |
| 20.S | • Increasing trend | | 12 | 50 |
| 21.S | • Decreasing trend | | 12 | 50 |
| 22.S | • Upward shift | | 12 | 50 |
| 23.S | • Downward shift | | 12 | 50 |
| | CCB: | 1000 | | |
| 24.S | • Constant level | | 30 | 50 |
| 25.S | • Double seasonality | | 30 | 50 |
| 26.S | • Increasing trend | | 30 | 50 |
| 27.S | • Decreasing trend | | 30 | 50 |
| 28.S | • Upward shift | | 30 | 50 |
| 29.S | • Downward shift | | 30 | 50 |
| | SDN: | 2200 | | |
| 30.S | • Constant level | | 25 | 110 |
| 31.S | • Increasing trend | | 25 | 110 |
| 32.S | • Decreasing trend | | 25 | 110 |
| 33.C | Logistic map | 550 | 4 | 28 |
| 34.C | Hénon Map | 3000 | 3 | 150 |
| 35.C | Mackey-Glass system | 3000 | 7 | 150 |
| 36.C | Lorenz system | 3000 | 25 | 150 |
| 37.C | Rössler system | 3000 | 14 | 150 |
| | Chaotic signals: | 550 | | |
| 38.C | • A | | 22 | 28 |
| 39.C | • B | | 7 | 28 |
| 40.C | ECGSYN | 3000 | 60 | 150 |

(b) Real data

| ID | Dataset | Acquisition | Start | Finish | *m* | *max_p* | *h* |
|---|---|---|---|---|---|---|---|
| 26.M | • Electricity production | | | | | 12 | 24 |
| 27.M | Chicken | Monthly | Jan-1999 | July-2014 | 187 | 12 | 7 |
| 28.M | *Consumo* | Monthly | Jan-1984 | Oct-1996 | 154 | 12 | 10 |
| 29.M | Darwin | Monthly | 1882 | 1998 | 1400 | 12 | 36 |
| 30.M | Dow Jones | Monthly | Jan-1950 | May-2003 | 641 | 12 | 29 |
| 31.M | *Energia* | Monthly | Jan-1968 | Sept-1979 | 141 | 12 | 9 |
| 32.M | Global | Monthly | Jan-1856 | Dec-2005 | 1800 | 12 | 36 |
| 33.M | *ICV* | Monthly | Jan-1970 | June-1980 | 126 | 12 | 6 |
| 34.M | *IPI* | Monthly | Jan-1985 | July-2000 | 187 | 12 | 7 |
| 35.M | Latex | Monthly | Jan-1998 | July-2014 | 199 | 12 | 7 |
| 36.M | Lavras | Monthly | Jan-1966 | Dec-1997 | 384 | 12 | 12 |
| 37.M | Maine | Monthly | Jan-1996 | Aug-2006 | 128 | 12 | 8 |
| 38.M | MPrime | Monthly | Jan-1949 | Nov-2007 | 707 | 12 | 23 |
| 39.M | OSVisit | Monthly | 1977 | 1995 | 228 | 12 | 12 |
| 40.M | *Ozônio* | Monthly | Jan-1956 | Dec-1970 | 180 | 12 | 12 |
| 41.M | *PFI* | Monthly | Jan-1991 | July-2000 | 115 | 12 | 7 |
| 42.M | Reservoir | Monthly | Jan-1909 | Dec-1980 | 864 | 12 | 24 |
| 43.M | STemp | Monthly | Jan-1850 | Dec-2007 | 1896 | 12 | 36 |
| | *Temperatura:* | Monthly | Jan-1976 | Dec-1985 | 120 | | |
| 44.M | • Cananéia | | | | | 12 | 12 |
| 45.M | • Ubatuba | | | | | 12 | 12 |
| 46.M | USA | Monthly | Jan-1996 | Oct-2006 | 130 | 12 | 6 |
| | Wine: | Monthly | Jan-1980 | July-1995 | 187 | | |
| 47.M | • Fortified white | | | | | 12 | 19 |
| 48.M | • Dry white | | | | | 12 | 19 |
| 49.M | • Sweet white | | | | | 12 | 19 |
| 50.M | • Red | | | | | 12 | 19 |
| 51.M | • Rose | | | | | 12 | 19 |
| 52.M | • Sparkling | | | | | 12 | 19 |
| | ECG: | 0.5s Intervals | Oct-1996 | Oct-1996 | 1800 | | |
| 53.I | • A | | | | | 60 | 120 |
| 54.I | • B | | | | | 60 | 120 |
| 55.I | Laser | 1.0s Intervals | 1991 | 1991 | 1000 | 8 | 100 |

**Table 3**
Algorithms and ranges of numeric variation defined for their parameters.

| Algorithm | Parameters | Variation range (initial : step : final) |
|---|---|---|
| SVM | Size of the search window ($l$) | $l =$ 3 : 2 : max_$p$ |
| | Regularization parameter ($\mathbb{C}$) | $\mathbb{C} =$ 0: 0.25: 1 |
| | Gaussian's width of the radial basis kernel function ($\sigma$) | $\sigma =$ 0.005: 0.05: 0.25 |
| MLP | Size of the search window ($l$) | $l =$ 3 : 2 : max_$p$ |
| | Number of units in the hidden layer ($n$) | $n =$ 3 : 2 : max_$p$ |
| | Learning rate ($\Upsilon$) | $\Upsilon =$ 0.03 |
| | Momentum ($\mathbb{M}$) | $\mathbb{M} =$ 0.2 |
| | Number of epochs ($\mathbb{E}$) | $\mathbb{E} =$ 500 |
| LSTM | Size of the search window ($l$) | $l =$ 3 : 2 : max_$p$ |
| | Number of units in the first hidden layer ($n_1$) | $n_1 =$ 3 : 2 : max_$p$ |
| | Number of units in the second hidden layer ($n_2$) | $n_2 =$ 3 : 2 : max_$p$ |
| | Learning rate ($\Upsilon$) | $\Upsilon =$ 0.03 |
| | Momentum ($\mathbb{M}$) | $\mathbb{M} =$ 0.2 |
| | Number of epochs ($\mathbb{E}$) | $\mathbb{E} =$ 500 |
| *k*NN-TSPI | Size of the search window ($l$) | $l =$ 3 : 2 : max_$p$ |
| | Number of nearest neighbors ($k$) | $k =$ 1: 2: 9 |
| MA | Number of observations used in the average ($r$) | $r =$ 3 : 2 : max_$p$ |
| SES | Smoothing constant associated with the level ($\alpha$) | $\alpha =$ 0: 0.25: 1 |
| HES | Smoothing constant associated with the level ($\alpha$) | $\alpha =$ 0: 0.25: 1 |
| | Smoothing constant associated with the trend ($\beta$) | $\beta =$ 0: 0.25: 1 |
| AHW | Smoothing constant associated with the level ($\alpha$) | $\alpha =$ 0: 0.25: 1 |
| MHW | Smoothing constant associated with the trend ($\beta$) | $\beta =$ 0: 0.25: 1 |
| | Smoothing constant associated with the seasonality ($\gamma$) | $\gamma =$ 0: 0.25: 1 |
| | Number of observations that make up a seasonal period ($s$) | $s =$ 3 : 2 : max_$p$ |
| ARIMA | Order of the autoregression procedure ($p$) | $p =$ $0 : 1 : \sqrt{\log(m-h)}$ |
| | Degree of the differentiation operator ($d$) | $d =$ 0: 1: 2 |
| | Order of the moving average procedure ($q$) | $q =$ $0 : 1 : \sqrt{\log(m-h)}$ |
| SARIMA | Order of the autoregression procedure ($p$) | $p =$ $0 : 1 : \sqrt{\log(m-h)}$ |
| | Degree of the differentiation operator ($d$) | $d =$ 0: 1: 2 |
| | Order of the moving average procedure ($q$) | $q =$ $0 : 1 : \sqrt{\log(m-h)}$ |
| | Order of the seasonal autoregression procedure ($P$) | $P =$ $0 : 1 : \sqrt{\log(m-h)}$ |
| | Degree of the seasonal differentiation operator ($D$) | $D =$ 0: 1: 2 |
| | Order of the seasonal moving average procedure ($Q$) | $Q =$ $0 : 1 : \sqrt{\log(m-h)}$ |
| | Number of observations that make up a seasonal period ($s$) | $s =$ *max_p* |

comparable results, since we did not find statistically significant differences between their performances. As MLP is a more straightforward approach and has faster execution time, we prefer to keep only the results of this neural network in this paper.

After estimating the best parameters, predictive models were constructed and adjusted to the training data. Each model was posteriorly extrapolated to predict $h$ periods ahead in agreement with the two projection strategies presented in Section 4 and designated as: (i) multi-step-ahead with approximate iteration; and (ii) multi-step-ahead with updated iteration.

In Step 3, the projected data were compared with the test data using MSE measure, Theil's $U$ (TU) coefficient, and hit rate Prediction Of Change In Direction (POCID).

