IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 June 2022, accepted 11 July 2022, date of publication 21 July 2022, date of current version 28 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192849

RESEARCH ARTICLE

Time Series Prediction via Similarity Search: Exploring Invariances, Distance Measures and Ensemble Functions

ANTONIO R. S. PARMEZAN[1], VINICIUS M. A. SOUZA[2], AND GUSTAVO E. A. P. A. BATISTA[3]

[1]Institute of Mathematics and Computer Science, University of São Paulo, São Carlos 13566-590, Brazil
[2]Graduate Program in Informatics, Pontifícia Universidade Católica do Paraná, Curitiba 80215-901, Brazil
[3]School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia

Corresponding author: Antonio R. S. Parmezan (parmezan@usp.br)

This work was supported in part by the São Paulo Research Foundation under Grant 2013/10978-8, Grant 2016/04986-6, and Grant 2018/05859-3; and in part by the Brazilian National Council for Scientific and Technological Development under Grant 306631/2016-4, Grant 140159/2017-7, and Grant 142050/2019-9.

ABSTRACT The rapid advance of scientific research in data mining has led to the adaptation of conventional pattern extraction methods to the context of time series analysis. The forecasting (or prediction) task has been supported mainly by regression algorithms based on artificial neural networks, support vector machines, and *k*-Nearest Neighbors (*k*NN). However, some studies provided empirical evidence that similarity-based methods, *i.e.* variations of *k*NN, constitute a promising approach compared with more complex predictive models from both machine learning and statistics. Although the scientific community has made great strides in increasing the visibility of these easy-to-fit and impressively accurate algorithms, previous work has failed to recognize the right invariances needed for this task. We propose a novel extension of *k*NN, namely *k*NN - Time Series Prediction with Invariances (*k*NN-TSPI), that differs from the literature by combining techniques to obtain amplitude and offset invariance, complexity invariance, and treatment of trivial matches. Our predictor enables more meaningful matches between reference queries and data subsequences. From a comprehensive evaluation with real-world datasets, we demonstrate that *k*NN-TSPI is a competitive algorithm against two conventional similarity-based approaches and, most importantly, against 11 popular predictors. To assist future research and provide a better understanding of similarity-based method behaviors, we also explore different settings of *k*NN-TSPI regarding invariances to distortions in time series, distance measures, complexity-invariant distances, and ensemble functions. Results show that *k*NN-TSPI stands out for its robustness and stability both concerning the parameter k and the accuracy of the projection horizon trends.

INDEX TERMS Forecasting, multi-step-ahead prediction, pattern sequence similarity, univariate analysis.

I. INTRODUCTION

Time series data are omnipresent with applications in computing, engineering, finances, medicine, and many other areas of knowledge. Especially in the last 20 years, where domain-driven data mining has become more prominent in science and industry, machine learning regression methods have extensively explored time series modeling and prediction[1] [1]–[3].

In contrast to parametric predictors from classical statistics, such as Autoregressive Integrated Moving Average (ARIMA) and Seasonal ARIMA (SARIMA) [4], machine learning algorithms require no assumptions about the data distribution nature. This characteristic makes them more straightforward to adjust to complex data, especially

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

[1]The terms "prediction" and "forecast" are employed interchangeably herein.

those with nonlinear behavior. However, what makes non-parametric methods even more attractive is the fact that their adjustment needs little human intervention and interpretation.

Non-parametric algorithms typically use two approaches to model and predict time series [5]: (i) global, which builds a predictive model from a training procedure that uses all the series observations as input; and (ii) local, which splits the time series into subsequences. A selection criterion chooses a subset of the subsequences to predict future observations.

Recently, some studies have evidenced the competitive performance and the simplicity of the local approach methods to predict series with distinct properties and components [1], [6], [7]. Most of these researchers have employed similarity-based models, *i.e.*, variations of the *k*-Nearest Neighbors (*k*NN) algorithm. The idea behind *k*NN is finding the k most similar examples to an unlabeled example given a proper distance measure. For the classification task, the labels of those k nearest examples will determine the output of the new example [8].

For the prediction task, a generic local *k*NN method identifies the k most similar subsequences to a reference query and then uses the next values of these subsequences (targets) as input to an ensemble function, which computes the future value of the series. This idea is quite intuitive since it assumes that patterns that emerged in the past are likely to repeat themselves in the future.

Although *k*NN has garnered much attention for temporal data prediction due to its simplicity, some research questions remain open. The first one refers to their sensitivity to the number of neighbors k and the sliding window (or reference query) length l. The second one is related to choosing an adequate similarity measure to compare subsequences. The challenge is efficiently searching similar patterns from large series with low computational costs. The third one involves determining the number of neighbors, which must be data-driven rather than a fixed given number. The last one covers choosing an ensemble function that enables issuing reliable predictions.

Despite the incessant work of the machine learning community in sampling [9] and calibration methods [10], [11], distance measures [12]–[15], and ensemble functions [16], [17], we are confident that previous studies did not identify the right invariances for the similarity-based prediction task. To mitigate such an issue, we propose a novel modification of the *k*NN algorithm for temporal data forecasting, namely *k*NN - Time Series Prediction with Invariances (*k*NN-TSPI). Our method leads to more accurate and meaningful results by combining techniques to obtain amplitude and offset invariance, complexity invariance, and treatment of trivial matches. The proper integration of these techniques makes *k*NN-TSPI invariant to temporal distortions, allowing it to discard information that does not favor matching perceptually similar subsequences in the search process.

Parmezan and Batista [7] is the first work to describe the core idea of *k*NN-TSPI. The present paper gives more technical details on its properties and performance in an extensive experimental evaluation. Unlike black-box models, *k*NN-TSPI is simple to encode and embed into any device. Moreover, considering the state-of-the-art predictors, the proposed method is highly competitive with the advantage of having parameters that can be easily estimated only by observing the data seasonality.

The main contributions that distinguish this article from our previous study include:

- Formalization of a rigorous experimental setup. Our assessment considers two multi-step-ahead projection strategies on 55 real datasets with four performance evaluation indexes and statistical significance tests. Supported by this assessment, we present a reasoned discussion of best practices for similarity-based time series prediction;
- Comparison of the developed method with previous similarity-based predictors. We compare *k*NN-TSPI with *k*NN according to the global and local approaches. We analyze the results regarding predictive performance, parameter sensitivity, and computational complexity. We show that *k*NN-TSPI is robust and stable both concerning the parameter value k and the projection horizon trends;
- Empirical demonstration that our method is competitive against traditional predictors. In particular, we compare *k*NN-TSPI with six machine learning regression algorithms and five statistical models;
- Evaluation of different aspects that impact the predictive quality of similarity-based methods. Among the inspected factors are invariances to distortion in time series, distance measures, complexity-invariant distances, and ensemble functions. Understanding these particularities is the key to improving the *k*NN-TSPI's performance in particular scenarios.

We also highlight the importance of empirical assessments in a set of publicly available data. The 55 datasets considered in this work correspond to real-world problems archived at the ICMC-USP Time Series Prediction Repository [18].

The remainder of this article is structured as follows: Section II describes the related work. Section III provides the background and definitions of temporal data prediction by similarity. Section IV presents our *k*NN proposal to predict time series, while Section V specifies the experimental evaluation protocol. We show results and discussion in Section VI. Lastly, Section VII punctuates our research line's achievements and remaining challenges.

II. RELATED WORK

A few studies carried out in the last ten years showed that the similarity-based methods are accurate predictors for highly nonlinear and complex series [1], [6], [7], [19], [20]. Also, several research papers have noted the potential of this machine learning paradigm in other relevant data analysis tasks such as classification and clustering. For instance, the *k*NN algorithm has been successfully used for decades to perform time series classification [21]. Moreover, substantial

evidence exists that the outcomes obtained by the simple 1-NN with Dynamic Time Warping (DTW) are hard to overcome, even by more sophisticated inductors, such as random forests and Support Vector Machines (SVM) [22]. An empirical study in time series clustering indicates that the distance measure choice is more crucial than selecting the clustering method, with DTW obtaining outstanding performance [23]. A survey on time series anomaly detection showed that algorithms based on similarity search produce the highest general results [24]. We suppose that the performance merit of similarity-based methods is mostly due to the continuous research on distance invariances such as warping, rotation, and occlusion [25], [26].

Despite the increase in the number of methodologies proposed for applying *k*NN to time series prediction, questions remain as to how to determine the algorithm's parameters, what distance measure to choose, and how to combine the k nearest neighbors for a final decision [19]. In this direction, many studies have been carried out to get more insights into the performance of *k*NN for temporal data forecasting.

Sovilj *et al.* [27] introduced a framework that employs input processing before building the predictive model. One and multi-step-ahead predictions were made using optimally pruned extreme learning machine and Optimally Pruned *k*NN (OP-*k*NN). The results indicated that only OP-*k*NN benefited from the projection adopting the genetic procedure and the delta test.

Huang and Shyu [28] designed a *k*NN based on Least Squares Support Vector Machines (LS-SVM) to perform long-term time series prediction. The proposal outperformed the traditional LS-SVM and multi-input multi-output local learning approaches. To reduce the training complexity of an LS-SVM, Shi and Zhou [29] developed a *k*NN scheme for multi-step-ahead time series forecasting. Experiments on chaotic datasets demonstrated that the proposed approach overcomes single-step methods.

The study conducted by Bedo *et al.* [30] presented a new financial prediction approach based on the similarity between time series subsequences employing a database-driven architecture. This approach outperformed boosted *k*NN, ARIMA, and Multilayer Perceptron (MLP). Differently, Wei *et al.* [31] designed a *k*NN based neuro-fuzzy predictor that provided more accurate results than original neuro-fuzzy, ARIMA, and Artificial Neural Networks (ANN).

De La Vega *et al.* [32] developed a framework for temporal data forecasting that integrates *k*NN and differential evolution. Empirical results showed that *k*NN appeared to be more effective than ARIMA, mainly when the projection horizon ranged from short to moderate. In contrast, Flores *et al.* [33] introduced the fuzzy version of the *k*NN predictor. The experimental protocol tested the method on chaotic time series and, in agreement with the authors, its results were satisfactory compared with the crisp version of *k*NN and ARIMA.

While Marcacini *et al.* [6] proposed an approach incorporating Websensors – models that represent knowledge extracted from text news of a particular domain as well as the temporal evolution of this knowledge – into the DTW distance to improve *k*NN forecasts, Talavera-Llames *et al.* [34] presented an algorithm based on weighted nearest neighbors for big data time series. Yu *et al.* [20] created a similarity-based model for predicting short-term traffic conditions that proved competitive with SVM and ANN.

Wang and Koprinska [35] faced a new Distance Weighted *k*NN method with seven predictors, including a standard *k*NN, SVM, and ARIMA. The two lazy algorithms achieved the highest performances. Chen and Kartini [36] found similar results in a study covering a novel methodology for the very short-term forecast of hourly global solar irradiance. This proposal combines *k*NN with an ANN model. Tang *et al.* [37] proposed a complex weighted *k*NN that integrates empirical mode decomposition and principal component analysis for financial time series forecasting.

Although similarity-based time series prediction has been investigated in the immediate past, we are sure that previous studies have not identified the right combination of invariances needed for this task. Section IV addresses this issue by proposing a method that integrates three techniques to become invariant to amplitude, offset, and complexity and invulnerable to trivial matches. Our solution efficiently handles eventual errors introduced by temporal distortions in the search process for the k most perceptually similar subsequences to a reference query. Correctly identifying the k nearest subsequences is critical because their subsequent values are used to estimate future observations.

III. BACKGROUND AND DEFINITIONS

This section presents the central concepts and definitions of time series prediction by similarity, with a brief review of distance measures and ensemble functions. Our work also evaluates the impact of these distance measures and ensemble functions in the similarity search process adopted by local *k*NN predictors.

A. PREDICTION BY SIMILARITY

Time series prediction via similarity search, as well as any other machine learning technique for this same task, can be conducted according to two main approaches [5]: (i) global and (ii) local.

The global approach uses the whole training data series to construct a prediction model. Generally, a transposition is made to transform the sequential data into an attribute-value table that will feed some machine learning regressor. We portray the global approach in Fig. 1.

In the example of Fig. 1, we consider a time series Z with a number of observations $m = 15$. To transpose the sequence into the attribute-value table, we iteratively shift a sliding window of length $l = 5$ along the observations storing all the consecutive subsequences. Each subsequence obtained from the sliding window refers to a pair (X_i, y_i) where: $X_i = (x_{i1}, \ldots, x_{il})$ and corresponds to the temporal pattern of length l; and y_i represents the next value to X_i,

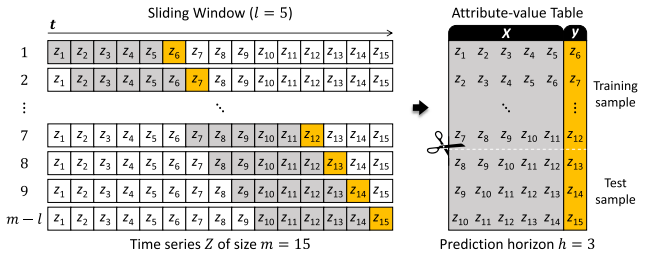
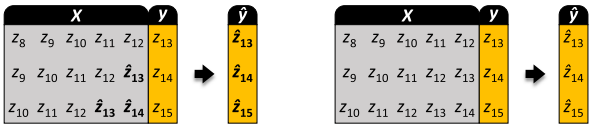


FIGURE 1. **In the global approach, the data sequence is converted to an attribute-value table.**

observed at time $l + 1$. Thus, the attribute-value table is constituted by the set of pairs (X_{ij}, y_{ij}), where $j \in [1, m - l]$.

We then define a prediction horizon $h = 3$ and, based on it, we partition the table into a training set to create the model and a test set to assess the built model. At this moment, we must choose a strategy to predict the series values some periods ahead ($h > 1$). The simplest strategy is the classical multi-step – also diffused as recursive in econometrics –, where the prediction of $h > 1$ is carried out h successive times adopting a prediction model with $h = 1$ [38]. Once the model is extrapolated, we can use the projected value or the respective actual value to estimate the subsequent prediction. We named the strategy as multi-step-ahead with approximate iteration when a model is fed back with predicted values. Conversely, when a model is fed back with actual values, it is called multi-step-ahead with updated iteration.

To assess the accuracy achieved by a given model, we need to compare the predicted values $\hat{y}_k$ with their actual values y_k, in which $k \in [1, h]$. An illustration of the multi-step-ahead prediction strategy, with approximate and updated iterations, is shown in Fig. 2.



(a) Approximate iteration (b) Updated iteration

FIGURE 2. **Multi-step-ahead prediction strategy with approximate and updated iterations.**

An advantage of the global approach is its low complexity of implementation. However, it is also susceptible to some drawbacks. For instance, the strategy considers that the pairs (X_{ij}, y_{ij}) are independent and identically distributed, which leads to a loss of temporal information on the part of the predictive model. Algorithms that typically employ the global approach are those based on ANN [39]–[41] and SVM [2], [42].