The MSE measure, denoted by Eq. (30), works with the difference (error) between the actual value observed ($z_t$) and the predicted value ($\hat{z}_t$).

$$MSE = \frac{1}{h}\sum_{t=1}^{h}(z_t - \hat{z}_t)^2 \tag{30}$$

In Eq. (30), the quadratic sum of the prediction error is divided by the number of observations. Thus, an efficient predictor will have an MSE value close to zero. The TU coefficient, expressed by Eq. (31), is based on the MSE of the predictor, normalized by the prediction error of a trivial (or naïve) model. The naïve model assumes that the best value for time $t+1$ is the value obtained at time $t$.

$$TU = \frac{\sum_{t=1}^{h}(z_t - \hat{z}_t)^2}{\sum_{t=1}^{h}(z_t - z_{t-1})^2} \tag{31}$$

The values obtained by Eq. (31) can be interpreted in the following way: if $TU > 1$, the algorithm's performance is lower than the naïve model; if $TU = 1$, the algorithm's performance is the same as the naïve model; if $TU < 1$, the algorithm's performance is higher than the naïve model; and if $TU \leq 0.55$, the algorithm of interest is trusted to carry out

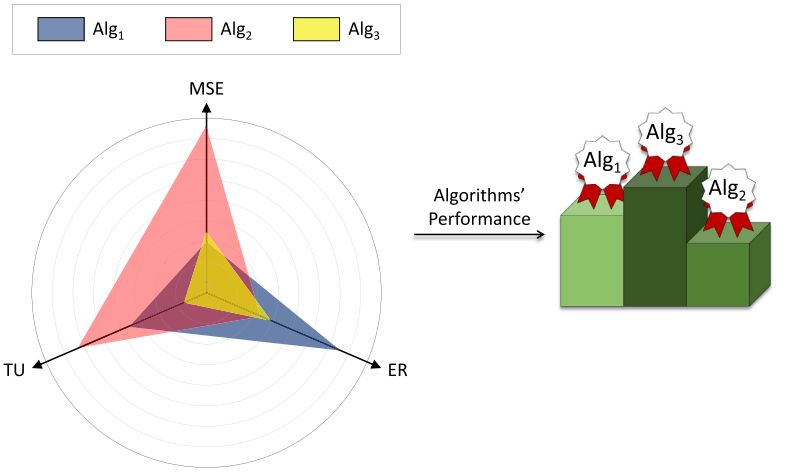



**Fig. 15.** Multi-criteria performance measure.

future predictions.

$$POCID = \frac{\sum_{t=1}^{h} D_t}{h} \times 100 \tag{32}$$

Another performance index considered was the POCID, which is formalized by the Eq. (32). In this equation, the term $D_t$ stores the value 1 if $(\hat{z}_t - \hat{z}_{t-1})(z_t - z_{t-1}) > 0$, and 0 otherwise. The idea of this index is to estimate the accuracy of direction's changes of the projected data, *i.e.*, if the future value will increase or decrease when compared to current value. We should use POCID in a complementary way to the analysis of the prediction errors. It is not advisable to make a decision based solely on POCID values.

Choosing the best model can be a difficult task when different performance measures are available. A method that comprises distinct relevant variables is a key element to workaround such problem. In this direction, we can use the Multi-Criteria Performance Measure (MCPM) developed in [34].

In this work, we employ the MCPM to combine the MSE, TU, and POCID indexes. Differently from the error measures, which yield values that must be minimized, POCID must be maximized. Therefore, we adopt the POCID complement, also called Error Rate (ER), which is determined by $ER = 100 - POCID$. Fig. 15 illustrates how to calculate the MCPM.

A radar chart consisting of three axes, each one representing an individual performance measure, is illustrated in Fig. 15. The final value of MCPM is achieved by the sum of the total area of each triangle. Lower values of MCPM correspond to a better predictive performance for an algorithm, as portrayed in the right side of Fig. 15.

From the values of the adopted evaluation measures, it was possible to compare the investigated algorithms objectively. Friedman's non-parametric statistical test for paired data and multiple comparisons, with a significance level of 5% ($p$-value $< 0.05$), followed by Nemenyi posthoc test[3], was employed to compare the results.

The experimental protocol execution contemplated the use of the following programming languages: MATLAB and GNU Octave, as well as their packages of functions for time series prediction; R with Forecast package; and Java with Weka library.

## 6. Results and discussion

We present and discuss the empirical results arranged into three large comparative studies: (i) predictive models applied to synthetic datasets – including an isolated assessment of deterministic, stochastic, and chaotic data; (ii) predictive models applied to real datasets; and (iii) predictive models applied to both synthetic and real datasets. Note that the sequence of our discussion accompanies the complexity increment to model the data and the difficulty of the prediction task.

We used Critical Difference (CD) diagrams to show the statistical validation results. In these diagrams, the scale indicates the rank position of each predictor according to their average performance [12]. Algorithms connected by a thick line have not presented Statistically Significant Differences (SSD) in quality. In the supplementary material or at the ICMC-USP Time Series Prediction Repository [32], we can see detailed results of MSE, TU and POCID, as well the values found in the step of parameters estimation.

In a complementary way and due to the particularities of TU and POCID indexes, they had their values summarized in full stacked area charts and bar charts with Standard Deviations (SD), respectively. We also employed a MCPM to provide an overview of the results.

[3] All statistical tests presented in this paper were performed using KEEL Software Tool for Windows, http://www.keel.es.

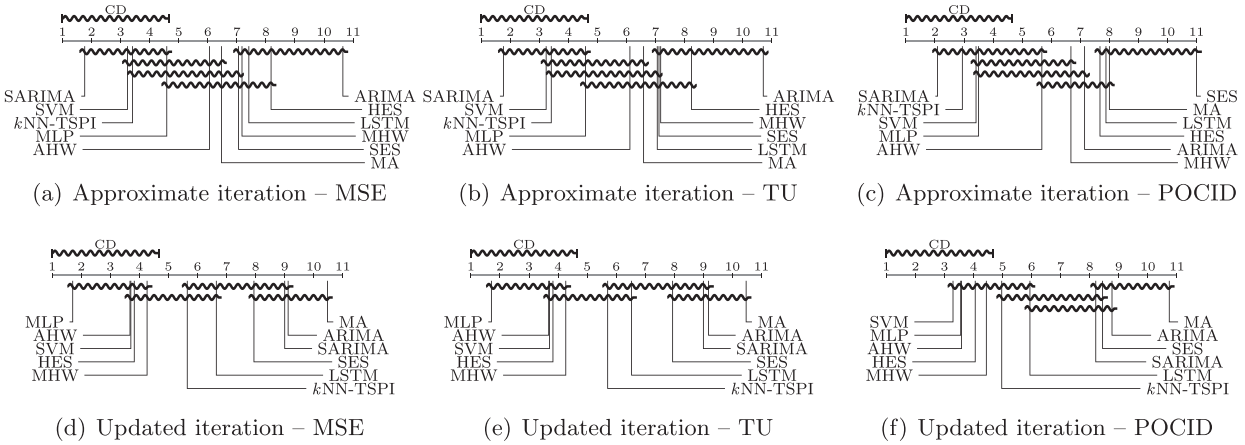



**Fig. 16.** CD diagrams for the MSE, TU, and POCID values coming from the predictors on deterministic time series.

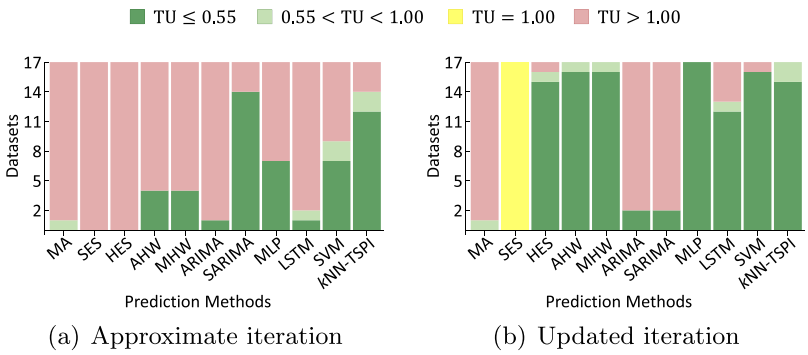



**Fig. 17.** Performance of the prediction methods for the four ranges of TU values in deterministic time series.

### 6.1. Synthetic data

This collection of experiments covers 40 synthetic time series, of which 17 are deterministic, 15 stochastic, and eight chaotic. In the next subsections we detail our analysis according to these categories.

#### 6.1.1. Deterministic time series

The computational tests conducted using deterministic data involved 374 configurations (11 predictors × 2 projection strategies × 17 datasets). Fig. 16 presents the CD diagrams concerning the MSE, TU, and POCID indexes.

According to the ranking shown in Fig. 16, *k*NN-TSPI with approximate iteration was in third place considering the MSE (Fig. 16(a)) and TU (Fig. 16(b)) measures, losing without SSD for SARIMA and SVM. For the POCID index (Fig. 16(c)), the SARIMA, *k*NN-TSPI and SVM methods, achieved the best results but without SSD among them.

For updated iteration, MLP presented the best MSE (Fig. 16(d)) and TU (Fig. 16(e)) averages. However, in terms of POCID (Fig. 16(f)), such configuration was in the second position without SSD compared to SVM.

In Fig. 17, we show for each method, how many datasets presented TU values in one of their four possible ranges. We can note that when using approximate iteration (Fig. 17(a)), SARIMA was reliable in 14 of 17 datasets ($TU \leq 0.55$). *k*NN-TSPI reached the second best performance, so that for 14 datasets (12 + 2) its use was preferable to the trivial model ($TU < 1$). As expected, SES failed to beat the naïve model for no one dataset. This fact reinforces that it does not support the prediction via approximate iteration.

As for updated iteration (Fig. 17(b)), MLP achieved reliable results for 17 datasets ($TU \leq 0.55$). From the same point of view, Holt-Winters models (AHW and MHW) presented the second best performance among all algorithms. In other words, their use was preferable ($TU < 1$) to the trivial model in 17 datasets (16 + 1), of which 16 provided reliable modelings ($TU \leq 0.55$).

Analyzing both projection strategies in Fig. 17, we can see that, in general, the *k*NN-TSPI performance was more stable than the other methods to predict deterministic time series.

In Fig. 18, we display the average results of POCID and their respective SD for each algorithm and projection strategy. SARIMA with approximate iteration achieved an average hit rate of 98.29% (SD = 3.83%) on the projection horizons trends.