On the other hand, the local approach consists of machine learning algorithms adapted to include temporal information during the forecasting process. In short, the methods split the sequential data into subsequences whose the closest or the most important values linked to the current value are fused to project the future observation. We can use approximation functions such as simple and weighted average to combine such values. Some variations of the *k*NN predictor follow this approach [19], [20].

To better understand the differences between global and local approaches for time series prediction, we describe the global *k*NN regressor and the local *k*NN predictor in Sections III-A1 and III-A2, respectively.

1) THE *k*-NEAREST NEIGHBORS ALGORITHM

*k*NN is a non-parametric method used for classification and regression. It is a lazy learner because, instead of learning a discriminative function from the training data, it stores all the examples to generalize when performing a query. Specifically, given some features of a new example to be classified or regressed, *k*NN retrieves the k training examples closest to the new example according to some similarity measure. For classification, the final output is set by the majority class of the k nearest examples; for regression, the predicted value is given by the average of these examples [8]. Fig. 3 exemplifies the execution of *k*NN with $k = 1$, 3, and 5 for the classification of a new example E_N from a training set composed of 12 examples, seven positive (+) and five negatives (−), described by the features A_1 and A_2.

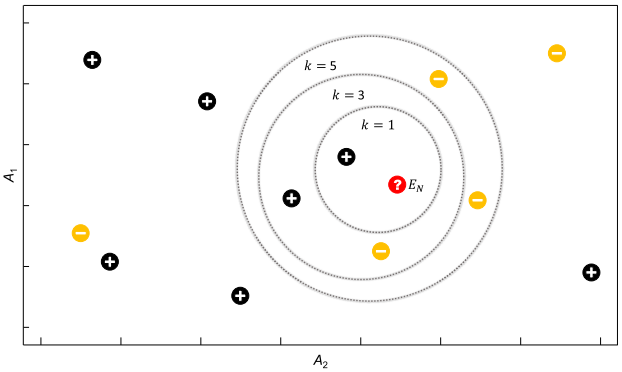


FIGURE 3. **Example of *k*NN with three different values for the parameter *k*.**

As we can see in Fig. 3, the number of nearest neighbors directly influences the algorithm's result. For example, for $k = 1$ and $k = 3$, example E_N would be classified as "positive", whereas, for $k = 5$, E_N would be "negative". The distance measure, in turn, must be selected based on the explanatory variables of the problem [43]. *k*NN demands low computational effort during the training phase, but the cost to classify new examples can be high since the method requires a full scan across all training set examples.

The *k*NN algorithm to predict temporal data is applied, following the global approach, in agreement with Fig. 1, where the explanatory variables are lagged values of the explained variable. The *k*NN regressor can be easily modified to deal locally with time series [16]. This adaptation, referenced in this paper as *k*NN-TSP, explicitly includes the temporal information into the prediction process.

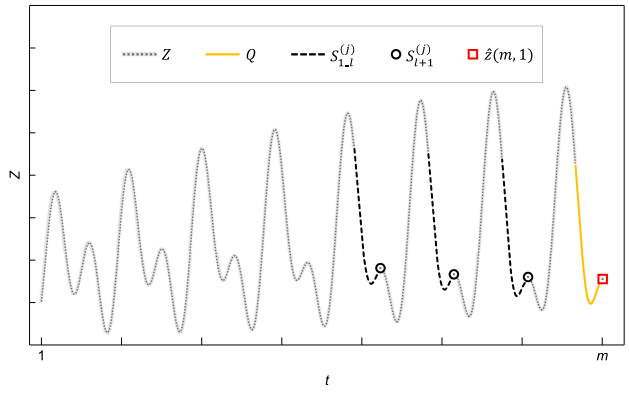


FIGURE 4. Example of *k*NN-TSP considering three nearest neighbors ($k = 3$) and a sliding window with 30 observations ($l = 30$).

2) THE *k*-NEAREST NEIGHBORS - TIME SERIES PREDICTION ALGORITHM

The general idea of *k*NN-TSP is intuitive and straightforward. Given a time series $Z = (z_1, z_2, \ldots, z_m)$ where $z_t \in \Re$, the aim is to project the value z_{m+h}, where h is the projection horizon. For simplicity but without any loss of generality, we will adopt a unitary horizon ($h = 1$) to represent the prediction of the next unseen observation z_{m+1} – also denoted as $\hat{z}(m, 1)$ – in the time series.

*k*NN-TSP uses the last l observations of the series as query Q and searches for the k most similar subsequences to Q, employing a sliding window of length l. Let $S^{(1)}_{1..l}, \ldots, S^{(k)}_{1..l}$ be the k most similar subsequences; the method captures the next values of each subsequence $S^{(j)}_{l+1}$ with $1 \leq j \leq k$ to estimate $\hat{z}(m, 1)$. To do so, the targets $S^{(j)}_{l+1}$ are passed to an ensemble function f, such as the Mean of Absolute Values (MAV) (1), which intends to approximate the value of $\hat{z}(m, 1)$.

$$f_{\text{MAV}}(R) = \frac{1}{k} \sum_{j=1}^{k} R^{(j)} \tag{1}$$

In (1), f refers to the ensemble function used to combine the k observations stored in R. This equation represents the most simple way to fuse values since it considers that all patterns within the time series are equally likely to be observed in the future.

Fig. 4 comprises an example of how *k*NN-TSP works, taking into account three neighbors ($k = 3$) and a sliding window with 30 observations ($l = 30$). We organize the pieces of information depicted in this figure as follows: the gray dotted line represents the entire time series; the yellow line symbolizes the query window; the black dashed line indicates the most similar subsequences found by the similarity search using the Euclidean distance; the black circles express the values employed for making the projection; and the red square denotes the observation we are willing to predict.

*k*NN-TSP defines an observation from the last l historical values. The dependence is restricted to a limited number of past observations since values that happened a long time ago usually do not influence the current value.

Algorithm 1: *k*NN-TSP

```
// Z represents a time series with m
   observations
// l refers to the search window length
// k indicates the number of nearest neighbors
// d expresses the distance measure
// f corresponds to the ensemble function
Input: Z, l, k, d, f
Output: ẑ(m, 1)
1 begin
      /* S^(i)_1..l contains the subsequence of length l
         which starts at observation i of the time
         series Z                                    */
2     S ← generate_subsequences(Z_1..(m−l), l);
      // Obtaining the query Q
3     Q ← Z_(m−l+1)..m;
      // D^(i) stores the distance between Q and S^(i)
4     D ← d(Q, S);
      // Choosing the k most similar subsequences
5     P ← search_nearest_neighbors(S, D, k);
      // Getting the next value P^(j)_l+1 of each
         nearest subsequence P^(j)
6     R ← {P^(1)_l+1, ..., P^(k)_l+1};
      // Calculation of a prediction
7     ẑ(m, 1) ← f(R);
8     return ẑ(m, 1);
9 end
```

Algorithm 1 presents the pseudocode of *k*NN-TSP. In the 2nd line, the variable S stores all subsequences of length l extracted from the time series Z. In the 3rd line, the method adopts the last subsequence of length l as a reference query (Q). The 4th line calculates the similarities (distances $\in D$) between the query Q and all subsequences previously assigned to S. From the distance values, in the 5th line, the algorithm searches for the k most similar subsequences to the query Q. In the 6th line, a variable R is used to store the subsequent values of each of the k most similar subsequences contained in P. Finally, in the 7th line, the ensemble function f (1) combines the values held in R to obtain an estimate of the future value.

The parameters k and l of Algorithm 1 are intuitive. We can easily estimate them by employing a training-testing validation procedure [44]. Furthermore, the value of l could be proportional to the number of values that form a seasonal station in the time series since the nearest neighbors would be more meaningful in predicting. The distance measure d and the ensemble function f are fixed statically before k and l. Choosing an appropriate distance measure and selecting an ensemble function are vital decisions to build an accurate predictor, and any similarity-based approach requires both. For this reason, we present a more detailed discussion about them in Sections III-B and III-C, distance measures and ensemble functions, respectively.

B. DISTANCE MEASURES

Similarity is a concept extensively used in a wide range of applications. We can understand it as an estimate of the similitude degree between two objects given a distance function [45], [46].

In time series prediction via machine learning models, quantifying how similar are two data subsequences to decide whether they belong to the same feature space is a highly subjective task that suffers influence by different factors. According to Hetland [47], the application domain and the method chosen to calculate the similarity are among the possible factors. The study formalized in Batista *et al.* [26] goes further, *i.e.*, it demonstrates that the performance of a similarity measure is related to the ability of this metric to capture the invariance required by the application domain correctly.

In what follows, we describe some of the most popular distance measures for time series. For the sake of organization, we categorize these metrics into two groups: (i) non-elastic distances, which include measures that perform a linear alignment between sequences; and (ii) elastic distances, which consider metrics that carry out a nonlinear alignment between series to calculate their similarity. As far as we know, this set of distances has never been empirically analyzed from a time series prediction perspective.

1) NON-ELASTIC DISTANCES

The vast number of available distance estimates reflects part of the researchers' efforts to explore the concept of similarity between pairs of objects [21], [26], [48]. These measures may be readily adapted or applied directly to compare time series [14].

Since each distance estimate contains specific particularities, the choice of one over the other is often conducted empirically on a particular problem. Still, a common property among them is linear alignment (Fig. 5).

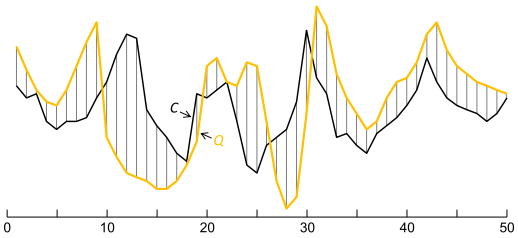


FIGURE 5. Example of alignment between two temporal data sequences using the Euclidean distance.

In Fig. 5, the similarity degree between the sequences $Q = (q_1, q_2, \ldots, q_l)$ and $C = (c_1, c_2, \ldots, c_l)$, both of length l, results from the squared differences between pairs of observations aligned on the time axis. We obtained this alignment using the Euclidean distance, derived from (2) [49].

$$L_{\boldsymbol{p}}(Q, C) = \left(\sum_{i=1}^{l} |q_i - c_i|^{\boldsymbol{p}} \right)^{\frac{1}{\boldsymbol{p}}} \tag{2}$$

TABLE 1. Description of non-elastic distances for time series.

Category	Distance Measure	Equation
$L_{\boldsymbol{p}}$	L_1 or Manhattan	$d_1(Q,C) = \sum_{i=1}^{l} \lvert q_i - c_i \rvert$
	L_2 or Euclidean	$d_2(Q,C) = \sqrt{\sum_{i=1}^{l} (q_i - c_i)^2}$
	L_3 or L_3 Metric	$d_3(Q,C) = \sqrt[3]{\sum_{i=1}^{l} \lvert q_i - c_i \rvert^3}$
	L_∞ or Chebychev	$d_4(Q,C) = \max_i \lvert q_i - c_i \rvert$
$\hat{L}_1$	Canberra	$d_5(Q,C) = \sum_{i=1}^{l} \frac{\lvert q_i - c_i \rvert}{\lvert q_i \rvert + \lvert c_i \rvert}$
	Kulczynski	$d_6(Q,C) = \frac{\sum_{i=1}^{l} \lvert q_i - c_i \rvert}{\sum_{i=1}^{l} \min(q_i, c_i)}$
	Lorentzian	$d_7(Q,C) = \sum_{i=1}^{l} \log(1 + \lvert q_i - c_i \rvert)$
	Sørensen	$d_8(Q,C) = \frac{\sum_{i=1}^{l} \lvert q_i - c_i \rvert}{\sum_{i=1}^{l} (q_i + c_i)}$
	Soergel	$d_9(Q,C) = \frac{\sum_{i=1}^{l} \lvert q_i - c_i \rvert}{\sum_{i=1}^{l} \max(q_i, c_i)}$
$\hat{L}_2$	Clark	$d_{10}(Q,C) = \sqrt{\sum_{i=1}^{l} \left(\frac{\lvert q_i - c_i \rvert}{q_i + c_i} \right)^2}$
	Neyman	$d_{11}(Q,C) = \sum_{i=1}^{l} \frac{(q_i - c_i)^2}{q_i}$
	Pearson	$d_{12}(Q,C) = \sum_{i=1}^{l} \frac{(q_i - c_i)^2}{c_i}$
	Squared χ^2	$d_{13}(Q,C) = \sum_{i=1}^{l} \frac{(q_i - c_i)^2}{q_i + c_i}$
	Additive Symmetric χ^2	$d_{14}(Q,C) = \sum_{i=1}^{l} \frac{(q_i - c_i)^2 (q_i + c_i)}{q_i c_i}$
Inner Product	Correlation	$d_{15}(Q,C) = 1 - \frac{m \sum_{i=1}^{l} q_i c_i - \sum_{i=1}^{l} q_i \sum_{i=1}^{l} c_i}{\sqrt{m \sum_{i=1}^{l} q_i^2 - (\sum_{i=1}^{l} q_i)^2} \sqrt{m \sum_{i=1}^{l} c_i^2 - (\sum_{i=1}^{l} c_i)^2}}$
	Cosine	$d_{16}(Q,C) = 1 - \frac{\sum_{i=1}^{l} q_i c_i}{\sqrt{\sum_{i=1}^{l} q_i^2 \sum_{i=1}^{l} c_i^2}}$
	Geodesic	$d_{17}(Q,C) = \arccos\left(\frac{\sum_{i=1}^{l} q_i c_i}{\sqrt{\sum_{i=1}^{l} q_i^2 \sum_{i=1}^{l} c_i^2}} \right)$
	Jaccard	$d_{18}(Q,C) = \frac{\sum_{i=1}^{l} (q_i - c_i)^2}{\sum_{i=1}^{l} q_i^2 \sum_{i=1}^{l} c_i^2 \sum_{i=1}^{l} q_i c_i}$
Information	Jeffreys	$d_{19}(Q,C) = \sum_{i=1}^{l} (q_i - c_i) \log\left(\frac{q_i}{c_i} \right)$
	Topsøe	$d_{20}(Q,C) = \sum_{i=1}^{l} \left(q_i \log\left(\frac{2q_i}{q_i + c_i} \right) + c_i \log\left(\frac{2c_i}{q_i + c_i} \right) \right)$
Hybrid	*Average Distance* (L_1, L_∞)	$d_{21}(Q,C) = \frac{\sum_{i=1}^{l} \lvert q_i - c_i \rvert + \max_i \lvert q_i - c_i \rvert}{2}$

Equation (2), typically known as the $L_{\boldsymbol{p}}$ norm or Minkowski metric, works with two l-dimensional vectors and a constant $\boldsymbol{p} \geq 0$. Its computational cost depends on the values of l and $\boldsymbol{p}$, so that the larger these constants, the greater the number of operations to be computed. The value assumed by $\boldsymbol{p}$ gives rise to several distance measures with specific behaviors. For instance, $\boldsymbol{p} = 1$ expresses the Manhattan (or City Block) distance, while $\boldsymbol{p} = 2$ refers to the Euclidean distance.