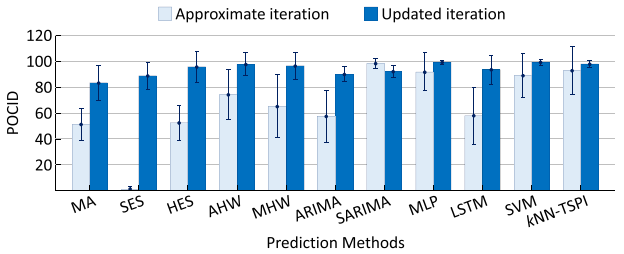



**Fig. 18.** Averages and SD of POCID obtained by the prediction methods on deterministic time series.

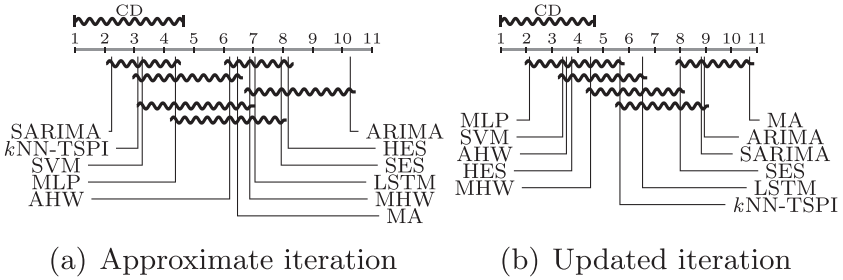



**Fig. 19.** MCPM over deterministic time series.

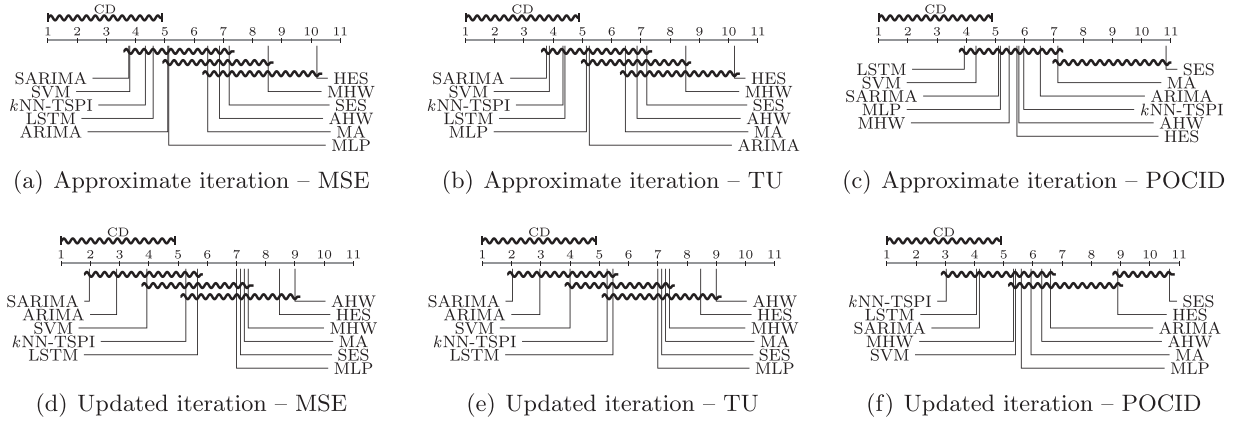



**Fig. 20.** CD diagrams for the MSE, TU, and POCID values coming from the predictors on stochastic time series.

In contrast, *k*NN-TSPI with approximate iteration reached an average hit rate of 92.99% (SD = 18.44%). Considering the projection strategy with updated iteration, the highest hit rates were obtained by MLP – 99.39% (SD = 1.24%) – and SVM – 99.36% (SD = 2.42%).

Fig. 19 shows the CD diagrams with respect to the MCPM total area values. SARIMA provided the best results when configured with approximate iteration (Fig. 19(a)). On the other hand, employing updated iteration (Fig. 19(b)), MLP recorded the smallest prediction errors. Nevertheless, SVM was the algorithm that obtained, for both projection strategies, the most stable performance in terms of prediction error and hit rate on the projection horizons trends.

*6.1.2. Stochastic time series*

The experiments performed from stochastic data encompassed 330 configurations (11 predictors × 2 projection strategies × 15 datasets). The CD diagrams portrayed in Fig 20 summarizes these computational tests according to MSE, TU, and POCID.

In Fig 20, *k*NN-TSPI with approximate iteration occupied the third position in the rankings of MSE (Fig. 20(a)) and TU (Fig. 20(b)). Such configuration has lost to SARIMA and SVM by a small difference margin. On average, LSTM with approximate iteration presented the best POCID rates (Fig. 20(c)), followed by SVM.

Considering the updated iteration, SVM achieved the third best result of MSE (Fig. 20(d)) and TU (Fig. 20(e)). The first two ranking positions were occupied by SARIMA and ARIMA, respectively. As for the POCID values (Fig. 20(f)), *k*NN-TSPI outperformed LSTM and SARIMA without SSD.

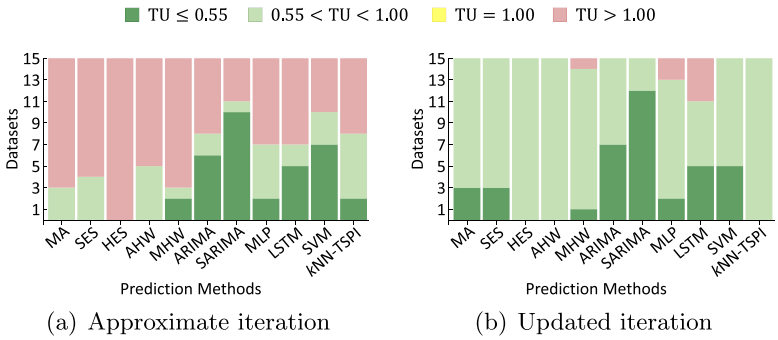



**Fig. 21.** Performance of the prediction methods for the four ranges of TU values in stochastic time series.

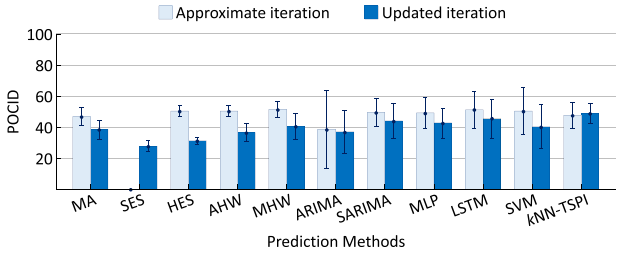



**Fig. 22.** Averages and SD of POCID obtained by the prediction methods on stochastic time series.

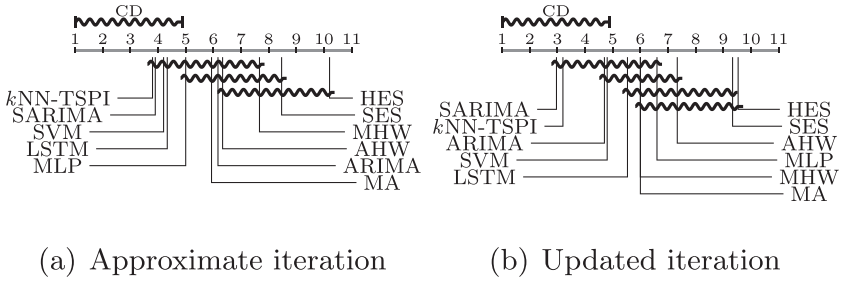



**Fig. 23.** MCPM over stochastic time series.

In Fig. 21, the ranges of values derived from TU coefficient evidence that SARIMA, with approximate and updated iterations, culminated in the most promising method to predict time series with stochastic behavior.

In Fig. 22, we can note that when employing approximate iteration, the highest hit rates were reached by MHW – 51.71% (SD = 5.39%) –, AHW – 50.73% (SD = 3.51%) –, and HES – 50.65% (SD = 3.41%). *k*NN-TSPI exhibited the eighth best result, *i.e.*, an average hit rate of 47.83% (SD = 8.46%). In contrast, for the updated iteration, it showed the highest average hit rate on the prediction horizons trends – 49.20% (SD = 6.47%). The poorest POCID values were obtained by SES – 27.94% (SD = 3.52%) –, HES – 31.47% (SD = 2.41%) –, and ARIMA – 37.02% (SD = 14.09%).

Fig. 23 covers the CD diagrams designed from the MCPM total area values. According to the multi-criteria analysis, SARIMA and *k*NN-TSPI were the best algorithms regardless of the employed projection strategy. Nevertheless, they did not present SSD regarding the ARIMA, MLP, LSTM, and SVM methods. Although *k*NN-TSPI has been more accurate, SARIMA provided the lowest error rates. This fact is due to SARIMA's own structure, which often includes an MA procedure that covers estimates of the innovation factor (white noise) that cannot be explained by the model.

### *6.1.3. Chaotic time series*

The computational tests conducted using chaotic data involved 176 configurations (11 predictors × 2 projection strategies × 8 datasets). As depicted in the CD diagrams of Fig. 24, the algorithms with approximate iteration MA, SVM, SES and LSTM showed, in increasing order and without SSD, the best results of MSE (Fig. 24(a)) and TU (Fig. 24(b)). The models with approximate iteration SVM, LSTM, and MLP assumed respectively the first, second, and third positions in the POCID average ranking (Fig. 24(c)).

Concerning the updated iteration, MLP and SVM achieved the best results in the three performance measures (Fig. 24 – (d), (e), and (f)). On the other hand, MA exhibited the highest prediction errors and the lowest hit rates on the projection horizons trends.

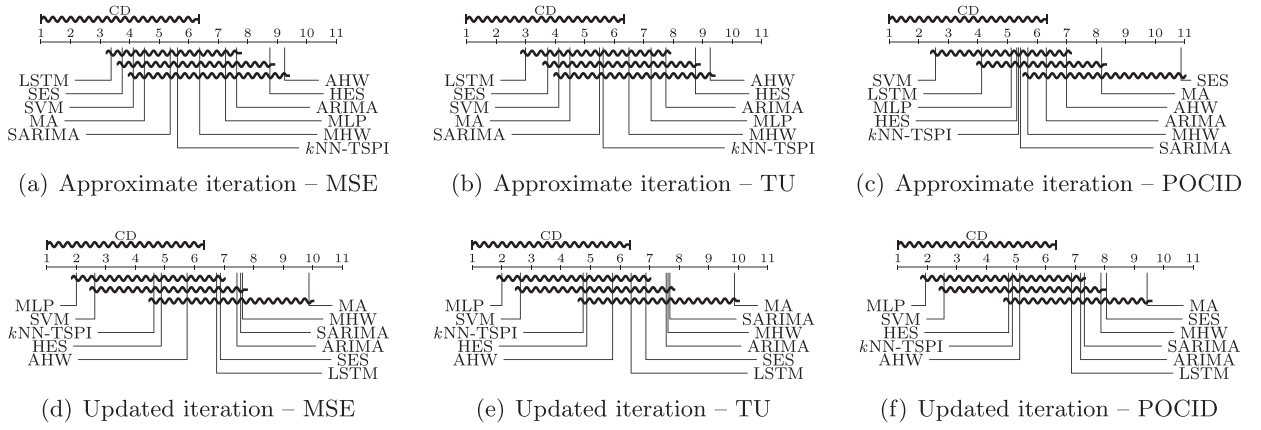



**Fig. 24.** CD diagrams for the MSE, TU, and POCID values coming from the predictors on chaotic time series.

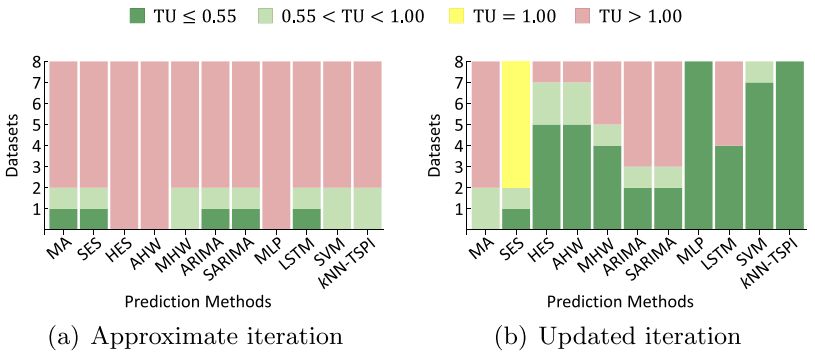



**Fig. 25.** Performance of the prediction methods for the four ranges of TU values in chaotic time series.

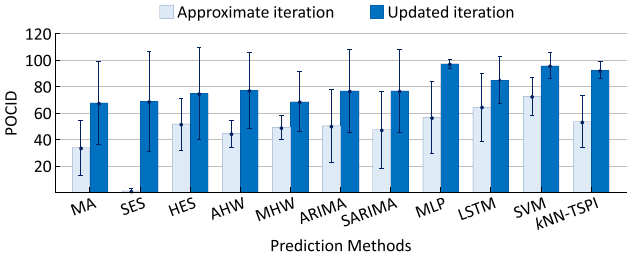


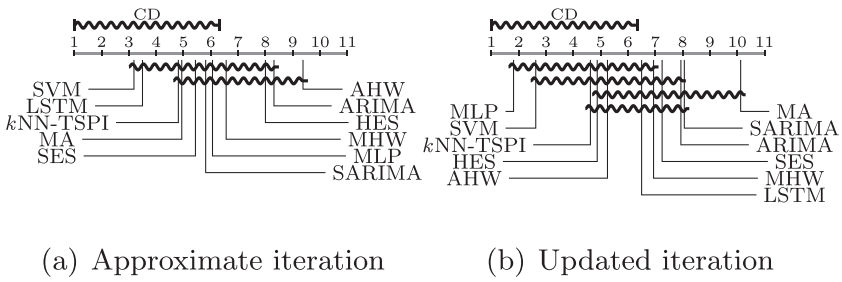

**Fig. 26.** Averages and SD of POCID obtained by the prediction methods on chaotic time series.

In Fig. 25, the statistics derived from TU coefficient show that, using the approximate iteration (Fig. 25(a)), the predictive models were convenient to predict, approximately, two of eight datasets (TU < 1). In this scenario, HES, AHW and MLP obtained, for all datasets, a performance lower than the trivial model (TU > 1). Such results reinforce the difficulty in predicting chaotic time series, especially when the employed projection strategy favors the error propagation along the prediction horizon.

Examining the updated iteration in Fig. 25(b), we can see that, in general, MLP, SVM and *k*NN-TSPI were reliable for modeling seven of eight datasets (TU ≤ 0.55). We expected such results since machine learning models are non-parametric. Although this premise is also valid for LSTM, we tend to have overfitting with such a network since it has many parameters and our datasets are not so large.

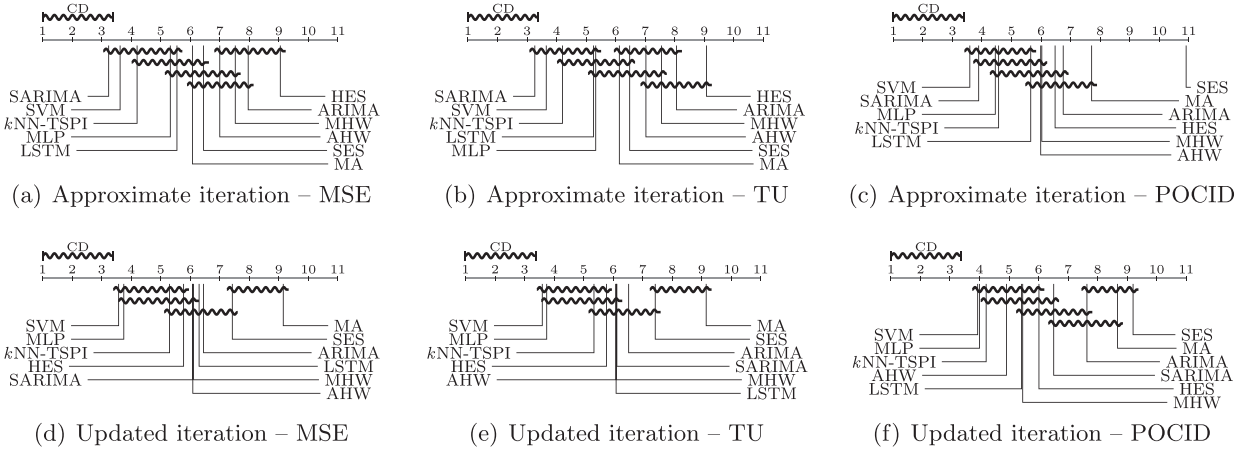
In Fig. 26, we expose the POCID averages and their respective SD. Analyzing the projection strategy with approximate iteration, SVM presented the best POCID values – 72.90% (SD = 14.47%) –, while SES – 0.98% (SD = 1.62%) – and MA – 33.76% (SD = 20.85%) – showed the poorest hit rates on the projection horizons trends. As for the projection strategy with updated iteration, the best POCID values were achieved by MLP – 97.41% (SD = 3.30%) –, followed by SVM – 95.93% (SD = 9.93%). The MA model maintained the poorest overall performance – 67.78% (SD = 31.16%).

Fig. 27 illustrates the CD diagrams with respect to the MCPM total area values. Inspecting both projection strategies, we can note that SVM and *k*NN-TSPI were more stable than the other methods for predicting chaotic data.




(a) Approximate iteration (b) Updated iteration

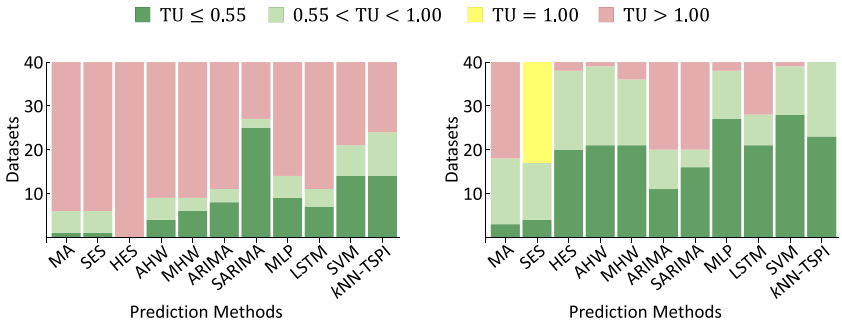
**Fig. 27.** MCPM over chaotic time series.




(a) Approximate iteration – MSE (b) Approximate iteration – TU (c) Approximate iteration – POCID

(d) Updated iteration – MSE (e) Updated iteration – TU (f) Updated iteration – POCID

**Fig. 28.** CD diagrams for the MSE, TU, and POCID values coming from the predictors on synthetic time series.




(a) Approximate iteration (b) Updated iteration

**Fig. 29.** Performance of the prediction methods for the four ranges of TU values in synthetic time series.

*6.1.4. Overall comparison*

After analyzing and discussing the results according to the different characteristics of the data, we are now in a position to interpreting the outcomes considering all the synthetic data. The experiments performed using all synthetic time series encompassed 880 configurations (11 predictors × 2 projection strategies × 40 datasets). The CD diagrams of Fig. 28 summarizes such results.

In agreement with these diagrams, SARIMA with approximate iteration showed the best values of MSE (Fig. 28(a)) and TU (Fig. 28(b)). Differently, when we examined the hit rates on the projection horizons trends (Fig. 28(c)), such configuration occupied the second position without SSD in comparison with SVM. Considering the updated iteration, SVM, MLP and *k*NN-TSPI assumed, in this order and by a small difference margin, the first, second and third positions in the rankings derived from MSE (Fig. 28(d)), TU (Fig. 28(e)), and POCID (Fig. 28(f)).

Looking at both projection strategies in Fig. 28, *k*NN-TSPI exhibited the third best performance for all evaluation measures, except when treated of the POCID index for the projection strategy with updated iteration. Besides, SVM was very competitive regarding the results obtained from SARIMA. Fig. 29 highlights this result in terms of TU values.