Table 1 lists the most popular distance functions that follow linear alignment. These non-elastic distances are organized into six categories according to their syntactic and semantic similarities.

Due to the simplicity of interpretation and coding, the $L_{\boldsymbol{p}}$ category contemplates distances that often guide the similarity search in real applications. A drawback of these measures is that they are sensitive to outliers and small distortions in the time axis of the series [50]. The metrics labeled as $\hat{L}_1$ are weighted versions of the Manhattan distance. The $\hat{L}_2$ category, in turn, covers measures similar to the Euclidean distance. Metrics belonging to the inner product category explicitly use scalar multiplication

TABLE 2. Distance measures and their implementation issues.

Distances	Problem
Clark, Squared χ^2, and Additive Symmetric χ^2	$0 \div 0$
Kulczynski, Pearson, Neyman, and Jeffreys	$x \div 0$
Topsøe	$0 \log(0)$
Jeffreys	$x \log(0)$ $x \log\left(\frac{x}{0}\right)$

in their definitions. The information category involves distances from the communication theory area; they are based on the concept of information entropy, *i.e.*, on the uncertainty estimation of a probability distribution [44]. Average Distance is a hybrid distance that averages the Manhattan (L_1) and Chebychev (L_∞) measures.

We must point out that even when the sequences are adequately normalized, some distance functions may involve undefined terms or terms that lead to a final score that is not in $\Re$. Table 2 exhibits the metrics whose terms can lead to implementation issues. In this table, x is a non-zero value observed.

In our experiments, we get around the problems punctuated in Table 2 in the following way [48]: we consider the result of operations $0 \div 0$ and $0 \log(0)$ to be zero. In cases of division by zero and logarithm of zero, we replace 0 by ϵ, a value extremely small and positive.

All the distance estimates indicated in Table 1 have linear time complexity, *i.e.*, $O(m)$, where m is the size of the time series.

2) ELASTIC DISTANCES

Unlike the non-elastic distance measures discussed in Section III-B1, elastic functions allow the alignment between two time series distorted on the time axis. Some examples of elastic distances are DTW, Longest Common Subsequence Similarity [51], Edit Distance with Real Penalty [52], Spatial Assembling Distance [53], Time Warp Edit [54], and Move-Split-Merge [55]. DTW is probably the most popular and accurate distance function for temporal data in related tasks such as classification and clustering. Therefore, we only focus our review and experimental evaluation on this measure as elastic distance.

DTW allows two time series visually similar but distorted on the time axis to be aligned for later point-to-point comparison [56]. As we shall explain in what follows, the inputs of DTW are two vector series, while the output is the aggregated distance between them.

Let be two sequences $Q = (q_1, q_2, \ldots, q_n)$ and $C = (c_1, c_2, \ldots, c_m)$, of length n and m, respectively. To compare them using DTW, we need to generate a matrix of size $n \times m$ in which the element of index (i, j) contains the distance d, typically the Euclidean, between the observations q_i and c_j. In practical terms, each cell (i, j) of the matrix corresponds to the alignment between the observations represented therein, as shown in Fig. 6.

An adjustment path $\bar{R} = (r_1, \ldots, r_t)$, where $max(n, m) \leq t < m + n - 1$, is a set of contiguous elements of the matrix that defines a mapping between Q and C. There are many possible adjustment paths, but the path to be chosen is the one that minimizes the deformation cost, *i.e.*, whose cumulative distance along the path is minimal. Equation (3) sets the optimal path, where r_i indicates the i^{th} element of the adjustment path.

$$DTW(Q, C) = \min_{\bar{R}} \left(\sum_{i=1}^{t} r_i \right) \tag{3}$$

The mentioned path is subject to three other constraints:

- Boundary constraint: The adjustment path need to begin and end at diagonally opposite corner cells of the matrix, *i.e.*, $r_1 = (1, 1)$ and $rt = (n, m)$;
- Continuity constraint: Matches must be performed in one-unit steps. Therefore, a match never jumps one or more observations;
- Monotonicity constraint: The relative order of the observations needs to be preserved so that the data sequence will not be able to "go back" in the cumulative cost matrix path. In other words, this restriction forces the points in the matrix to be monotonically spaced in time.

We can measure the cumulative distances applying a dynamic programming algorithm, which implements the following recurrence relation:

$$DTW(i, j) = d(q_i, c_j) + \min \begin{cases} DTW(i-1, j) \\ DTW(i, j-1) \\ DTW(i-1, j-1) \end{cases} \tag{4}$$

In (4), we sum the result accumulated in the current cell with the shortest distance from its three adjacent ones: the cell to the left, upper, or upper right diagonal. The number of iterations required to perform these calculations gives DTW an $O(n \times m)$ complexity. However, it is possible to reduce this computational time cost by using a "warping" window. Such a window limits how much the adjustment path can move away from the leading diagonal of the matrix [57], [58].

We considered in our experimental assessment DTW with the Sakoe-Chiba band [57]. The warping window length has been set to correspond to 10% of the length of the subsequences.

C. ENSEMBLE FUNCTIONS

As we explained in Algorithm 1, after kNN-TSP returns the set P with the k most similar subsequences $P^{(1)}_{1..l}, \ldots, P^{(k)}_{1..l}$ of length l from a query Q, the local similarity-based method uses the next observation of each subsequence $P^{(j)}_{l+1}$, where $1 \leq j \leq k$, to predict the value z_{m+1} of a series Z with size m. Thus, we need an ensemble function f that combines the subsequent values of the different nearest subsequences to output a new observation. Although the mean of these

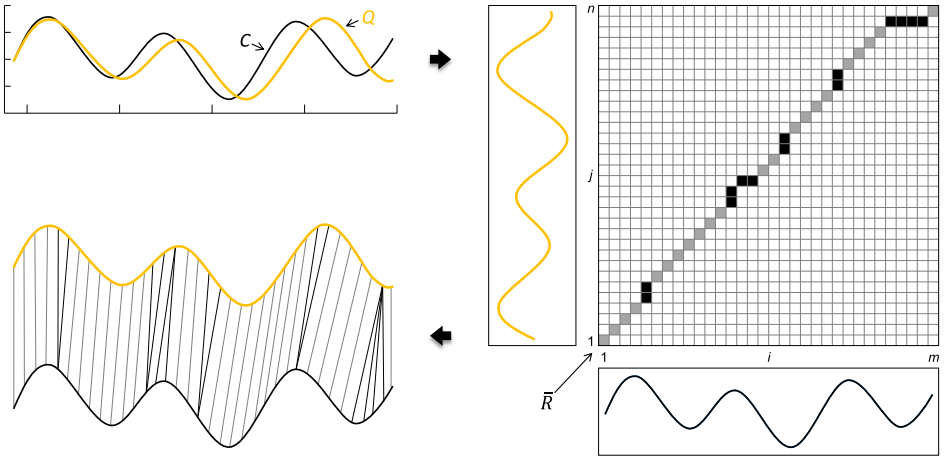


FIGURE 6. Schematization of a cumulative cost matrix resulting from the DTW application with the adjustment path highlighted (modified from [56]).

values is a good data fusion measure, we may choose to employ other ensemble functions, such as median, Distance Weighted (DW), and Mean of Relative Values (MRV).

The ensemble function DW, defined by (5) [44], weighs the subsequent observations $P_{l+1}^{(j)}$ by the distances between the reference query Q and the data subsequences $P_{1..l}^{(j)}$.

$$f_{DW}(P) = \frac{\sum_{j=1}^{k} w_j P_{l+1}^{(j)}}{\sum_{j=1}^{k} w_j} \tag{5}$$

Different mathematical relationships can determine the weights w_j of (5), but the following expression gives a usual way:

$$w_j = \frac{1}{d\left(Q, P_{1..l}^{(j)}\right)^2}$$

MRV (6) [16], in turn, computes the future value based on the sum of the last value z_m sampled from the time series with the average of the differences between the last value $P_l^{(j)}$ of each most similar subsequence and the value $P_{l+1}^{(j)}$ that succeeds it.

$$f_{\text{MRV}}(P) = z_m + \frac{\sum_{j=1}^{k}\left(P_{l+1}^{(j)} - P_l^{(j)}\right)}{k} \tag{6}$$

MRV is more pertinent when there is variance in amplitude and offset since it projects values taking patterns at different trend levels into account.

IV. PROPOSED METHOD: *k*-NEAREST NEIGHBORS - TIME SERIES PREDICTION WITH INVARIANCES

We have identified that the naive procedure of similarity search adopted by the original *k*NN-TSP has two main limitations that still need to be overcome: (i) occurrence of incorrect matches due to the absence of mechanisms that deal with invariances to amplitude, offset, and complexity; and (ii) undesirable results because of the occurrence of

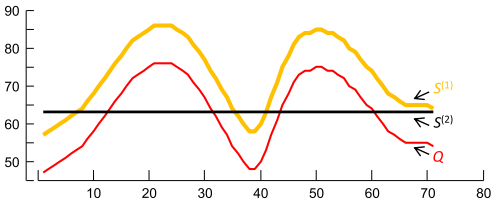


FIGURE 7. Example of variance to amplitude and offset. Popular distances measures such as Euclidean and DTW indicate that Q is more similar to $S^{(2)}$ than $S^{(1)}$.

trivial matches. We discuss such issues and their impact individually in what follows.

To better illustrate how the amplitude and offset variances can be a critical problem for time series prediction, we show an example in Fig. 7 with a query Q and two subsequences, $S^{(1)}$ and $S^{(2)}$. In this example, we can conclude from a visual analysis that $S^{(1)}$ is very similar to Q but with a small offset increase of ten units. On the other hand, $S^{(2)}$ is a simple straight line that differs from the behavior of Q. However, the Euclidean distance (d) outputs not intuitive results indicating that the most similar subsequence to Q is $S^{(2)} - d(Q, S^{(2)}) = 84.26$ –, while $d(Q, S^{(1)}) = 114.54$. The same wrong match happens when we employ the well-known DTW distance measure. Such an unexpected result occurs because the small offset differences are accumulated during the matching of the subsequences, leading to a final value higher than awaited.

We need to reinforce that amplitude and offset can be essential features to characterize subsequences in some cases. However, in most similarity-based search applications, incorrect matches are returned if the search does not address amplitude and offset invariances. This problem arises because the occurrence of different subsequences in the same offset is rare and even small differences in the offset are sufficient to increase the values of a distance metric and lead to incorrect matches, as represented in Fig. 7.

A simple way to deal with amplitude and offset is to perform the z-normalization of the series, making the sequence have a zero average and unit standard deviation. The z-normalization is given by (7), where z'_t and z_t are, in this order, the normalized value and the original observation of the time series Z at time t. The other terms correspond to the average (μ) and standard deviation (σ) of a sequence covering z_t.

$$z'_t = \frac{z_t - \mu}{\sigma} \tag{7}$$

The second issue approached in our proposal is the complexity invariance. The problem here is when a pair of subsequences with a similarly shaped but complex pattern is considered unmatched by a distance measure due to the data morphology.

Fig. 8 exemplifies the dilemma of variance to complexity. In this figure, we used z-normalized versions of the query in Fig. 7 and introduce some data corruptions in a small proportion, such as noise, phase differences, and amplitude distortion. Even so, Q and $S^{(1)}$ still resemble each other. Note that this setting is more realistic than the one portrayed in Fig. 7 since, in most application domains, we expect that one event of interest will generate similar subsequences but never exact copies. Once again, the Euclidean distance is counter-intuitive and indicates that Q is more similar to $S^{(2)}$ than $S^{(1)}$. This error is justified because time series with simple shapes like the straight line $S^{(2)}$ generally match well with complex shapes. Differently, sequences with complex behaviors are typically described by various characteristics, including peaks and valleys, that make them difficult to match, even when the patterns are similar to the human eye.

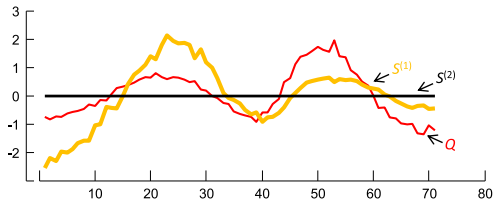


FIGURE 8. Example of variance to complexity. The Euclidean distance demonstrates that Q is more similar to $S^{(2)}$ than $S^{(1)}$ due to peaks and valleys in $S^{(1)}$.

An efficient solution to treat such variance is to employ the Complexity-Invariant Distance (CID) [26]. We present the CID measure and its complexity estimate in Section IV-A.

The last issue contemplated by our proposal is the disregard of trivial matches [59]. A trivial match is a subsequence drawn from a sliding window that begins at observation m and which is very similar to the subsequences starting at time $m + 1$ (or $m - 1$). These subsequences are trivial matches because they share all but two values. Fig. 9 reproduces this idea, where we display three subsequences that are essentially the same despite a shift in one unit to the left and right between them. In this scenario, the distances between the query Q and the subsequences $S^{(1)}$ and $S^{(2)}$ are so small that

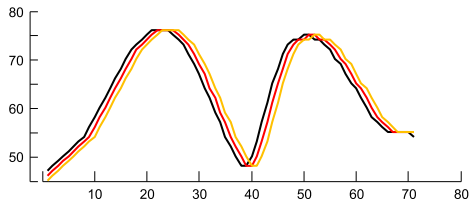


FIGURE 9. Examples of trivial matches. The distances between the query Q (red subsequence) and their respective left (black subsequence) and right (yellow subsequence) shifted versions by one observation are minimal, leading to undesired matches.

the search procedure will tend to choose $S^{(1)}$ and $S^{(2)}$ as two of the k most similar subsequences to Q.

Trivial matches are especially problematic when we want to use the k most similar subsequences to a given query, rather than the single most similar subsequence, aiming to include diversity and reduce errors in searching for temporal patterns. A similarity search with little diversity will almost always return exact copies of the query subsequence with some minor variations. An easy solution to obtain the desired diversity is to exclude trivial matches by iterative checking.

Given a subsequence $S^{(1)}_{i..(i+l-1)}$ and a subsequence $S^{(2)}_{j..(j+l-1)}$, such that $j > i$, it is said that $S^{(2)}$ is a trivial match of $S^{(1)}$ if $|j-i| \leq l$. Obviously, this condition implies identifying trivial matches for their subsequent exclusion.

To mitigate the problems faced by kNN-TSP, we propose in this paper a new extension that employs the three solutions discussed above to improve the search for similar subsequences. Algorithm 2 presents our method kNN-TSPI. The time complexity of this similarity-based predictor is $O(m \times l)$, where m is the size of the time series and l is the length of the search window.

In the 2$^{\text{nd}}$ line of Algorithm 2, we store all subsequences of length l extracted from the time series Z in the variable S. Then, in the 3$^{\text{rd}}$ line, all the collected subsequences are z-normalized, and their respective computed averages and standard deviations are reserved for later use. In the 4$^{\text{th}}$ line, we memorize the reference query Q. In the 5$^{\text{th}}$ line, we calculate the similarities (complexity-invariant distances $\in D$) between the normalized query Q and all z-normalized subsequences through the CID measure. To make the similarity search invariant to amplitude and offset, the z-normalization is performed both on the query Q and the sliding window, not on the whole time series. In the 6$^{\text{th}}$ line, we search for the k most similar subsequences. With the treatment of trivial matches, we guarantee that the k nearest subsequences returned do not overlap with the query Q or each other. We get the next value of each most similar subsequence in the 7$^{\text{th}}$ line and, in the 8$^{\text{th}}$ line, we normalized them employing the averages and standard deviations previously computed from their source subsequences. Such normalization has the purpose of making the value $P^{(j)}_{l+1}$ have the same distribution as the subsequence $P^{(j)}_{1..l}$. Afterward, in the 9$^{\text{th}}$ line, the subsequent z-normalized values are mapped to the query values space in

Algorithm 2: *k*NN-TSPI

```
// Z represents a time series with m observations
// l refers to the search window length
// k indicates the number of nearest neighbors
Input: Z, l, k
Output: ẑ(m, 1)
begin
    /* S(i)_{1..l} contains the subsequence of length l which starts at observation i of the time series Z */
    S ← generate_subsequences(Z_{1..(m−l)}, l);
    /* S(i)' is the z-normalization of the subsequence S(i). The average and standard deviation used for normalization are also allocated for reuse */
    [S', σ, μ] ← z_scores(S);
    // Obtaining the query Q
    Q ← Z_{(m−l+1)..m};
    /* D(i) stores the complexity-invariant distance between Q and S(i), both z-normalized */
    D ← CID(z_scores(Q), S');
    // Choosing the k most similar subsequences (non-trivial matches)
    P ← search_nearest_neighbors(S, D, k);
    // Getting the next value P(j)_{l+1} of each nearest subsequence P(j)
    R ← {P^(1)_{l+1}, ..., P^(k)_{l+1}};
    /* Normalization in z-scores of each value R(j) using the average and standard deviation of their respective subsequence P(j)_{1..l} */
    R' ← { (R^(1) − μ(P^(1)_{1..l})) / σ(P^(1)_{1..l}), ..., (R^(k) − μ(P^(k)_{1..l})) / σ(P^(k)_{1..l}) };
    // Mapping the z-normalizations to the values of the query Q (8)
    R ← map_query_values(Q, R');
    // Calculation of a prediction
    ẑ(m, 1) ← f(R);
    return ẑ(m, 1);
end
```

agreement with (8) and used by the ensemble function f (1) to estimate the future value.

$$R = \sigma(Q) \times R' + \mu(Q) \tag{8}$$

In (8), σ and μ correspond, in this order, to the standard deviation and the average of the reference query Q. This equation comprises the inverse function of the z-normalization and allows mapping, to the value space of the subsequence Q, the content of the variable R' that was previously z-normalized.

A. COMPLEXITY-INVARIANT DISTANCE MEASURES

Temporal data commonly presents inopportune distortions. These undesirable effects may cause distance measures to fail to adequately capture the similarity between time series. Thus, when sequential data exhibit distortions in the time axis, traditional metrics to measure the similitude between pairs of objects end up associating too large distances with similar objects [26].

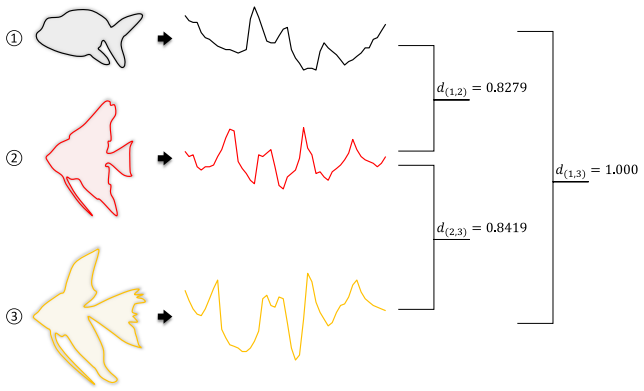


FIGURE 10. Example of the need for invariance to complexity. In terms of Euclidean distance, even objects 2 and 3 being similar to the human eye, object 2 is more alike to object 1.

Fig. 10 portrays the problem of variance to complexity using three time series computationally generated from the contour of fish images by the radial scanning technique [60]. By employing this procedure, the distances between the center point of the object of interest and the consecutive points of its contour are extracted to compose a sequence of distances. In Fig. 10, the initial reference point for calculating the distances was the fish mouth, and the clockwise direction was adopted. The Euclidean distance between the built series shows that, although the shapes of objects 2 and 3 are visually similar, object 2 is more alike to object 1 ($d_{(1,2)} = 0.8279$ and $d_{(2,3)} = 0.8419$). This fact introduces errors in the similarity search process since complex data sequences can be considered more similar to simpler ones.

The application of complexity-invariant distances, such as CID [26] and DTW - Delta (DTW-D) [61], allows morphological information of time series to be estimated and used as a correction factor for existing distance measures. As a result of this weighting, we have more precise and relevant pattern matching. Next, we introduce these measures in detail.

1) CID

CID considers the morphology of the sequences being compared and assigns greater distances to the sequences with different complexities. We may define CID from the Euclidean distance following (9) [26].

$$CID(Q, C) = ED(Q, C) \times CF(Q, C) \tag{9}$$

In (9), Q and C represent two time series, ED is the Euclidean distance, and CF comprises a complexity correction factor defined by (10). In such an equation, $CE(Q)$ and $CE(C)$ reflect complexity estimates of the sequences Q and C, respectively. If Q and C have the same complexity, CID degenerates to the Euclidean distance.

$$CF(Q, C) = \frac{\max(CE(Q), CE(C))}{\min(CE(Q), CE(C))} \tag{10}$$

The original CID adopts a reasonably simple complexity estimate [26]; it is based on the physical intuition that if we

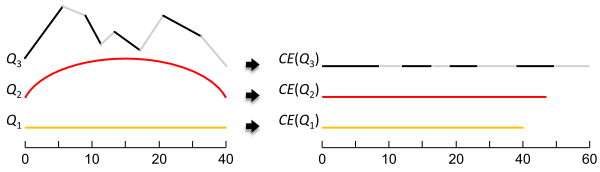


FIGURE 11. Complexity estimates of three time series with distinct behaviors. The less complex series (Q_1) "stretches" to the shortest line segment, while the most complex series (Q_3) "stretches" to the most extended line segment (modified from [26]).

could "stretch" a temporal data sequence until it becomes a straight line, a complex series would result in a longer line than a simple time series. Fig. 11 synthesizes this idea with an example.

The complexity estimate illustrated in Fig. 11 and used by CID can be determined according to (11).

$$CE(Q) = \sqrt{\sum_{i=1}^{n-1}(q_i - q_{i+1})^2} \tag{11}$$

Equation (11) should be applied for sequences with the same number of observations since only the differences between the observed values are taken into account, ignoring the differences in the time the observations occurred. Such a condition does not restrict the complexity usage when coupled with a distance metric like the Euclidean, which also requires that the time series have the same size. However, it limits the employment of other more flexible distances, such as DTW. Further, (11) requires that the sequences to be compared are previously normalized in amplitude and offset. Although this normalization is a standard requirement for most application domains, a complexity estimate that does not claim invariance to amplitude and offset may be quite useful in other fields of study.

We found dozens of complexity measures in the signal processing area that can also be applied to time series. Five of these complexity estimates are described below based on information theory, chaos theory, and Kolmogorov approximation concepts. Our experimental assessment compares these five complexity measures with the original CID.

- **Absolute Difference:** This measure estimates the complexity of a given data sequence in an analogous way to Squared Difference (11). However, we took the absolute differences instead of computing the squared differences between consecutive observations. The following equation formalizes this concept: $CE(Q) = \sum |q_i - q_{i+1}|$;
- **Compression:** Based on the Kolmogorov complexity, this measure can be approximated by compression algorithms, such as Lempel-Ziv-Welch. Initially, each time series is converted to symbols using the Symbolic Aggregate approXimation (SAX) [62]. The discretized sequences are then compressed using a file compression utility. The complexity estimate of a time series equals the size in bytes of the compressed file;
- **Edges:** This measure considers the number of edges in the signal, which we can interpret as the number of changes in the trend or how many times the first derivative of the data sequence changes its signal to positive or negative values;
- **Zero-crossings:** This measure expresses the complexity of a given time series from the zero-crossing rate, *i.e.*, from the number of times the signal changes its amplitude from a positive value to negative or vice versa. In speech analysis, this estimate is widely used to recognize voiced fragments apart from unvoiced segments and noisy breaks [63];
- **Permutation Entropy:** This measure reflects the entropy of a set of patterns produced from a permutation of natural numbers [64]. In the generation of these patterns, all possible permutations of numbers between 0 and $n-1$ are considered, with the value of n commonly chosen in a range from 3 to 7. For example, for $n = 3$, the following permutations are generated: {[0, 1, 2], [1, 0, 2], ..., [2, 1, 0]}. We can interpret the pattern [0, 1, 2] as a temporal sequence of three values in which the second observation is larger than the first one, and the third value is greater than the second one. Concerning the pattern [1, 0, 2], it is a temporal sequence of three observations where the second value is smaller than the first one, and the third observation is greater than the first one. We can analyze all the generated patterns following this same reasoning. To obtain the probability of each pattern, we run a sliding window with n observations across the temporal sequence, counting the number of times each pattern occurs. The complexity estimate is nothing less than the entropy of the set of patterns identified in the time series.

We can employ all the complexity estimates introduced above to compare sequences composed of a few observations. In similarity-based forecasting, this restriction is valid considering that the compared subsequences are usually short, with lengths proportional to the number of observations that characterizes a seasonal period within the time series.

2) DTW-D

Proposed initially to improve time series classification using semi-supervised learning, DTW-D can be considered a complexity-invariant distance by approximating globally similar series employing (12) [61].

$$DTWD(Q, C) = \frac{DTW(Q, C)}{ED(Q, C) + \epsilon} \tag{12}$$

Equation (12) divides the DTW result by the Euclidean distance plus an extremely small positive value ϵ, which avoids the divide-by-zero error.

V. EXPERIMENTAL SETTING

We organized an experimental protocol in three steps to evaluate and compare our method kNN-TSPI with

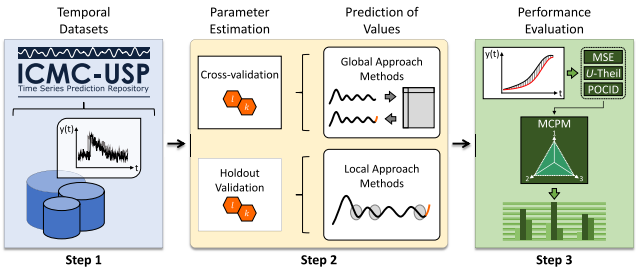


FIGURE 12. Overview of the experimental setting.

previous similarity-based predictors and state-of-the-art predictive models (Fig. 12).

In Step 1, we collected 55 real-world datasets currently managed by the ICMC-USP Time Series Prediction Repository [18]. These temporal data come from different domains such as finance, medicine, engineering, agriculture, and tourism. Table 3 shows, for each dataset, the acquisition period, the time series size (m), the maximum number of observations that constitute a seasonal period in the data sequence (max_p), and the projection horizon (h), which refers to the number of values we want to predict.

In Step 2, we estimated the best parameter values for each predictor. We considered ten-fold cross-validation for the methods that follow the global approach and holdout validation for the algorithms that run according to the local approach. We sought to minimize the Mean Square Error (MSE) in both cases.

The two parameters required by the similarity-based methods are the length of the reference query (l) and the cardinality of the set of most similar subsequences (k). For l, we tested values between 3 and max_p in increments of 2; for k, values between 1 and 9 also in increments of 2. Since max_p is an upper bound for the number of observations corresponding to a variation in the seasonal pattern, the value of l will be proportional to the seasonal cycle within the temporal sequence.

To show that kNN-TSPI can be as accurate as, or more accurate than, state-of-the-art predictors, we confronted the proposed algorithm with seven traditional methods [65]: six non-parametric and five parametric. Table 4 indicates the 11 predictors used for comparison, the input arguments required by them, and the value ranges adopted in the adjustment step. As summarized in this table, we applied the machine learning regression algorithms following the global approach and adjusted their parameters through ten-fold cross-validation with MSE minimization. The statistical models, in turn, had their parameters estimated using the Box-Jenkins method with minimizing the Akaike Information Criterion (AIC) and maximum likelihood. Exceptionally, we determined the parameters of LSTM, MA, HWA, and HWM by employing holdout validation with minimizing the MSE.

After the parameter estimation step, we built and adjusted predictive models on the training data. Then, we extrapolated

TABLE 3. Dataset description and evaluation settings.

ID	Dataset	Acquisition	Begin	End	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	1,499	7	88
06.D	CEMIG	Daily	Jan-03-1995	Dec-27-2000	1,499	7	88
07.D	*IBV*	Daily	Jan-03-1995	Dec-27-2000	1,499	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	1,499	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	3,128		
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
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	1,400	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	1,800	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	1,896	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	1,800		
53.I	• A					60	120
54.I	• B					60	120
55.I	Laser	1.0s Intervals	1991	1991	1,000	8	100

the fitted models to predict h periods ahead, taking into account the two projection strategies described in Section III and referenced in this study as:

- Multi-step-ahead projection strategy with approximate iteration;
- Multi-step-ahead projection strategy with updated iteration.

Finally, in Step 3, the results achieved by the predictors were assessed. We adopted three performance measures to compare the projected data with their respective actual values: MSE, Theil's U (TU), and Prediction Of Change In Direction (POCID).

The MSE measure is popularly known for confronting the actual values (z_t) with the predicted values ($\hat{z}_t$), as formalized by (13). It expresses the ratio between the quadratic sum of the forecast error and the number of projected observations (h). MSE values close to zero suggests efficient predictors.

$$MSE = \frac{1}{h}\sum_{t=1}^{h}(z_t - \hat{z}_t)^2 \tag{13}$$

TABLE 4. Algorithms and search spaces considered in the parameter estimation step. The acronyms not yet defined are: Gaussian Process (GP), Reduces Error Pruning Tree (REPTree), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Moving Average (MA), Holt-Winters Additive model (HWA), and Holt-Winters Multiplicative model (HWM).