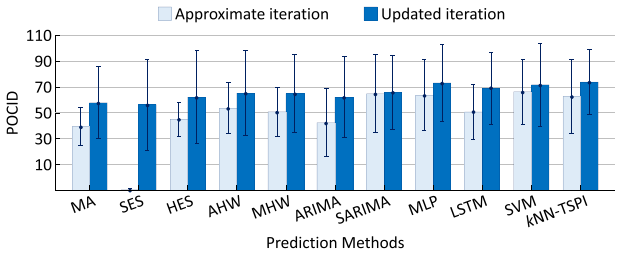



**Fig. 30.** Averages and SD of POCID obtained by the prediction methods on synthetic time series.

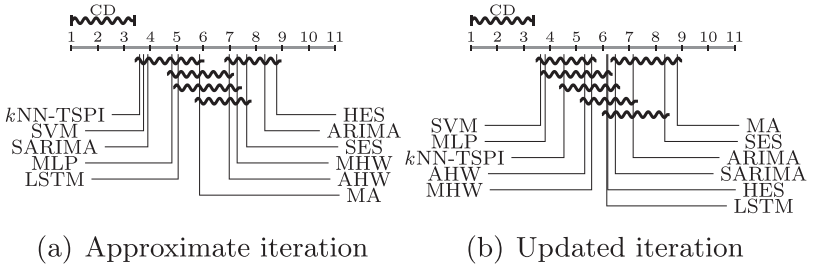



**Fig. 31.** MCPM over synthetic time series.

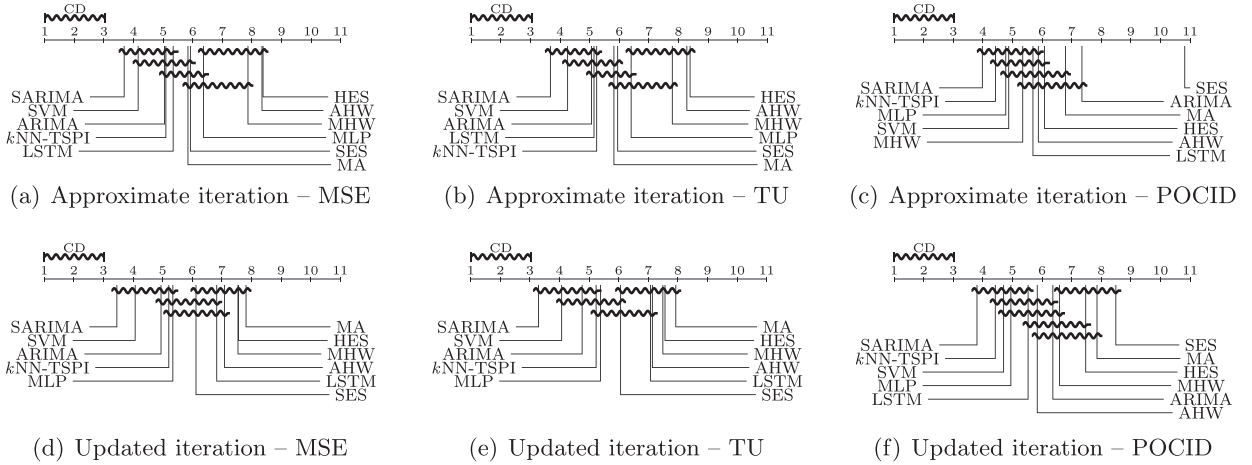



**Fig. 32.** CD diagrams for the MSE, TU, and POCID values coming from the predictors on real time series.

Analyzing the approximate iteration (Fig. 29(a)), SARIMA was adequate in 27 (25 + 2) datasets (TU < 1), which 25 were very well modeled (TU ≤ 0.55). In contrast, HES failed to outperform the naïve method for no one of the 40 datasets. Considering the updated iteration (Fig. 29(b)), the use of MLP, SVM and *k*NN-TSPI was appropriate for 39 datasets (TU < 1), which 26 resulted in reliable predictive models (TU ≤ 0.55).

Fig. 30 demonstrates that the machine learning methods MLP, SVM, and *k*NN-TSPI were able to achieve approximately 73.55% of precision (SD = 26.19%) on the projection horizons trends, regardless of the projection strategy adopted.

Fig. 31 contemplates the CD diagrams concerning the MCPM values considering all synthetic datasets. We can see that *k*NN-TSPI with approximate iteration achieved the best multi-criteria results (Fig. 31(a)), surpassing by a minimum difference SVM, SARIMA, MLP, and LSTM. On the other hand, SVM with updated iteration showed the smallest prediction errors and the highest hit rates on the projection horizons trends (Fig. 31(b)), being very competitive with MLP and *k*NN-TSPI. In relation to stability, the machine learning algorithms outperformed the state-of-the-art statistical methods.

### *6.2. Real data*

The computational tests conducted using real data totaled 1210 configurations (11 predictors × 2 projection strategies × 55 datasets). Fig. 32 presents the CD diagrams regarding the MSE, TU, and POCID measures obtained from the aforementioned experiments.

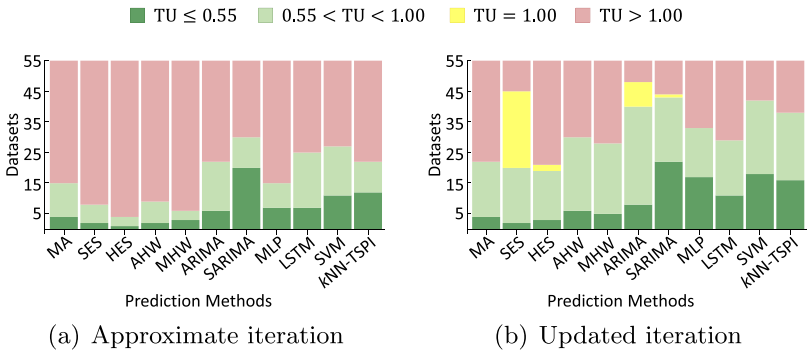



**Fig. 33.** Performance of the prediction methods for the four ranges of TU values in real time series.

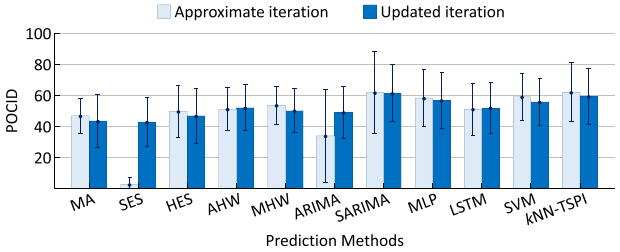



**Fig. 34.** Averages and SD of POCID obtained by the prediction methods on real time series.

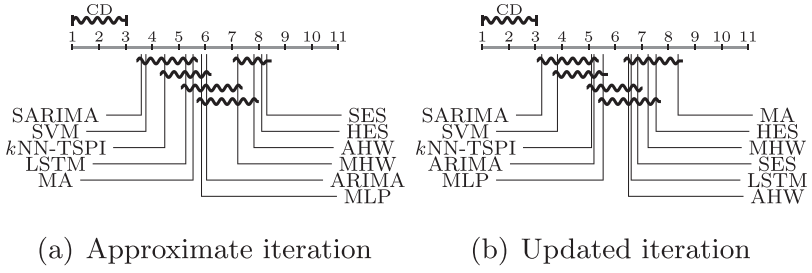



**Fig. 35.** MCPM over real time series.

In Fig. 32, we can note that SARIMA achieved the best results for both projection strategies given the three evaluated measures. In general, models based on simple exponential smoothing obtained the worst results, being that the MA method maintained the poorest overall performance. We did not verify SSD between the machine learning algorithms concerning the ARIMA and SARIMA models.

The four ranges of TU values indicated in Fig. 33 show that, using approximate iteration (Fig. 33(a)), SARIMA was adequate to predict 30 (20 + 10) of 55 datasets (TU < 1). The SVM algorithm was suitable for modeling 27 (11 + 16) time series, while the ARIMA and *k*NN-TSPI methods were preferable to the trivial model for 22 (12 + 10) of 55 datasets.

For the updated iteration (Fig. 33(b)), SARIMA was convenient (TU < 1) to predict 43 (22 + 21) datasets. Over this total, 22 provided a solid modeling which is reliable to make future projections (TU ≤ 0.55). SVM and *k*NN-TSPI were favorable (TU < 1) to predict the values of 42 (18 + 24) and 38 (16 + 22) datasets, respectively.

In Fig. 34 we display the POCID results for real time series. We can see that when using approximate iteration, *k*NN-TSPI obtained an average hit rate on the prediction horizons trends equivalent to 61.86% (SD = 18.83%), competing with SARIMA – 61.64% (SD = 26.29%) – and MLP – 58.07% (SD = 18.04%). SARIMA and *k*NN-TSPI, both with updated iteration, achieved the highest POCID values, *i.e.*, 61.27% (SD = 18.20%) and 59.11% (SD = 18.34%), respectively.

Fig. 35 portrays the CD diagrams designed from the MCPM total area values. SARIMA and SVM culminated in the best overall results. *k*NN-TSPI, in turn, also provided good results for both projection strategies and without SSD in comparison to SARIMA, MLP, and SVM.

### 6.3. Overall comparison

Our last comparison comprises all the 95 datasets, *i.e.*, involves synthetic and real datasets with high variability in their characteristics. We evaluated a total of 2090 configurations (11 predictors × 2 projection strategies × 95 datasets). The CD