Learning Algorithm	Parameter Description	Search Space (begin : step : end)
GP	Search window length (l)	$l = 3:2:max\_p$
	Gaussian noise level (N)	$N = 0.25:0.25:1$
	Gaussian's width of the radial basis kernel function (σ)	$\sigma = 0.005:0.05:0.25$
REPTree	Search window length (l)	$l = 3:2:max\_p$
	Minimum proportion of the variance (ϑ)	$\vartheta = 0.001$
MLP	Search window length (l)	$l = 3:2:max\_p$
	Number of units in the hidden layer (n)	$n = 3:2:max\_p$
	Learning rate (Υ)	$\Upsilon = 0.03$
	Momentum (M)	$M = 0.2$
	Number of epochs (E)	$E = 500$
RNN LSTM	Search window length (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 (Υ)	$\Upsilon = 0.03$
	Momentum (M)	$M = 0.2$
	Number of epochs (E)	$E = 500$
SVM	Search window length (l)	$l = 3:2:max\_p$
	Regularization parameter ($\mathcal{C}$)	$\mathcal{C} = 0:0.25:1$
	Gaussian's width of the radial basis kernel function (σ)	$\sigma = 0.005:0.05:0.25$
Statistical Algorithm	**Parameter Description**	**Search Space (begin : step : end)**
MA	Number of values taken by the average (r)	$r = 3:2:max\_p$
HWA HWM	Level smoothing constant (α)	$\alpha = 0:0.25:1$
	Trend smoothing constant (β)	$\beta = 0:0.25:1$
	Seasonality smoothing constant (γ)	$\gamma = 0:0.25:1$
	Number of values that constitute a seasonal period (s)	$s = 3:2:max\_p$
ARIMA	Autoregression order (p)	$p = 0:1:\log(m-h)^{1/2}$
	Differentiation degree (d)	$d = 0:1:2$
	Moving average order (q)	$q = 0:1:\log(m-h)^{1/2}$
SARIMA	Autoregression order (p)	$p = 0:1:\log(m-h)^{1/2}$
	Differentiation degree (d)	$d = 0:1:2$
	Moving average order (q)	$q = 0:1:\log(m-h)^{1/2}$
	Seasonal autoregression order (P)	$P = 0:1:\log(m-h)^{1/2}$
	Seasonal differentiation degree (D)	$D = 0:1:2$
	Seasonal moving average order (Q)	$Q = 0:1:\log(m-h)^{1/2}$
	Number of values that constitute a seasonal period (s)	$s = max\_p$

The TU coefficient considers the MSE error normalized by the error resulting from a trivial model. This trivial model supposes that the best value for $t+1$ is the value observed at time t. Equation (14) defines the TU index.

$$TU = \frac{\sum_{t=1}^{h}(z_t - \hat{z}_t)^2}{\sum_{t=1}^{h}(z_t - z_{t-1})^2} \tag{14}$$

We can interpret the values of the TU coefficient as follows: TU $>$ 1, the predictor did not outperform the trivial model; TU $=$ 1, the predictor's performance is equivalent to the trivial model; TU $<$ 1, the predictor outperformed the trivial model; TU ≤ 0.55, the predictor is reliable for making future predictions.

Another performance measure considered in our experiments is the POCID hit rate, established by (15). In this equation, $\boldsymbol{D}_t$ receives the value 1 if $(\hat{z}_t - \hat{z}_{t-1})(z_t - z_{t-1}) > 0$, and is 0 otherwise. The concept behind POCID is to evaluate the accuracy of the data direction changes, *i.e.*, if the future value of the time series will increase or decrease given the current observation.

$$POCID = \frac{\sum_{t=1}^{h}\boldsymbol{D}_t}{h} \times 100 \tag{15}$$

Because of the different assessment indexes and to guide the choice of suitable predictors, we decided to perform a multi-criteria analysis. This paper uses the Multi-Criteria Performance Measure (MCPM) proposed in [66] to combine the results of MSE, TU, and POCID. Unlike the error indexes, POCID values must be maximized rather than minimized. Hence, we adopted the POCID counterpart or error rate (ε), defined as $\varepsilon = 100 - POCID$.

The MCPM calculates, for each algorithm-dataset pair, the total area of a three-sided polygon whose vertices are equivalent to the values of the individual performance measures obtained by the predictor using the temporal data. Algorithms with low MCPM are considered good predictors.

To detect statistical differences between a set of prediction algorithms, we employed the non-parametric statistical test of Friedman with a confidence interval of 95% (p-value < 0.05). Whenever the significance test identified differences, we applied the Nemenyi posthoc test to determine the best-performing predictors.

We also explored distinct aspects intrinsic to the time series prediction task based on similarity search. To do so, we ran *k*NN-TSPI in the same way as the one already presented in this section, but we configured it with several distance metrics (Section III-B), complexity-invariant distance measures (Section IV-A), and ensemble functions (Section III-C).

All programs involving our experimental evaluation were implemented using the following technologies: MATLAB and GNU Octave; R with the Forecast package; and Java with the Weka library.

VI. RESULTS AND DISCUSSION

We organized the discussion of our experimental results into three primary studies: (i) *k*NN-TSPI versus two previous similarity-based predictors; (ii) *k*NN-TSPI versus 11 state-of-the-art prediction methods; and (iii) performance analysis of *k*NN-TSPI with different distance measures and ensemble functions.

The statistical validation findings were summarized in Critical Difference (CD) diagrams whose scale indicates the average performance ranking of each prediction algorithm [67]. In these diagrams, predictors joined by a thick line showed no Statistically Significant Differences (SSD).

We restrict ourselves to displaying here only the diagrams obtained from the MCPM. Our supplementary material provides the other diagrams and the detailed results achieved by the three individual performance measures and the values found in the parameter estimation step.

A. *k*NN-TSPI VERSUS OTHER SIMILARITY-BASED PREDICTORS

The global *k*NN regressor (Section III-A1) assumes that the input data are independent and identically distributed. The local methods *k*NN-TSP (Section III-A2) and *k*NN-TSPI (Section IV) were formulated to consider the temporal characteristics of the data in their modeling. To show the cost-benefit of these algorithms, we faced them regarding predictive performance. Such comparisons totaled 330 configurations (3 predictors $\times$ 2 projection strategies $\times$ 55 datasets).

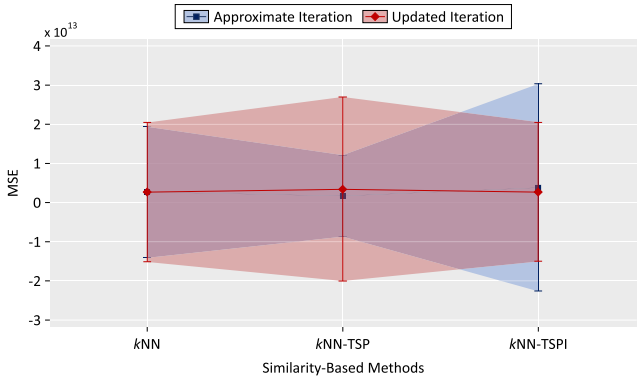
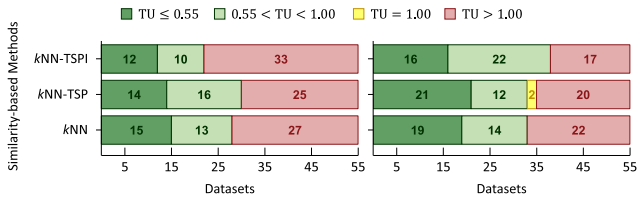


FIGURE 13. Averages and SD of MSE obtained by the similarity-based predictors.

We display in Fig. 13 the averages and Standard Deviations (SD) of MSE obtained by the similarity-based methods over the 55 datasets. In this figure, *k*NN-TSP with approximate iteration exhibited the best results. Although *k*NN and *k*NN-TSPI showed some atypical values of SD, they had similar MSE averages. In the updated iteration scenario, *k*NN-TSPI achieved better MSE values than *k*NN-TSP, and its performance was very close to *k*NN.

Given the four possible ranges of TU values, Fig. 14 presents the results reached by each predictor considering all the time series. We can state from this figure that *k*NN-TSPI with approximate iteration should provide reasonably lower performance than its version with updated iteration. Also, *k*NN and *k*NN-TSP may display similar results or even overcome the predictive error of *k*NN-TSPI. We should not view this fact as negative since, in *k*NN-TSPI, the similarity search is conducted with diversity to avoid erroneous choices of the most similar subsequences. We will clarify this point further when Table 5 and Figs. 16-17 are introduced.



(a) Approximate iteration (b) Updated iteration

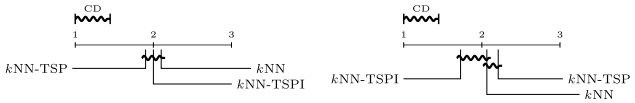
FIGURE 14. Performance, in terms of TU, of the similarity-based predictors across the 55 datasets.

Table 5 exhibits the averages and SD of the values obtained with the POCID index application. According to the information summarized in this table, *k*NN-TSPI and *k*NN provided, in this order, the best and worst hit rates relative to the projected horizon trends regardless of the prediction strategy employed.

We sketch in Fig. 15 the CD diagrams for the MCPM total area values resulting from the execution of the similarity-based predictors. As for the recursive strategy by approximate

TABLE 5. Average and SD values of POCID achieved by the similarity-based predictors.

Predictors	Approximate Iteration	Updated Iteration
*k*NN	53.83 (24.79)	53.88 (22.88)
*k*NN-TSP	59.31 (20.96)	57.44 (18.63)
*k*NN-TSPI	**61.86 (18.83)**	**59.11 (18.34)**



(a) Approximate iteration – MCPM (b) Updated iteration – MCPM

FIGURE 15. MCPM derived from the similarity-based predictors.

iteration, *k*NN-TSP provided the best result (Fig. 15(a)). However, our method obtained the smallest prediction errors using updated iteration (Fig. 15(b)). Moreover, for both prediction strategies, *k*NN-TSPI was the most stable algorithm regarding the accuracy of the projected horizon trends.

To reinforce the results of Table 5 and compare our method's prediction quality concerning the previous similarity-based predictors, Fig. 16 displays the behavior of the algorithms on four challenging datasets. The four illustrated examples show that *k*NN-TSPI stands out for its solid performance in determining the trend direction of the projection horizons.

As mentioned, *k*NN-TSPI has only two readily identifiable parameters: l and k. The parameter l defines the length of the search window, *i.e.*, the number of previous observations on which the current value is dependent. The value of l is proportional to a seasonal station in the time series. For example, for series with daily samples and weekly seasonality, we could establish a search window $l = 7$, so *k*NN-TSPI would employ the seasonality information to obtain the most similar subsequences. Particularly in time series with seasonality, we must adopt a search window length greater than or equal to the size of the seasonal cycle to allow *k*NN-TSPI to use the seasonality information.

Another factor to consider when choosing the length of the search window is the temporal dependency. For example, given a series representing the daily number of patients visiting a specific hospital ward, it would not be meaningful to employ a search window $l = 30$ since there is not much relationship between the current observation and a value observed 30 days ago. Therefore, defining a search window length that describes the actual dependency of the current values with historical observations is key to ensuring more accurate predictions.

The parameter k, in turn, defines the number of nearest neighbors. The value of k depends on the number of observations recorded in the time series. Considering the parameter l, the space of historical values covered by k nearest neighbors is $l \times k$. In this sense, the condition $l \times k \ll m$ must be satisfied to ensure diversity and have good candidates for nearest neighbors.

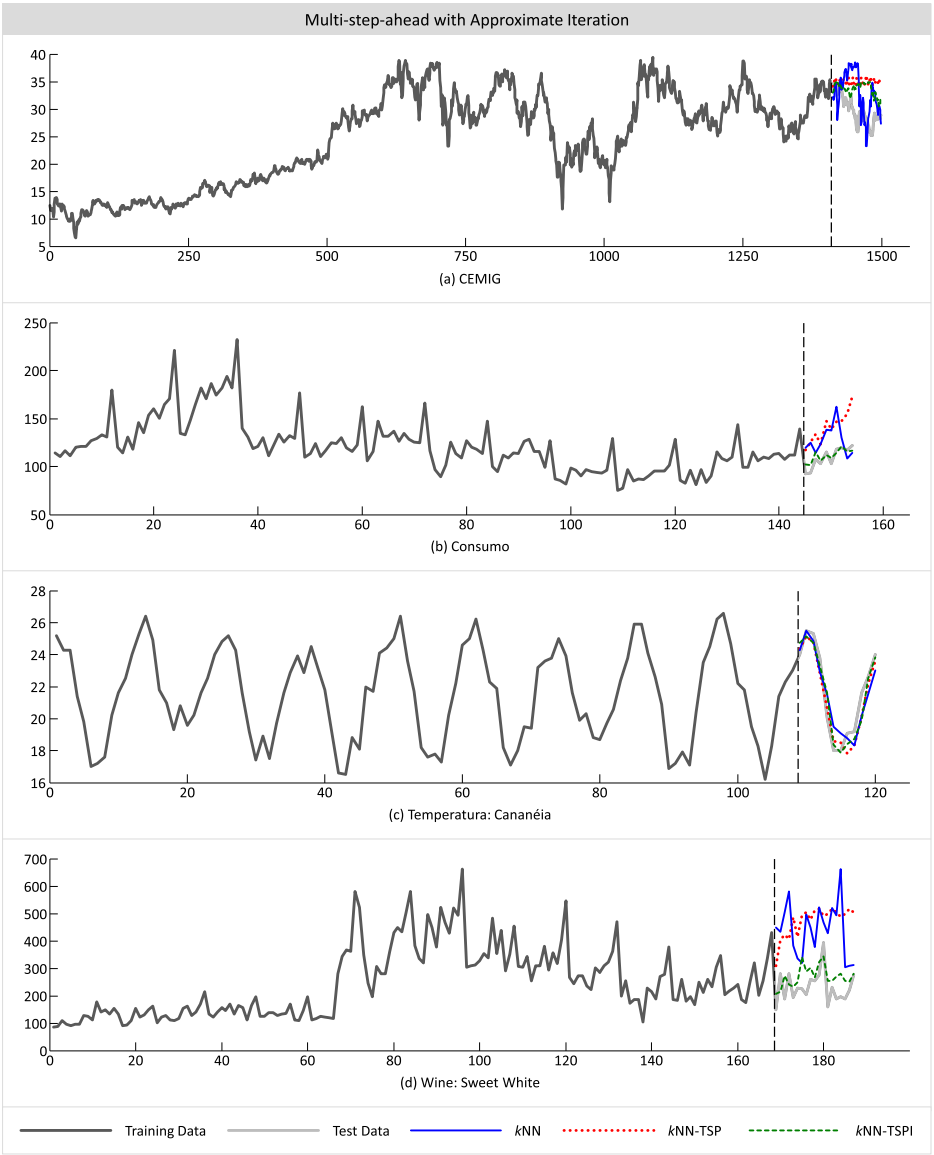


FIGURE 16. **Predictions for the out-of-sample period obtained from the multi-step-ahead projection strategy with approximate iteration using the similarity-based predictors.**

An optimization tool can also be used to estimate the values of l and k. In this case, we split the training set into training and validation sets, and the two parameters are chosen to minimize a performance measure for the validation data employing the training data. However, as with any estimation method, the best value in practice is not always selected.