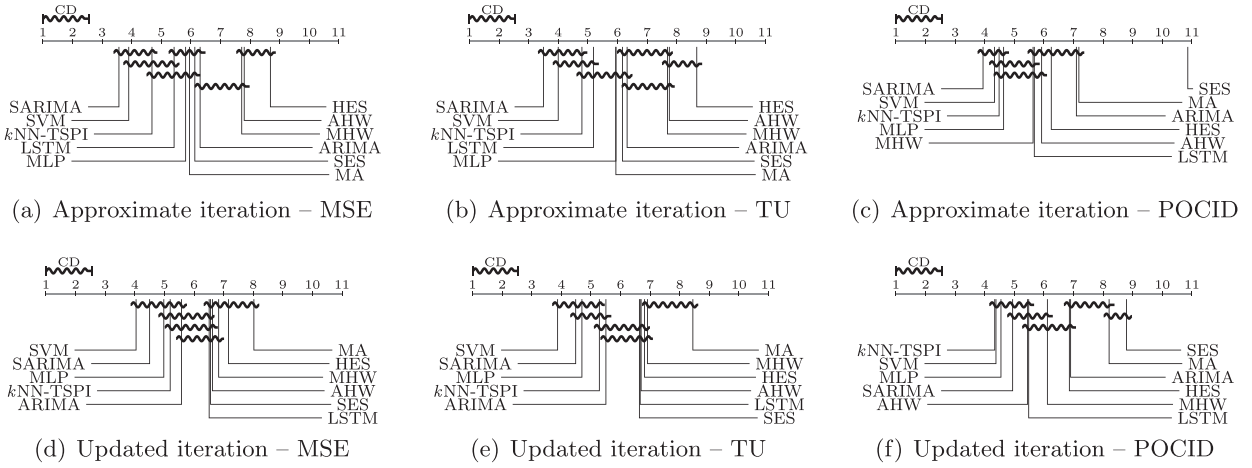



**Fig. 36.** CD diagrams for the MSE, TU, and POCID values coming from the predictors on synthetic and real time series.

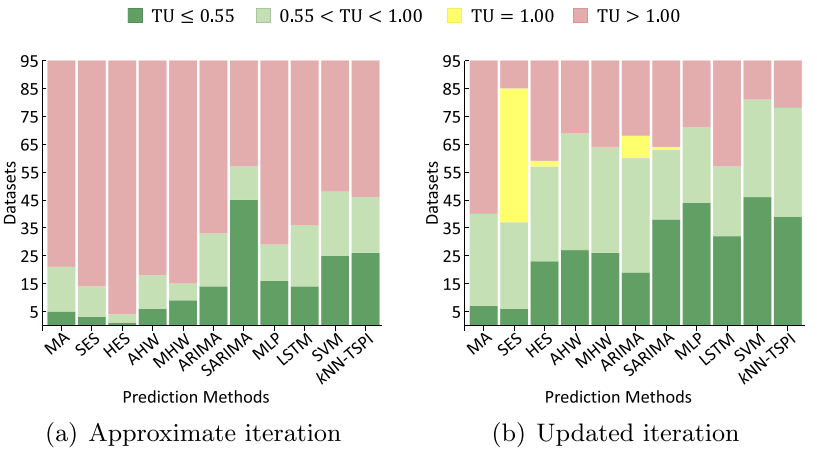



**Fig. 37.** Performance of the prediction methods for four ranges of TU values in synthetic and real time series.

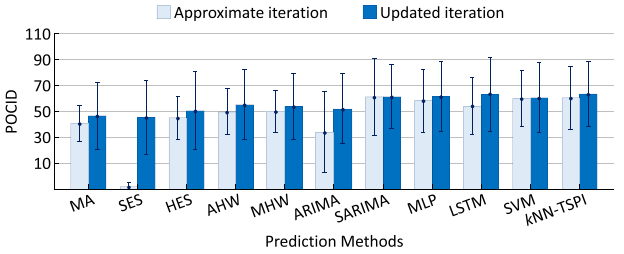
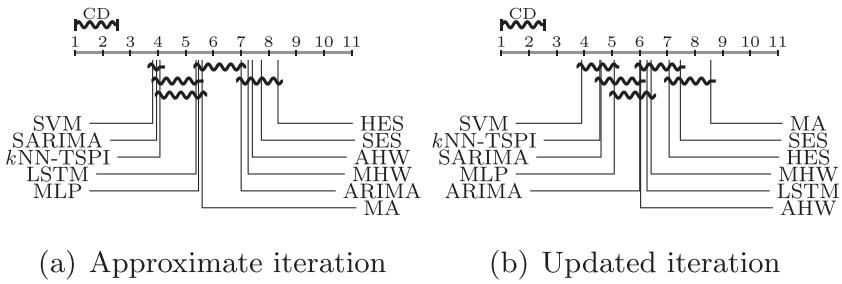



**Fig. 38.** Averages and SD of POCID obtained by the prediction methods on synthetic and real time series.

diagrams of Fig. 36 summarizes such results according to the MSE, TU, and POCID measures. In this figure, it is possible to notice how competitive the machine learning methods were in relation to the ARIMA and SARIMA state-of-the-art models.

In Fig. 37, the statistics derived from TU values show that, applying the approximate iteration (Fig. 37(a)), SARIMA was preferable to the naïve model in 57 (45 + 12) of 95 datasets (TU < 1). The performance of SVM and *k*NN-TSPI were better than the trivial model (TU < 1) in 48 (25 + 23) and 46 (26 + 20) datasets, respectively. For the updated iteration (Fig. 37(b)), SVM and *k*NN-TSPI were the algorithms that demonstrated the smallest projection errors for approximately 80 datasets (TU < 1).

From the graphic representation of the averages and SD of the POCID values in Fig 38, it is possible to verify that the hit rates over the projections horizons trends are slightly distributed among the SARIMA, MLP, SVM, and *k*NN-TSPI models. Each one of these algorithms, regardless of the projection strategy employed, recorded a POCID average of approximately 64.89% (SD = 23.33%).




(a) Approximate iteration (b) Updated iteration

**Fig. 39.** MCPM over synthetic and real time series.

**Table 4**
Prediction algorithms and their properties.

| Algorithm | Prediction approach | #P | Technique for parameter estimation | $h > 1$ | Non-stationarity | $T$ | $S$ |
|---|---|---|---|---|---|---|---|
| MA | Parametric | 1 | Holdout validation | ✓ | | | |
| SES | Parametric | 1 | Holdout validation | | | | |
| HES | Parametric | 2 | Holdout validation | ✓ | | ✓ | |
| AHW | Parametric | 4 | Holdout validation | ✓ | | ✓ | ✓ |
| MHW | Parametric | 4 | Holdout validation | ✓ | | ✓ | ✓ |
| ARIMA | Parametric | 4 | Box-Jenkins method | ✓ | ✓ | ✓ | |
| SARIMA | Parametric | 7 | Box-Jenkins method | ✓ | ✓ | ✓ | ✓ |
| MLP | Non-parametric | 5 | Cross-validation | ✓ | ✓ | ✓ | ✓ |
| LSTM | Non-parametric | 6 | Holdout validation | ✓ | ✓ | ✓ | ✓ |
| SVM | Non-parametric | 3 | Cross-validation | ✓ | ✓ | ✓ | ✓ |
| *k*NN-TSPI | Non-parametric | 2 | Holdout validation | ✓ | ✓ | ✓ | ✓ |

Fig. 39 illustrates the CD diagrams with respect to the MCPM total area values for synthetic and real data. The outcomes of this multi-criteria analysis indicate that SARIMA, MLP, SVM, and *k*NN-TSPI are the most promising methods for time series modeling and prediction. Such fact is in line with our initial hypothesis, *i.e.*, that machine learning algorithms offer results similar to or better than those reached by state-of-the-art statistical methods with fewer parameters and without the requirement of *a priori* knowledge of data distribution.

## 7. Limitations, recommendations, and practical implications of the outcomes

We limited our study to predictors that deal with univariate data. Such methods receive as input unidimensional time series without considering possible explanatory variables. In future works, we want to explore the multivariate scenario which still constitutes an important gap in the literature.

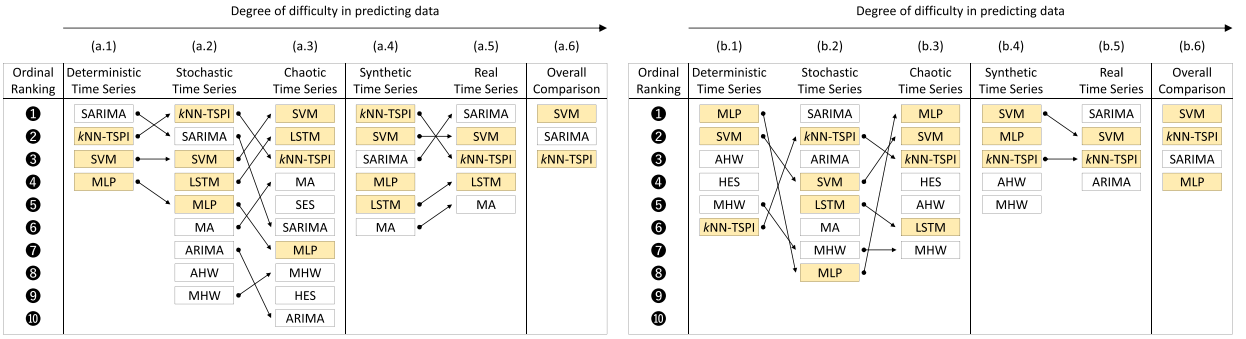
Table 4 summarizes the main characteristics of the statistical and machine learning models discussed and evaluated in this paper. The table exhibits the following information: the algorithm's name, the prediction approach, the number of parameters (**#P**) required by each predictor, the sampling method for the parameter estimation, if the algorithm supports the multi-step-ahead projection strategy ($\boldsymbol{h > 1}$), and if the method is prepared to deal with non-stationarity, trend ($\boldsymbol{T}$), and seasonality ($\boldsymbol{S}$). Note that the computational complexity is empirically related to the number of parameters of a model. Thus, SARIMA is the most expensive and complex among the evaluated methods.

Fig. 40 summarizes the key results of this work. This figure shows, for each time series group, the rank position of the predictors given the multi-criteria performance measure. Such illustration can be an interesting tool to guide practitioners and researchers to choose the most promising method according to the data characteristics and the type of multi-step-ahead projection strategy desired.

The multi-step-ahead projection with approximate iteration faces some difficulties, such as reduced performance and increase uncertainty given the error accumulation. These problems become more visible as the prediction horizon grows and the predicted values are not replaced by the actual values observed. However, in most of the real applications, it is the only strategy feasible to predict large horizons.

In agreement with the aforementioned projection strategy, SARIMA is the candidate to the most promising method to model and predict deterministic time series (Fig. 40(a.1)), followed by *k*NN-TSPI, SVM, and MLP. These four algorithms also obtained the highest POCID values. Machine learning models have an advantage over more complex methods because the use of default parameters can generate suitable predictive models. This would not be possible for methods like SARIMA and ARIMA.

Since *k*NN-TSPI and SARIMA presented very close results, they are candidates to the most appropriate model for stochastic time series (Fig. 40(a.2)). Coincidentally, AHW and MHW achieved the highest POCID values. Also, MHW did not present SSD compared to *k*NN-TSPI, SARIMA, SVM, LSTM, MLP, MA, and ARIMA. It is important to highlight that ARIMA and SARIMA




(a) Multi-step-ahead with approximate iteration

(b) Multi-step-ahead with updated iteration

**Fig. 40.** Overview of our findings.

usually outperform exponential smoothing algorithms when the time series are relatively long and well-behaved. If the data present many irregularities, the results of ARIMA and SARIMA will be lower than those obtained by exponential smoothing models.

SVM was the more stable method for chaotic series (Fig. 40(a.3)). Note that SVM did not show SSD when compared with *k*NN-TSPI, reinforcing the competitiveness of similarity-based algorithms. This fact becomes even more evident in the evaluation with 40 synthetic datasets (Fig. 40(a.4)): *k*NN-TSPI exhibited the best multi-criteria performance, remaining without SSD concerning SVM, SARIMA, MLP, LSTM, and MA. If on the one hand, LSTM performs well when there are lots of training data compared to the number of weights to be learned, on the other hand, we do not recommend its use in scenarios with little data because of the high risk of overfitting.

SARIMA and SVM, followed by *k*NN-TSPI, LSTM and MA, demonstrated the best results on real datasets, as we can see in Fig. 40(a.5). *k*NN-TSPI is the best candidate to model and predict real time series, given it obtained POCID hit rates greater than SARIMA and SVM.