We experimented with varying the k from 1 to 11 to show the impact of this parameter on the performance of kNN-TSPI regarding the other similarity-based predictors over the 55 datasets. To reach this goal, we fixed $l = max\_p$ for each dataset (Table 3). Fig. 17 exhibits the reached results considering the two investigated projection strategies.

As we can see in Fig. 17, regardless of the projection strategy, kNN-TSPI displayed the highest number of wins given the 55 datasets and different values of k. These results

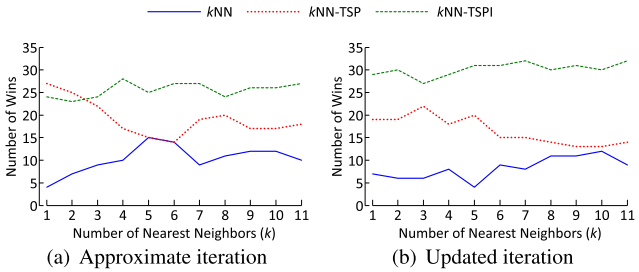


FIGURE 17. **Number of wins achieved by each similarity-based predictor given 55 datasets and the variation of parameter k from 1 to 11.**

allow us to state that, unlike the other two similarity-based predictors, kNN-TSPI is less sensitive to the parameter k. Thus, even though the three similarity-based methods have

presented similar predictive performances, the stability of our algorithm both regarding the parameter k and the accuracy of the projection horizon trends is a valuable advantage that justifies the use of techniques to obtain invariances to distortions in temporal data.

Finally, the low computational complexity of similarity-based methods makes them particularly attractive for real applications that require high efficiency. The global kNN regressor has a time complexity of $O(n \times l + n \times k)$, where n comprises the number of examples, l indicates the search window length, and k is the number of nearest neighbors. In contrast, the local kNN predictors have a time complexity of $O(m \times l)$, where m is the size of the time series.

B. kNN-TSPI VERSUS STATE-OF-THE-ART METHODS

To demonstrate that our predictor can provide results as good as other machine learning algorithms and statistical methods, we compared kNN-TSPI with 11 state-of-the-art models. The experiments carried out included 1,320 configurations (12 predictors × 2 projection strategies × 55 datasets).

Fig. 18 displays the averages and SD of the MSE values for each configuration and prediction strategy. REPTree provided the smallest prediction errors by adopting approximate iteration, while HWA obtained the largest ones. LSTM and HWM yielded, in this order, the best and worst results using updated iteration. The other methods achieved MSE values close to each other.

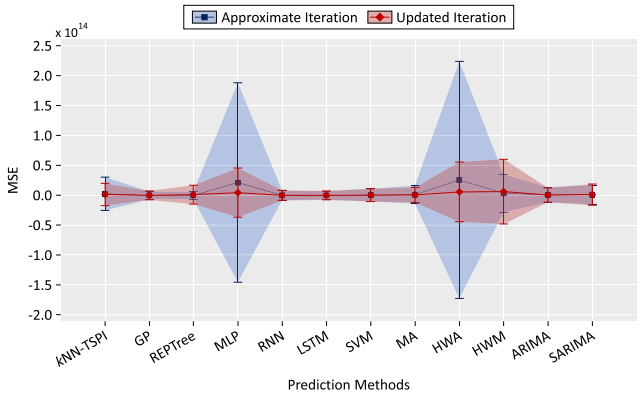
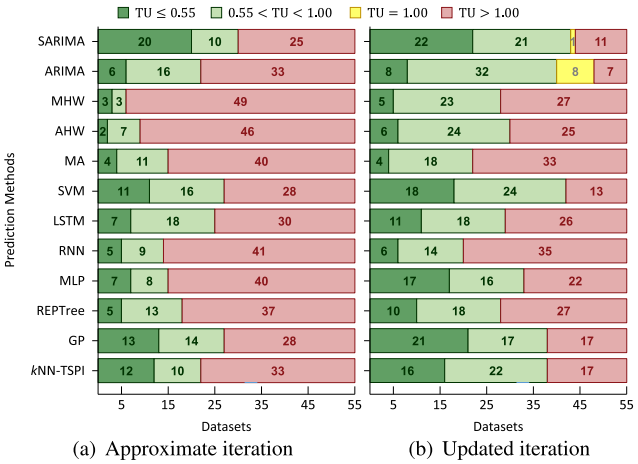


FIGURE 18. Averages and SD of MSE obtained by the state-of-the-art methods.

We sketch in Fig. 19 the TU values resulting from the predictors ran with the two projection strategies. Examining the approximate iteration (Fig. 19(a)), SARIMA achieved better results than the trivial model in 30 (20 + 10) of 55 datasets (TU < 1). GP, SVM, and kNN-TSPI outperformed the naive model (TU < 1) in 27 (13 + 14), 27 (11 + 16), and 22 (12 + 10) datasets, respectively. Looking at the updated iteration (Fig. 19(b)), SARIMA, GP, SVM, kNN-TSPI, and ARIMA overcame the trivial model by about 40 datasets (TU < 1).

Table 6 summarizes the average and SD values of POCID coming from the state-of-the-art algorithms. For the



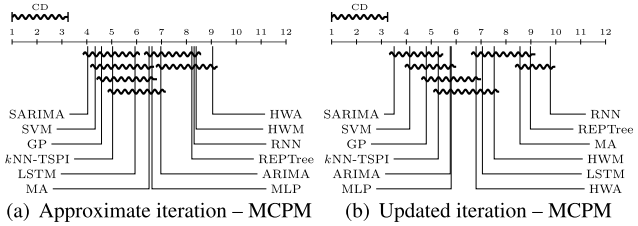
(a) Approximate iteration (b) Updated iteration

FIGURE 19. Performance, in terms of TU, of the state-of-the-art methods across the 55 datasets.

TABLE 6. Average and SD values of POCID achieved by the state-of-the-art methods.

Predictors	Approximate Iteration	Updated Iteration
kNN-TSPI	**61.86 (18.83)**	59.11 (18.34)
GP	59.36 (17.13)	56.42 (16.83)
REPTree	21.07 (24.10)	31.36 (20.58)
MLP	58.07 (18.04)	56.61 (17.96)
RNN	43.64 (17.50)	49.17 (15.31)
LSTM	50.73 (16.78)	51.62 (16.46)
SVM	58.85 (14.83)	55.65 (15.21)
MA	46.66 (11.44)	43.40 (17.26)
HWA	51.05 (13.68)	51.84 (14.82)
HWM	53.47 (12.27)	49.99 (14.07)
ARIMA	33.73 (29.91)	48.87 (16.83)
SARIMA	61.64 (26.29)	**61.27 (18.20)**

approximate iteration, we can see that our proposal reached the best POCID average (61.86% with SD = 18.83%), outperforming SARIMA (61.64% with SD = 26.29%) and MLP (58.07% with SD = 18.04%). SARIMA showed the best POCID values in terms of updated iteration (61.27% with SD = 18.20%), followed by kNN-TSPI (59.11% with SD = 18.34%). We must highlight that the error eventually propagated along the projection horizon influences SARIMA more than kNN-TSPI.



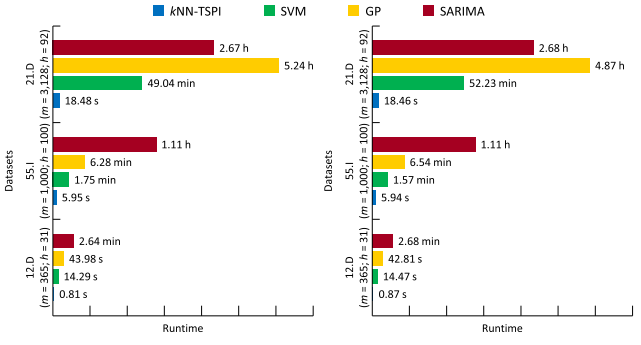
(a) Approximate iteration – MCPM (b) Updated iteration – MCPM

FIGURE 20. MCPM derived from the state-of-the-art methods.

Fig. 20 exhibits the MCPM values from the investigated methods by means of CD diagrams. The multi-criteria results reveal that SARIMA, SVM, GP, and kNN-TSPI are competitive forecasting algorithms with very close outputs.

Although SARIMA, SVM, and GP achieved a relatively higher overall precision than the similarity-based predictor without SSD, our proposal is more straightforward to encode, fit, and embed into any device. Most importantly, *k*NN-TSPI has only two parameters, whereas SARIMA has seven, and SVM and GP have three. Furthermore, the two input arguments of the algorithm with invariances are intuitive and easily determined based on the seasonal characteristic of the data. This property is highly desirable since the computational complexity is empirically related to the number of parameters.

To better illustrate the time efficiency of *k*NN-TSPI, we compared the time cost of our method against the most promising predictors: GP, SVM, and SARIMA. In this experiment, we chose three datasets with a small (ID = 12.D in Table 3), medium (ID = 55.I in Table 3), and large (ID = 21.D in Table 3) number of observations to see the behavior of the algorithms in different situations. We did not perform this experiment with all 55 datasets due to the infeasible time required to run each method individually and measure their cost fairly. However, we believe the three scenarios discussed are enough to present the efficiency of *k*NN-TSPI. Fig. 21 comprises more details about the evaluated data and the time costs spent by the algorithms. The times are in seconds (s), minutes (min), or hours (h), and each one of them represents the sum of time costs to search parameters, build the predictive model, and extrapolate the model for future periods. We performed the tests on a server running a 2.10 GHz Intel Xeon E5-2620 v4 processor (32-core) with 92 GB RAM and operational system Debian 4.9.130-2 (64 bits) under the same processing conditions for all measurements.



(a) Approximate iteration (b) Updated iteration

FIGURE 21. Time costs of the traditional methods from literature compared with our algorithm to predict time series with a small (12.D), medium (55.I), and large (21.D) length.

In Fig. 21, we can see that *k*NN-TSPI showed a drastically reduced time cost regarding the state-of-the-art predictors for both projection strategies. For example, given the dataset 21.D with 3,128 observations and a predictive horizon of 92 observations, while *k*NN-TSPI spent only 18.48 seconds in the approximate iteration, SVM exhibited a cost of 49.04 minutes, SARIMA spent 2.67 hours, and GP ran over 5.24 hours. Thus, although there is no statistical difference in the predictive power of these four methods, *k*NN-TSPI has an important advantage in terms of time efficiency.

C. EXPLORING SIMILARITY-BASED PREDICTION PROPERTIES WITH kNN-TSPI

This subsection reports a sequence of computational tests involving distinct invariances, distance measures, complexity-invariant distances, and ensemble functions to explore the intrinsic characteristics of the time series prediction via similarity. Besides helping to find the best settings for *k*NN-TSPI, these experiments also aim to present some insights that can guide research with similarity-based predictors.

1) DISTANCE MEASURES

Our proposal differs from other published methods by employing techniques to deal with amplitude, offset, and complexity invariances and treat trivial matches. In this context, we evaluated 25 distances to verify the similarity measure's influence on the *k*NN-TSPI performance. These combinations totaled 2,750 configurations (1 predictor × 25 similarity measures × 2 projection strategies × 55 datasets).

Fig. 22 displays the MSE results for each configuration and projection strategy. L_3, Chebychev, and Additive Symmetric χ^2 were the metrics that provided, on average, the best MSE values regarding the approximate iteration. However, they did not present a large difference margin compared with the usual distances, such as Euclidean, Cosine, Geodesic, Average Distance, CID, and DTW. In contrast, Sørensen, Squared χ^2, DTW-D, and CIDDTW showed the poorest MSE results; they seem to have less discriminatory power than the other measures.

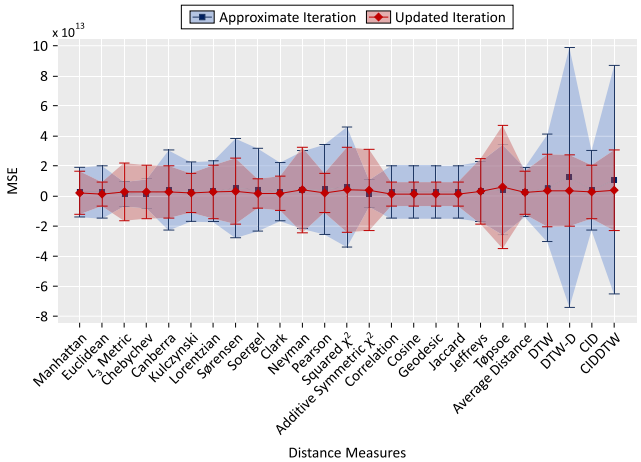
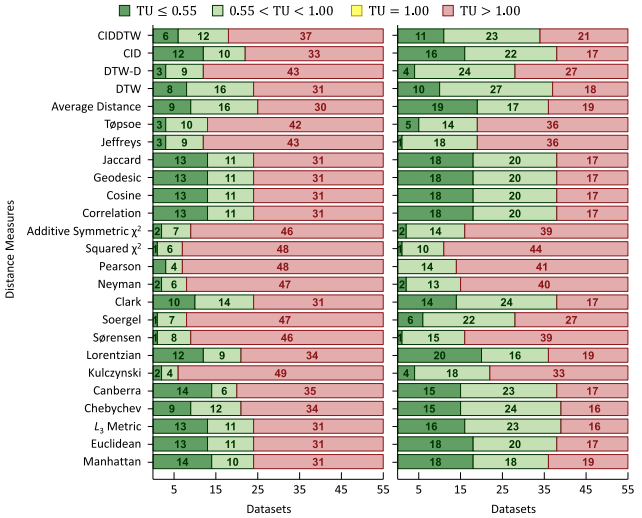


FIGURE 22. Averages and SD of MSE obtained by *k*NN-TSPI using different similarity measures.

Euclidean and the distances from the internal product category (Correlation, Cosine, Geodesic, and Jaccard) recorded the best performances in terms of updated iteration. Topsøe, in turn, implied the highest prediction errors.

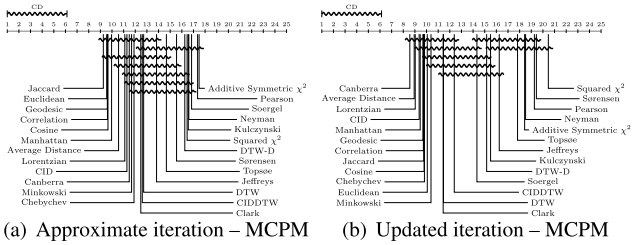


(a) Approximate iteration (b) Updated iteration

FIGURE 23. Performance, in terms of TU, of *k*NN-TSPI employing distinct similarity measures.

TABLE 7. Average and SD values of POCID achieved by *k*NN-TSPI adopting several similarity measures.

Similarity Measure	Approximate Iteration	Updated Iteration
Manhattan	58.78 (19.53)	58.75 (19.08)
Euclidean	60.44 (19.81)	58.39 (18.17)
Minkowski	58.93 (20.58)	58.18 (18.95)
Chebychev	59.10 (21.52)	57.77 (18.85)
Canberra	60.06 (20.84)	59.11 (18.56)
Kulczynski	54.94 (12.66)	51.59 (15.77)
Lorentzian	57.33 (19.79)	58.85 (19.82)
Sørensen	49.02 (15.59)	49.35 (13.82)
Soergel	53.80 (14.28)	53.23 (14.55)
Clark	54.54 (16.86)	55.47 (18.51)
Neyman	46.94 (12.88)	46.68 (13.55)
Pearson	47.95 (13.30)	48.27 (14.37)
Squared χ^2	47.02 (13.50)	50.52 (14.26)
Additive Symmetric χ^2	47.11 (13.93)	45.63 (13.58)
Correlation	61.05 (18.80)	58.82 (17.88)
Cosine	60.93 (18.91)	58.71 (17.87)
Geodesic	61.01 (18.66)	58.60 (17.90)
Jaccard	60.93 (18.99)	58.82 (17.92)
Jeffreys	51.76 (12.68)	49.57 (13.77)
Topsøe	49.35 (15.53)	49.41 (15.60)
Average Distance	60.71 (18.98)	58.32 (19.89)
DTW	55.90 (18.05)	55.88 (17.54)
DTW-D	51.76 (13.61)	49.79 (14.51)
CID	**61.86 (18.83)**	**59.11 (18.34)**
CIDDTW	55.90 (17.11)	54.55 (18.10)



(a) Approximate iteration – MCPM (b) Updated iteration – MCPM

FIGURE 24. MCPM derived from *k*NN-TSPI configured with various similarity measures.

In Fig. 23(a), for the approximate iteration strategy, the four ranges of TU values show that the metrics belonging to the L_p norm (Manhattan, Euclidean, Minkowski, and Chebychev) and internal product (Correlation, Cosine, Geodesic, and Jaccard), and Canberra, Lorentzian, Clark, Average Distance, DTW, and CID performed well in about 23 datasets (TU $<$ 1), of which 12 showed reliable modeling (TU $\leq$ 0.55). The results of these distances and CIDDTW with updated iteration (Fig. 23(b)) were adequate, on average, for 37 datasets (TU $<$ 1), of which 16 led to reliable predictive models (TU $\leq$ 0.55).

The average and SD values of POCID are arranged, for each *k*NN-TSPI configuration and projection strategy, in Table 7. CID provided the best average hit rate considering approximate iteration (61.86% with SD = 18.83%), while Neyman produced the worst average performance index (46.94% with SD = 12.88%). As for the updated iteration, CID obtained the best POCID average (59.11% with SD = 18.34%), while Additive Symmetric χ^2 recorded the worst outputs (45.63% with SD = 13.58%). Therefore, CID culminated in the best results regardless of the projection strategy adopted.

Fig. 24 exhibits the CD diagrams concerning the MCPM values coming from *k*NN-TSPI configured with several distances measures. The distances belonging to the internal product family (Correlation, Cosine, Geodesic, and Jaccard) and Euclidean provided the smallest prediction errors for *k*NN-TSPI with approximate iteration. This fact was also observed for Canberra, Average Distance, Lorentzian, CID, and Manhattan in relation to *k*NN-TSPI with updated iteration. None of the said distances demonstrated SSD compared with the metrics commonly employed for classification and clustering, *e.g.*, Manhattan, DTW, and CIDDTW. It is important to emphasize that CID was very competitive and, regardless of the projection strategy, it provided the *k*NN-TSPI with the best hit rates regarding the extrapolated horizon trends.

In agreement with the overall results, we recommend using distances that have a linear time complexity. For instance, in addition to its simplicity and fast running times, CID performed consistently better than the other similarity measures in terms of POCID.

Although DTW has been applied successfully to various tasks, such as classification, clustering and anomaly detection, our study did not identify the potentiality of this nonlinear distance for time series prediction. We believe that the poor performance of DTW is because it cannot align short subsequences properly.

2) COMPLEXITY MEASURES

As discussed in Section IV, CID uses a complexity estimate founded on the physical intuition that a series can be "stretched" until it becomes a straight line. Following this reasoning, a complex sequence would result in a straight line longer than a simple series. This complexity estimate, aka

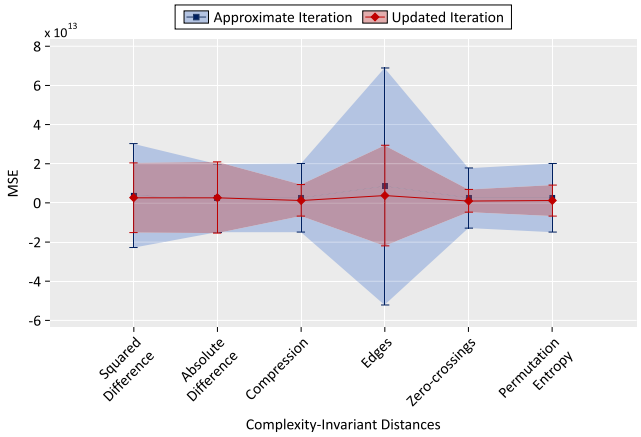


FIGURE 25. Averages and SD of MSE obtained by *k*NN-TSPI using different complexity measures.

Squared Difference, can assign higher distances to subsequences with distinct complexities.

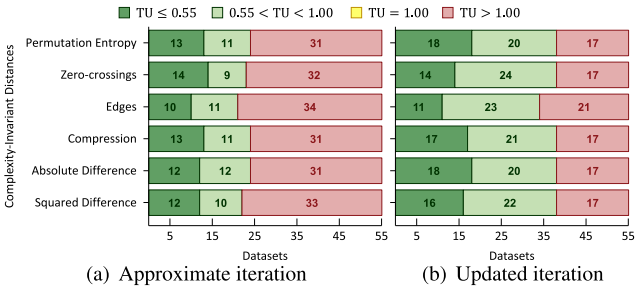
To assess the influence of the complexity measures in the similarity search performed by *k*NN-TSPI, we compared five other estimates besides the Squared Difference adopted by the original CID. The experiments derived from these applications totaled 660 configurations (1 predictor × 6 complexity measures × 2 projection strategies × 55 datasets).

Fig. 25 illustrates the MSE averages and their respective SD for each *k*NN-TSPI configuration and projection strategy. Examining the approximate iteration, Zero-crossings, Absolute Difference, and Compression provided the best overall performances. However, their values were not so far removed from Permutation Entropy and Squared Difference. Edges recorded the worst MSE results.

In the scenario where we iteratively update the models with actual values, Edges showed the worst results. Note that some measures like Compression, Zero-crossings, and Permutation Entropy presented comparable results with Squared Difference. Nevertheless, Squared Difference is conceptually more straightforward.

In Fig. 26, the ranges of values obtained by the TU coefficient demonstrated that, among all the complexity estimates, Edges implied the most unsatisfactory results. Specifically, for the approximate iteration (Fig. 26(a)), *k*NN-TSPI configured with the referenced measure was not preferable to the trivial model in 34 of 55 datasets (TU $>$ 1). Applying the updated iteration (Fig. 26(b)), Edges was adequate (TU $<$ 1) to model and predict 34 (11 + 23) datasets, of which only 11 resulted in reliable models.

We can see in Table 8 that, for both projection strategies, the average and SD values of POCID are evenly distributed among the six estimates of complexity. In other words, using any of these measures provided the *k*NN-TSPI with a POCID average of 59.21% (SD $\approx$ 18.27%). Considering all the complexity estimates, Squared Difference exhibited the best results via approximate (61.86% with SD = 18.83%) and updated (59.11% with SD = 18.34%) iterations.

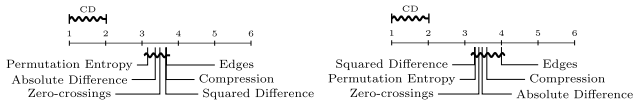


(a) Approximate iteration (b) Updated iteration

FIGURE 26. Performance, in terms of TU, of *k*NN-TSPI employing distinct complexity measures.

TABLE 8. Average and SD values of POCID achieved by *k*NN-TSPI adopting several complexity estimates.

Complexity Measure	Approximate Iteration	Updated Iteration
Squared Difference	**61.86 (18.83)**	**59.11 (18.34)**
Absolute Difference	61.04 (17.69)	57.50 (17.16)
Compression	58.16 (20.12)	58.91 (17.77)
Edges	59.53 (17.17)	57.05 (18.05)
Zero-crossings	60.84 (19.72)	58.30 (17.06)
Permutation Entropy	60.15 (20.02)	58.10 (17.29)



(a) Approximate iteration – MCPM (b) Updated iteration – MCPM

FIGURE 27. MCPM derived from *k*NN-TSPI configured with various complexity measures.

Fig. 27 shows the CD diagrams that represent the MCPM values resulting from *k*NN-TSPI employing the six complexity measures. Squared Difference and Absolute Difference did not present SSD compared with the other complexities. For the encoding simplicity and providing the best hit rates on future trends, we recommend using CID with Squared Difference, whose good performance was also verified in [7].

3) ENSEMBLE FUNCTIONS

The ensemble function used by *k*NN-TSPI should allow the projection of data that present variation in amplitude over time. Thus, we analyzed four prediction functions: Median; MAV; MRV; IW, which consists of applying coefficients weighted by the temporal index of the compared subsequence; and DW, which is an average weighted by the values of the computed distances (d). For the latter case, we investigated six weight variations (Table 9). The performed combinations resulted in 1,100 configurations (1 predictor × 10 ensemble functions × 2 projection strategies × 55 datasets).

Fig. 28 displays the MSE averages and their respective SD for each configuration and projection strategy. Analyzing both projection strategies, MRV provided the *k*NN-TSPI with the smallest prediction errors. Median, in turn, was the data

TABLE 9. Pre-established weights for the DW function.

ID	Weight (w)
1	$w = 1/d(Q,S)$
2	$w = 1/d(Q,S)^2$
3	$w = \exp(-d(Q,S)^2)$
4	$w = \exp(-d(Q,S)^2/(2\sigma^2))$ with $\sigma = 0.5$
5	$w = \exp(-d(Q,S)/(2\sigma^2))$ with $\sigma = 0.5$
6	$w = \exp(-d(Q,S)/\sigma)$ with $\sigma = 0.5$

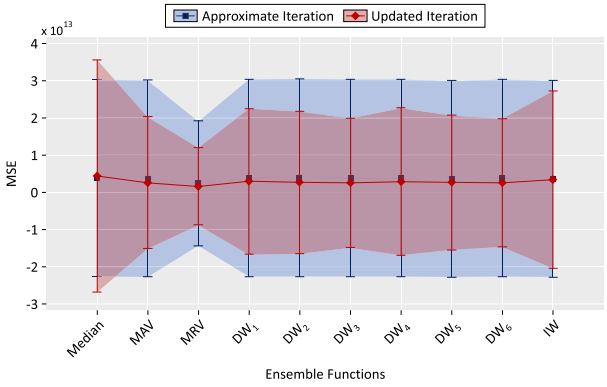


FIGURE 28. Averages and SD of MSE obtained by *k*NN-TSPI using different ensemble functions.

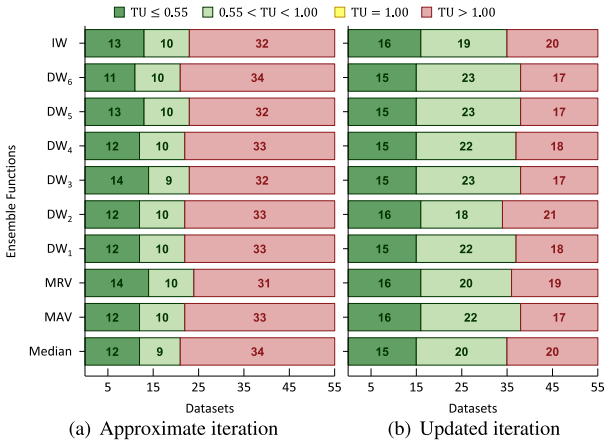


FIGURE 29. Performance, in terms of TU, of *k*NN-TSPI employing distinct ensemble functions.

fusion function that led to the worst results. DW_3, DW_5, and DW_6 were the most stable ensemble functions.

In Fig. 29, the TU statistics show that with approximate iteration (Fig. 29(a)), MRV associated with *k*NN-TSPI promoted, for 24 (14 + 10) of the total of 55 datasets, the lowest TU values (TU < 1). As for the updated iteration (Fig. 29(b)), MAV, DW_3, DW_5, and DW_6 recorded, for 38 of the total of 55 datasets, the best performances (TU < 1).

From the average and SD values of POCID (Table 10), it is possible to verify that the results are distributed slightly among the ten ensemble functions for both projection strategies. This means that using any of these ensemble functions by *k*NN-TSPI resulted in an average hit rate over the projection horizon trends of approximately 60.05% (SD ≈ 19.19%).

TABLE 10. Average and SD values of POCID achieved by *k*NN-TSPI adopting several ensemble functions.

Ensemble Function	Approximate Iteration	Updated Iteration
Median	61.71 (18.49)	57.78 (19.03)
MAV	61.86 (18.83)	59.11 (18.34)
MRV	60.65 (21.74)	57.32 (17.09)
DW_1	60.68 (20.12)	58.82 (18.93)
DW_2	59.86 (21.27)	**60.32 (18.58)**
DW_3	60.75 (20.21)	59.00 (19.03)
DW_4	60.94 (20.01)	58.77 (18.88)
DW_5	62.24 (19.68)	58.53 (18.40)
DW_6	62.35 (18.50)	57.49 (19.37)
IW	**62.70 (18.44)**	60.13 (18.79)

Notably, in the prediction via approximate iteration, the use of IW implied the best performances (62.70% with SD = 18.44%). Differently, in the prediction conducted by updated iteration, the best results were obtained employing DW_2 (60.32% with SD = 18.58%).

Fig. 30 illustrates the CD diagrams given the MCPM total area values of *k*NN-TSPI configured with the ten ensemble functions. In the matter of prediction error and POCID rates, DW_3 was the more stable ensemble function regardless of the projection strategy applied. IW, DW_5, and MRV were also competitive in relation to the other prediction functions. In this context, DW_3 and IW are, in general terms, good candidates for ensemble function. IW also has the advantage of assigning larger weights to the most recent observations.

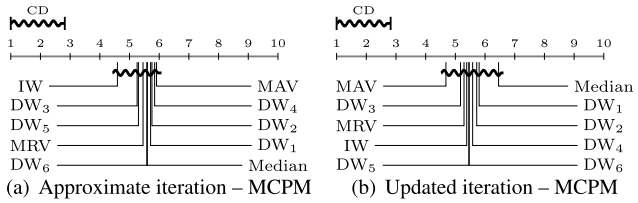


FIGURE 30. MCPM derived from *k*NN-TSPI configured with various ensemble functions.

On the one hand, MAV allowed *k*NN-TSPI with approximate iteration to obtain the best multi-criteria results (Fig. 30(a)). On the other hand, this same ensemble function occupied the last position in the multi-criteria performance ranking for the multi-step-ahead strategy with updated iteration (Fig. 30(b)). These facts demonstrate that the MAV's performance is highly influenced by the error eventually propagated along the projection horizon.