SVM, SARIMA and *k*NN-TSPI exposed, in this order, the best multi-criteria results when executed over the 95 datasets (Fig. 40(a.6)). It is relevant to observe how the error eventually propagated along the projection horizon influences the performance of MLP and ARIMA.

In applications where the data acquisition period is not extreme, we can use the multi-step-ahead projection with updated iteration. These applications acquire the observations at a reasonable deadline which allows to feed (or update) the model continuously over time. For this projection strategy, MLP is a proper candidate to model and predict deterministic time series (Fig. 40(b.1)). Beyond to present the smallest prediction errors, it was able to hit the extrapolated horizons trend accurately. Also note that MLP did not show SSD regarding two machine learning algorithms (SVM and *k*NN-TSPI) and three exponential smoothing methods (AHW, HES and MHW). In this scenario, despite the competitiveness of the exponential smoothing models, the machine learning algorithms culminated in the best POCID rates.

SARIMA and *k*NN-TSPI seem to be the most appropriate models for dealing with stochastic time series (Fig. 40(b.2)). *k*NN-TSPI still stands out for having achieved the highest POCID values. It is important to emphasize that SARIMA and *k*NN-TSPI did not present SSD concerning three parametric (ARIMA, MA and MHW) and three non-parametric (SVM, LSTM and MLP) methods.

We can consider MLP, SVM and *k*NN-TSPI the most suitable algorithms to model and predict chaotic time series (Fig. 40(b.3)). MLP obtained the smallest errors and the best POCID rates, followed by SVM and *k*NN-TSPI. We observe a similar behavior on the 40 synthetic datasets (Fig. 40(b.4)), where SVM, MLP and *k*NN-TSPI were the most appropriate methods both concerning the low prediction error and the high hit rates on the projection horizons trends.

SARIMA, SVM, *k*NN-TSPI, and ARIMA are the most promising algorithms for real time series (Fig. 40(b.5)). ARIMA's performance was very close to *k*NN-TSPI. Also, SARIMA and *k*NN-TSPI showed the highest POCID values.

For all the 95 datasets (Fig. 40(b.6)), SVM surpassed *k*NN-TSPI and SARIMA by a small margin. These three methods did not exhibit large performance differences regarding MLP. Besides, they achieved similar POCID results.

Considering both projection strategies, SVM was the algorithm that obtained, regarding prediction error and POCID rates, the most stable performance for deterministic time series (Fig. 40 – (a.1) and (b.1)). In contrast, SARIMA and *k*NN-TSPI were the best models for stochastic data (Fig. 40 – (a.2) and (b.2)). Although *k*NN-TSPI was more accurate concerning the projection horizons trends, SARIMA provided the lowest error rates. Regarding the chaotic time series (Fig. 40 – (a.3) and (b.3)), SVM and *k*NN-TSPI were more stable than the other predictors.

The learning algorithms *k*NN-TSPI and SVM outperformed the state-of-the-art statistical methods (SARIMA and ARIMA) when examining all the 40 synthetic datasets (Fig. 40 – (a.4) and (b.4)).

Given the 55 real datasets (Fig. 40 – (a.5) and (b.5)), SARIMA and SVM showed very similar performances. *k*NN-TSPI also provided good results for both projection strategies and without SSD in comparison to SARIMA and SVM.

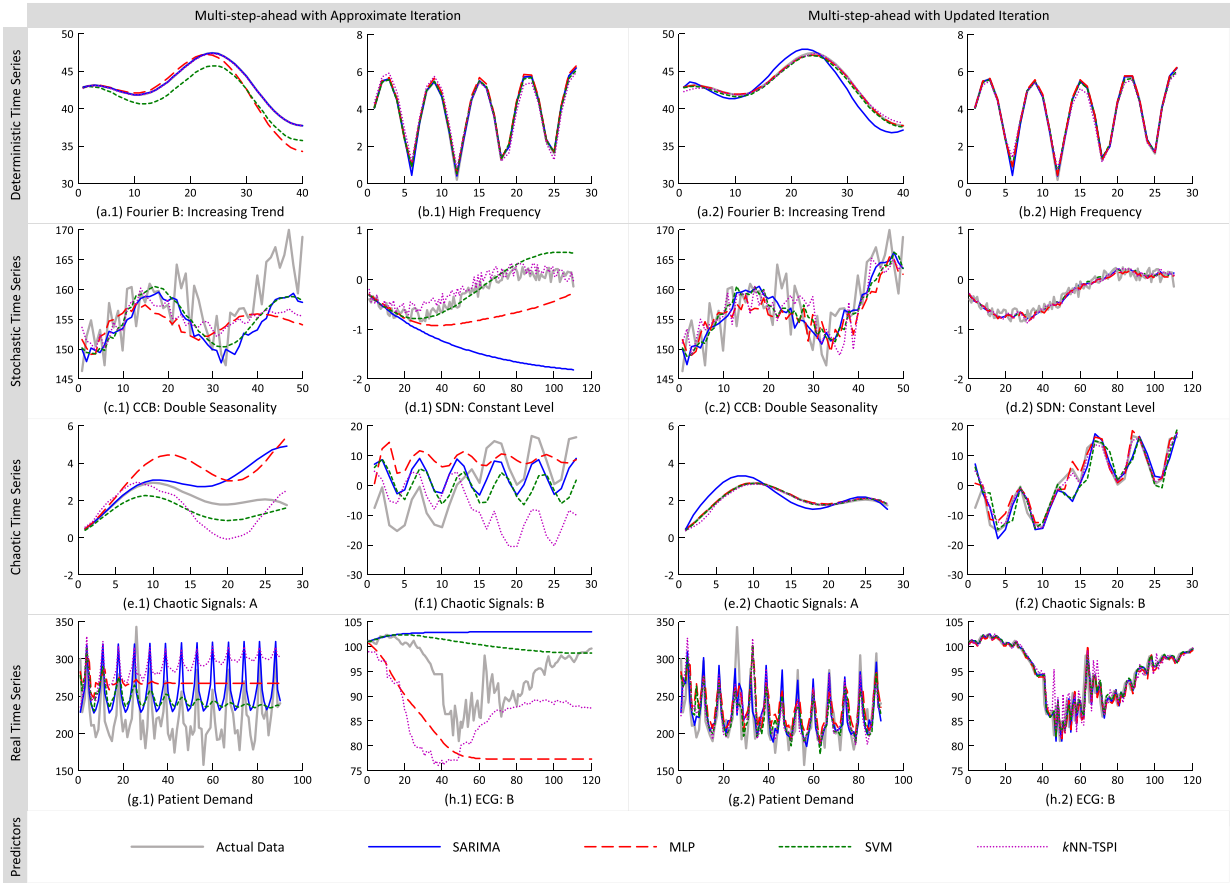



**Fig. 41.** Predictions obtained for the out of sample period.

On the overall comparison involving 95 datasets (Fig. 40 – (a.6) and (b.6)), the multi-criteria analysis indicates that SARIMA, SVM and *k*NN-TSPI are the most promising methods for temporal data modeling and prediction.

To compare the prediction quality of SARIMA, MLP, SVM, and *k*NN-TSPI, we show eight examples in Fig. 41. We chose these time series to illustrate some cases where the modeling is challenging.

In Fig. 41, we can see the impact of error propagation in the multi-step-ahead projection with approximate iteration when compared to the updated iteration. Among the four algorithms, *k*NN-TSPI stands out for its robustness and stability concerning the projection horizons trends. An interesting case where the algorithm performed very well is the example illustrated in Fig. 41(d.1).

Although SARIMA and SVM have achieved about the same prediction accuracy as the similarity-based method, the algorithm with invariances is simpler to understand, encode, and adjust. While SARIMA has seven parameters and SVM has three, *k*NN-TSPI have only two. Most importantly, the two input arguments of *k*NN-TSPI are intuitive and can be easily estimated based on the data seasonality. The first parameter ($l$) is the query length in number of observations, and the second ($k$) is the number of similar subsequences required to make a prediction.

The use of baseline and topline methods is an essential guideline for conducting empirical assessments. In this direction, we suggest MA and HW as baseline models. MA is quite simple and generally performs better than the naïve (one-step-ahead) technique. HW methods, besides being indicated for time series with trend and seasonality, show reasonable results concerning the projections horizons trends. In respect to topline models, we emphasize the performance of SARIMA, SVM, and *k*NN-TSPI.

We are certain that the study of machine learning methods is right now at the same maturity stage as temporal data modeling researchers using statistical models. From the practical point of view, the results put forward in this work will serve as a reference for the advance of time series prediction field.

## 8. Conclusion

The comprehensive review performed in this work allowed us to identify an important gap in the literature: the lack of an objective comparison between parametric and non-parametric models for time series prediction. In this sense, to present

a relevant contribution to the areas of statistics and machine learning, this paper investigated several methods from both fields for univariate time series prediction.

Our study focused on the presentation of the most popular predictive algorithms and their confrontation considering 95 datasets. To the best of our knowledge, we not only conducted one of the most extensive experimental evaluations ever done for time series prediction, but we also carry out a broad analysis of 2090 results.

We detailed discuss all procedures related to predictive model induction and evaluation, from its conception to its extrapolation. The empirical assessment carry out sought to meet two goals: (i) to make feasible the consolidation of the efficiency and effectiveness of each method analyzed; and (ii) to characterize the different algorithms for model building, describing the advantages and disadvantages of the use of each one.

Besides recommendations of predictors to be used as baseline and topline, we also highlighted the benefits of our experimental protocol and how it can be used as a reference for models selection, parameters setting, and the employment of statistical and machine learning methods for time series prediction.

The datasets considered in this work, as well as the estimated parameters and the results achieved, are available at the ICMC-USP Time Series Prediction Repository [32]. We developed this online archive to provide open access to all the materials employed and produced in this study. Such repository is an initiative that aims to encourage the reproduction of our results and which will support more rigorous evaluations of new prediction algorithms.

The outcomes from the overall comparison that covered 95 datasets (40 synthetic time series and 55 real time series) indicate that SARIMA, SVM and *k*NN-TSPI are the most promising methods for temporal data modeling and prediction. Such fact is in line with our initial hypothesis, *i.e.*, that machine learning algorithms offer results similar to or better than those reached by state-of-the-art statistical methods. The similarity-based algorithm with invariances, proposed by us in [33], still contemplates the advantages of being easy to explain, adjust, and embed into any device.

As future works, we intend to explore the properties inherent to similarity-based time series prediction, such as invariances to distortions in temporal data, distance measures, complexity estimates applied to complexity-invariant distances, and prediction functions. An empirical analysis of these particularities will allow a better understanding of the predictive performance of similarity-based methods and the conclusions drawn may guide future research with the *k*NN-TSPI algorithm.

## Acknowledgments

This work was supported by the São Paulo Research Foundation [grant numbers 2013/10978-8, 2016/04986-6, and 2018/05859-3]; and the Brazilian National Council for Scientific and Technological Development [grant number 306631/2016-4].

## Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2019.01.076.

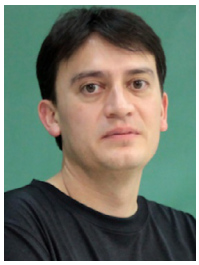
## References

[1] N.K. Ahmed, A.F. Atiya, N.E. Gayar, H. El-Shishiny, An empirical comparison of machine learning models for time series forecasting, Econ. Rev. 29 (5–6) (2010) 594–621.
[2] R.R. Andrawis, A.F. Atiya, H. El-Shishiny, Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition, Int. J. Forecast. 27 (3) (2011) 672–688.
[3] S. Ben Taieb, G. Bontempi, A.F. Atiya, A. Sorjamaa, A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition, Expert Syst. Appl. 39 (8) (2012) 7067–7083.
[4] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw.s 5 (2) (1994) 157–166.
[5] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, Time Series Analysis: Forecasting and Control, Wiley Series in Probability and Statistics, 5, Wiley, New Jersey, USA, 2015.
[6] R. Carbonneau, K. Laframboise, R. Vahidov, Application of machine learning techniques for supply chain demand forecasting, Eur. J. Oper. Res. 184 (3) (2008) 1140–1154.
[7] M.-Y. Chen, B.-T. Chen, A hybrid fuzzy time series model based on granular computing for stock price forecasting, Inf. Sci. 294 (2015) 227–241.
[8] O. Claveria, S. Torra, Forecasting tourism demand to {catalonia}: {neural} networks {vs}. Time series models, Econ. Model. 36 (C) (2014) 220–228.
[9] P. Cortez, Sensitivity analysis for time lag selection to forecast seasonal time series using neural networks and support vector machines, in: International Joint Conference on Neural Networks, IEEE, Barcelona, Spain, 2010, pp. 3694–3701.
[10] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (4) (1989) 303–314.
[11] J. de Jesús Rubio, A method with neural networks for the classification of fruits and vegetables, Soft Comput. 21 (23) (2017) 7207–7220.
[12] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[13] J.L. Elman, Finding structure in time, Cognit. Sci. 14 (2) (1990) 179–211.
[14] T. Fu, A review on time series data mining, Eng. Appl. Artif. Intell. 24 (1) (2011) 164–181.
[15] E.S. Gardner, Exponential smoothing: the state of the art, J. Forecast. 4 (1) (1985) 1–28.
[16] J.G.D. Gooijer, R.J. Hyndman, 25 Years of time series forecasting, Int. J. Forecast. 22 (3) (2006) 443–473.
[17] S.R. Gunn, et al., Support vector machines for classification and regression, Technical Report, Faculty of Engineering, Science and Mathematics, University of Southampton, Southampton, UK, 1998.
[18] H. Hassani, E.S. Silva, N. Antonakakis, G. Filis, R. Gupta, Forecasting accuracy evaluation of tourist arrivals, Ann. Tourism Res. 63 (2017) 112–127.
[19] S.S. Haykin, Neural Networks and Learning Machines, third ed., Prentice Hall, Upper Saddle River, USA, 2009.
[20] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, J. Uncertain. FuzzinessKnowl.-Based Syst. 6 (2) (1998) 107–116.
[21] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[22] C.C. Holt, Forecasting seasonals and trends by exponentially weighted moving averages, Int. J. Forecast. 20 (1) (2004) 5–10.
[23] R.J. Hyndman, A.B. Koehler, R.D. Snyder, S. Grose, A state space framework for automatic forecasting using exponential smoothing methods, Int. J. Forecast. 18 (3) (2002) 439–454.
[24] M. Islam, B. Sivakumar, Characterization and prediction of runoff dynamics: a nonlinear dynamical view, Adv. Water Resour. 25 (2) (2002) 179–190.
[25] K. Kandananond, A comparison of various forecasting methods for autocorrelated time series, Int. J. Eng. Bus.Manag. 4 (2012) 1–6.
[26] C. Lemke, B. Gabrys, Meta-learning for time series forecasting and forecast combination, Neurocomputing 73 (10–12) (2010) 2006–2016.
[27] R.P. Lippmann, An introduction to computing with neural nets, IEEE ASSP Mag. 4 (2) (1987) 4–22.
[28] C. Lu, Wavelet fuzzy neural networks for identification and predictive control of dynamic systems, IEEE Trans. Ind. Electron. 58 (7) (2011) 3046–3058.
[29] J.M. Lucas, M.S. Saccucci, Exponentially weighted moving average control schemes: properties and enhancements, Technometrics 32 (1) (1990) 1–12.
[30] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[31] D.C. Montgomery, C.L. Jennings, M. Kulahci, Introduction to Time Series Analysis and Forecasting, Wiley Series in Probability and Statistics, 2, Wiley, New Jersey, USA, 2015.
[32] A.R.S. Parmezan, G.E.A.P.A. Batista, ICMC-USP Time Series Prediction Repository, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brasil, 2014.
[33] A.R.S. Parmezan, G.E.A.P.A. Batista, A study of the use of complexity measures in the similarity search process adopted by kNN algorithm for time series prediction, in: International Conference on Machine Learning and Applications, IEEE, Miami, USA, 2015, pp. 45–51.
[34] A.R.S. Parmezan, H.D. Lee, F.C. Wu, Metalearning for choosing feature selection algorithms in data mining: proposal of a new framework, Expert Syst. Appl. 75 (2017) 1–24.
[35] M. Pratama, E. Lughofer, M.J. Er, S. Anavatti, C.-P. Lim, Data driven modelling based on recurrent interval-valued metacognitive scaffolding fuzzy neural network, Neurocomputing 262 (2017) 4–27.
[36] M. Rafiei, T. Niknam, M.-H. Khooban, Probabilistic forecasting of hourly electricity price by generalization of ELM for usage in improved wavelet neural network, IEEE Trans. Ind. Inf. 13 (1) (2017) 71–79.
[37] G. Ristanoski, W. Liu, J. Bailey, A time-dependent enhanced support vector machine for time series regression, in: International Conference on Knowledge Discovery and Data Mining, ACM, Chicago, USA, 2013, pp. 946–954.
[38] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychol. Rev. 65 (6) (1958) 386–408.
[39] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1, MIT Press, Cambridge, USA, 1986, pp. 318–362.
[40] N.I. Sapankevych, R. Sankar, Time series prediction using support vector machines: a survey, Comput. Intell. Mag. 4 (2) (2009) 24–38.
[41] J. Schiller, M. Spiegel, R. Srinivasan, Schaum'S Outline of Probability and Statistics, McGraw-Hill, New York, USA, 2012.
[42] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[43] H.T. Siegelmann, B.G. Horne, C.L. Giles, Computational capabilities of recurrent NARX neural networks, IEEE Trans. Syst. Man Cybern. 27 (2) (1997) 208–215.
[44] J.W. Taylor, A comparison of univariate time series methods for forecasting intraday arrivals at a call center, Manag. Sci. 54 (2) (2008) 253–265.
[45] V.N. Vapnik, The Nature of Statistical Learning Theory, Information Science and Statistics, second ed., Springer Science & Business Media, New York, USA, 1999.
[46] L. Wang, Y. Zeng, T. Chen, Back propagation neural network with adaptive differential evolution algorithm for time series forecasting, Expert Syst. Appl. 42 (2) (2015) 855–863.
[47] P.R. Winters, Forecasting sales by exponentially weighted moving averages, Manag. Sci. 6 (3) (1960) 324–342.
[48] B. Yu, X. Song, F. Guan, Z. Yang, B. Yao, K-Nearest Neighbor model for multiple-time-step prediction of short-term traffic condition, J. Transp. Eng. 142 (6) (2016) 1–10.
[49] J. Zhang, G.M. Lee, J. Wang, A comparative analysis of univariate time series methods for estimating and forecasting daily spam in United States, in: Twenty-second Americas Conference on Information Systems, 2016, pp. 1–10. San Diego, USA
[50] X. Zhang, T. Zhang, A.A. Young, X. Li, Applications and comparisons of four time series models in epidemiological surveillance data, PLoS One 9 (2) (2014) e88075.

**Antonio Rafael Sabino Parmezan** is a Ph.D. student of Computer Science and Computational Mathematics at Universidade de São Paulo (USP), Brazil. He holds an M.Sc. degree in Computer Science and Computational Mathematics in USP and a B.Sc. in Computer Science from Universidade Estadual do Oeste do Paraná (2016 and 2012), Brazil. His research interests include machine learning, time series processing and analysis, data mining, data streams, feature selection, and metalearning.

**Vinicius M. A. Souza** is a Postdoctoral Research Fellow at Universidade de São Paulo (USP), Brazil. He is Ph.D in Computer Science and Computational Mathematics (2016) at USP. He holds a B.Sc. degree in Informatics (2008) and an M.Sc. degree in Computer Science (2011) from Universidade Estadual de Maringá, Brazil. His research interests include data mining, data streams, signal processing, telematics, and time series.



**Gustavo E. A. P. A. Batista** received M.Sc. and Ph.D. degrees in Computer Science from Universidade de São Paulo at São Carlos. In 2007, he joined the Instituto de Ciências Matemáticas e de Computação as an assistant professor and became an associate professor in 2016. From 2010 to 2011, he was a visiting researcher at University of California, Riverside. Dr. Batista has more than 100 papers in peer-reviewed conference and journals and more than 4,800 citations. He has served as program committee member of top-tier conferences such as ACM-KDD, SIAM-SDM, IEEE-ICDM and IJCAI, and as a member of the editorial board of the Machine Learning Journal. His research has been funded byseveral agencies such as FAPESP, CNPq, USAID, and Google. He is a CNPq fellow researcher since 2013. His research interests include machine learning, data mining and time series and data stream processing with applications in music retrieval, public health, agriculture, and the environment.