VII. CONCLUSION

This paper presented *k*NN-TSPI, a novel similarity-based time series prediction method that explores invariances to achieve more reliable results. We demonstrated that the correct combination of amplitude, offset, and complexity invariances, together with a solution to treat trivial matches, leads to better matches between reference queries and data subsequences.

*k*NN-TSPI is univariate and suitable for short-term prediction. The algorithm can be implemented quickly and has

only two parameters: the query length (l) and the number of similar subsequences (k). The time complexity of kNN-TSPI is $O(m \times l)$, where m is the time series size.

We faced our predictor with two previous similarity-based methods and 11 state-of-the-art models considering 55 real datasets and four evaluation measures. In addition to being more stable than the similarity-based algorithms for different values of k and competing with the state-of-the-art predictors, our method is simpler to explain, adjust, and embed on any device. We also performed a comprehensive experimental assessment to understand the aspects intrinsic to the similarity-based time series prediction task, such as invariances to distortions, distance measures, complexity measures applied to CID, and ensemble functions.

The lessons learned are valuable to all researchers wishing to apply kNN-TSPI or any other similarity-based algorithm. Furthermore, the theoretical-practical foundation that sustains our study can be extended to various artificial intelligence techniques.

As future work, we want to investigate the use of predictor committees to compose kNN-TSPI's predictions to improve the final projection. Adapting kNN-TSPI to handle multivariate data is also part of our plans.

REFERENCES

[1] A. R. S. Parmezan, V. M. Souza, and G. E. Batista, "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," *Inf. Sci.*, vol. 484, pp. 302–337, May 2019.

[2] G. Ristanoski, W. Liu, and J. Bailey, "A time-dependent enhanced support vector machine for time series regression," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Chicago, Illinois, Aug. 2013, pp. 946–954.

[3] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econ. Rev.*, vol. 29, nos. 5–6, pp. 594–621, Sep. 2010.

[4] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control.*, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[5] M. N. Islam and B. Sivakumar, "Characterization and prediction of runoff dynamics: A nonlinear dynamical view," *Adv. Water Resour.*, vol. 25, no. 2, pp. 179–190, Feb. 2002.

[6] R. M. Marcacini, J. C. Carnevali, and J. Domingos, "On combining websensors and DTW distance for kNN time series forecasting," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Cancun, Mexico, Dec. 2016, pp. 2521–2525.

[7] A. R. S. Parmezan and G. E. Batista, "A study of the use of complexity measures in the similarity search process adopted by kNN algorithm for time series prediction," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Miami, Florida, Dec. 2015, pp. 45–51.

[8] E. Fix and J. L. Hodges, "Discriminatory analysis, nonparametric discrimination, consistency properties," U.S. Air Force School Aerosp. Med., Randolph Field, Tech. Rep. 4, 1951.

[9] J. G. D. Gooijer and R. J. Hyndman, "25 years of time series forecasting," *Int. J. Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[10] C. Lemke and B. Gabrys, "Meta-learning for time series forecasting and forecast combination," *Neurocomputing*, vol. 73, nos. 10–12, pp. 2006–2016, Jun. 2010.

[11] H. Cheng and P.-N. Tan, "Semi-supervised learning with data calibration for long-term time series forecasting," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Las Vegas, Nevada, 2008, pp. 133–141.

[12] D. F. Silva, V. M. A. Souza, D. P. W. Ellis, E. J. Keogh, and G. E. Batista, "Exploring low cost laser sensors to identify flying insect species," *J. Intell. Robotic Syst.*, vol. 80, pp. 313–330, Dec. 2015.

[13] J. Serrà and J. L. Arcos, "An empirical evaluation of similarity measures for time series classification," *Knowl.-Based Syst.*, vol. 67, pp. 305–314, Sep. 2014.

[14] R. Giusti and G. E. Batista, "An empirical comparison of dissimilarity measures for time series classification," in *Proc. Brazilian Conf. Intell. Syst.*, Fortaleza, Brazil, Oct. 2013, pp. 82–88.

[15] J. A. Junior, "Estudo da influência de diversas medidas de similaridade na previsão de séries temporais utilizando o algoritmo kNN-TSP," M.S. thesis, Centro de Engenharias e Ciências Exatas, Univ. Estadual do Oeste do Paraná, Foz do Iguaçu, Brazil, 2012.

[16] C. A. Ferrero, "Algoritmo kNN para previsão de dados temporais: Funções de previsão e critérios de seleção de vizinhos próximos aplicados à variáveis ambientais em limnologia," M.Sc. thesis, Instituto de Ciências Matemáticas e de Computação, Univ. de São Paulo, São Paulo, Brazil, 2009.

[17] C. A. S. Coelho, S. Pezzulli, M. Balmaseda, F. J. Doblas-Reyes, and D. B. Stephenson, "Forecast calibration and combination: A simple Bayesian approach for ENSO," *J. Climate*, vol. 17, no. 7, pp. 1504–1516, Apr. 2004.

[18] A. R. S. Parmezan and G. E. A. P. A. Batista, "ICMC-USP time series prediction repository," Instituto de Ciências Matemáticas e de Computação, Univ. de São Paulo, São Carlos, Brazil, 2014. [Online]. Available: http://sites.labic.icmc.usp.br/icmc_tspr/

[19] F. Martínez, M. P. Frías, M. D. Pérez, and A. J. Rivera, "A methodology for applying k-nearest neighbor to time series forecasting," *Artif. Intell. Rev.*, vol. 52, no. 3, pp. 2019–2037, 2019.

[20] B. Yu, X. Song, F. Guan, Z. Yang, and B. Yao, "K-nearest neighbor model for multiple-time-step prediction of short-term traffic condition," *J. Transp. Eng.*, vol. 142, no. 6, 2016, Art. no. 04016018.

[21] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," in *Proc. Int. Conf. Very Large Data Bases*, Auckland, New Zealand, 2008, pp. 1542–1552.

[22] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining Knowl. Discovery*, vol. 31, no. 3, pp. 606–660, May 2017.

[23] Q. Zhu, G. Batista, T. Rakthanmanon, and E. Keogh, "A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets," in *Proc. SIAM Int. Conf. Data Mining*, Anaheim, CA, USA, Apr. 2012, pp. 999–1010.

[24] V. Chandola, D. Cheboli, and V. Kumar, "Detecting anomalies in a time series database," Dept. Comput. Sci. Eng., Univ. Minnesota, Minneapolis, Tech. Rep. 09-004, 2009.

[25] D. F. Silva, R. Giusti, E. Keogh, and G. E. Batista, "Speeding up similarity search under dynamic time warping by pruning unpromising alignments," *Data Mining Knowl. Discovery*, vol. 32, no. 4, pp. 988–1016, Jul. 2018.

[26] G. E. Batista, E. J. Keogh, O. M. Tataw, and V. M. A. de Souza, "CID: An efficient complexity-invariant distance for time series," *Data Mining Knowl. Discovery*, vol. 28, no. 3, pp. 634–669, 2014.

[27] D. Sovilj, A. Sorjamaa, Q. Yu, Y. Miche, and E. Séverin, "OPELM and OPKNN in long-term prediction of time series using projected input data," *Neurocomputing*, vol. 73, nos. 10–12, pp. 1976–1986, Jun. 2010.

[28] Z. Huang and M.-L. Shyu, "K-NN based LS-SVM framework for long-term time series prediction," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Las Vegas, Nevada, Aug. 2010, pp. 69–74.

[29] A. Shi and B. Zhou, "K-nearest neighbor LS-SVM method for multi-step prediction of chaotic time series," in *Proc. IEEE Symp. Electr. Electron. Eng. (EEESYM)*, Kuala Lumpur, Malaysia, Jun. 2012, pp. 407–409.

[30] M. V. N. Bedo, D. P. D. Santos, D. S. Kaster, and C. Traina, "A similarity-based approach for financial time series analysis and forecasting," in *Proc. Int. Conf. Database Exp. Syst. Appl.* Prague, Czech Republic: Springer, 2013, pp. 94–108.

[31] C.-C. Wei, T.-T. Chen, and S.-J. Lee, "K-NN based neuro-fuzzy system for time series prediction," in *Proc. 14th ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput.*, Honolulu, Hawaii, Jul. 2013, pp. 569–574.

[32] E. De La Vega, J. J. Flores, and M. Graff, "K-nearest-neighbor by differential evolution for time series forecasting," in *Proc. Mex. Int. Conf. Artif. Intell.* Tuxtla Gutiérrez, Mexico: Springer, 2014, pp. 50–60.

[33] J. J. Flores, J. Ortiz, J. R. C. Gonzalez, C. Lara, and R. L. Farias, "FNN a fuzzy version of the nearest neighbor time series forecasting technique," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, Ixtapa, Mexico, Nov. 2015, pp. 1–6.

[34] R. L. Talavera-Llames, R. Pérez-Chacón, M. Martínez-Ballesteros, A. Troncoso, and F. Martínez-Álvarez, "A nearest neighbours-based algorithm for big time series data forecasting," in *Proc. Int. Conf. Hybrid Artif. Intell. Syst.* Seville, Spain: Springer, 2016, pp. 174–185.
[35] Z. Wang and I. Koprinska, "Solar power prediction with data source weighted nearest neighbors," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, Alaska, May 2017, pp. 1411–1418.
[36] C.-R. Chen and U. Kartini, "K-nearest neighbor neural network models for very short-term global solar irradiance forecasting based on meteorological data," *Energies*, vol. 10, no. 2, p. 186, Feb. 2017.
[37] L. Tang, H.-P. Pan, and Y.-Y. Yao, "Computational intelligence prediction model integrating empirical mode decomposition, principal component analysis, and weighted k-nearest neighbor," *J. Electron. Sci. Technol.*, vol. 18, no. 4, pp. 341–349, 2021.
[38] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition," *Exp. Syst. Appl.*, vol. 39, no. 8, pp. 7067–7083, Jun. 2012.
[39] L. Wang, Y. Zeng, and T. Chen, "Back propagation neural network with adaptive differential evolution algorithm for time series forecasting," *Exp. Syst. Appl.*, vol. 42, no. 2, pp. 855–863, 2015.
[40] W. Huang, K. K. Lai, Y. Nakamori, and S. Wang, "Forecasting foreign exchange rates with artificial neural networks: A review," *Int. J. Inf. Technol. Decis. Making*, vol. 3, no. 1, pp. 145–165, Mar. 2004.
[41] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.
[42] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 24–38, May 2009.
[43] D. T. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining*. Hoboken, NJ, USA: Wiley, 2004.
[44] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2011.
[45] C. Cheng and F. Xiao, "A distance for belief functions of orderable set," *Pattern Recognit. Lett.*, vol. 145, pp. 165–170, May 2021.
[46] R. Xu and D. C. Wunsch, *Clustering* (IEEE Series Computational Intelligence). Hoboken, NJ, USA: Wiley, 2009.
[47] M. L. Hetland, "A survey of recent methods for efficient retrieval of similar time sequences," in *Data Mining in Time Series Databases* (Series in Machine Perception and Artificial Intelligence), vol 57. Danvers, MA, USA: World Scientific, 2004, ch. 2, pp. 27–49.
[48] S. H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *Int. J. Math. Models Methods Appl. Sci.*, vol. 1, no. 4, pp. 300–307, Nov. 2007.
[49] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Database Theory* (Lecture Notes in Computer Science), vol. 1973. London, U.K.: Springer, 2001, pp. 420–434.
[50] M. Vlachos, D. Gunopulos, and G. Das, "Indexing time-series under conditions of noise," in *Data Mining in Time Series Databases*. Danvers, MA, USA: World Scientific, 2004, pp. 67–100.
[51] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Washington, 2003, pp. 216–225.
[52] L. Chen and R. Ng, "On the marriage of Lp-norms and edit distance," in *Proc. 30th Int. Conf. Very Large Data Bases*, vol. 30, Toronto, CA, USA, 2004, pp. 792–803.
[53] M. D. Morse and J. M. Patel, "An efficient and accurate method for evaluating time series similarity," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Beijing, China, 2007, pp. 569–580.
[54] P.-F. Marteau, "Time warp edit distance with stiffness adjustment for time series matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 306–318, Feb. 2009.
[55] A. Stefan, V. Athitsos, and G. Das, "The move-split-merge metric for time series," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1425–1438, Jun. 2013.
[56] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowl. Inf. Syst.*, vol. 7, no. 3, pp. 358–386, 2005.
[57] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, no. 1, pp. 43–49, Feb. 1978.
[58] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-23, no. 1, pp. 67–72, Feb. 1975.
[59] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover, "Exact discovery of time series motifs," in *Proc. SIAM Int. Conf. Data Mining*, Sparks, Apr. 2009, pp. 473–484.
[60] G. E. Batista, B. Campana, and E. Keogh, "Classification of live moths combining texture, color and shape primitives," in *Proc. 9th Int. Conf. Mach. Learn. Appl.*, Washington, Dec. 2010, pp. 903–906.
[61] Y. Chen, E. J. Keogh, and G. E. Batista, "DTW-D: Time series semi-supervised learning from a single example," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Chicago, Illinois, 2013, pp. 383–391.
[62] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery (DMKD)*, New York, NY, USA, 2003, pp. 2–11.
[63] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Upper Saddle River, NJ, USA: Prentice-Hall, 1978.
[64] C. Bandt and B. Pompe, "Permutation entropy: A natural complexity measure for time series," *Phys. Rev. Lett.*, vol. 88, no. 17, pp. 1–4, Apr. 2002.
[65] A. R. S. Parmezan, "Predição de séries temporais por similaridade," M.Sc. thesis, Instituto de Ciências Matemáticas e de Computação, Univ. de São Paulo, São Paulo, Brazil, 2016.
[66] A. R. S. Parmezan, H. D. Lee, and F. C. Wu, "Metalearning for choosing feature selection algorithms in data mining: Proposal of a new framework," *Exp. Syst. Appl.*, vol. 75, pp. 1–24, Jun. 2017.
[67] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.

ANTONIO R. S. PARMEZAN received the Ph.D. degree in computer science from the University of São Paulo (USP), Brazil, in 2022. He is currently a Postdoctoral Researcher with the Computational Intelligence Group, USP. His publications have appeared in a number of international journals, such as *Information Sciences*, *Expert Systems with Applications*, *Knowledge-Based Systems*, and *Knowledge and Information Systems*. His research interests include data mining, machine learning, time series analysis, data stream processing, feature selection, and metalearning.

VINICIUS M. A. SOUZA received the Ph.D. degree in computer science from the University of São Paulo, Brazil, in 2016. He was a Postdoctoral Researcher at the University of New Mexico, USA. He is an Assistant Professor with the Graduate Program in Informatics, Pontifical Catholic University of Paraná, Brazil. He has authored papers in top-tier peer-reviewed conferences and journals, including *Data Mining and Knowledge Discovery*, *Information Sciences*, ACM SIGKDD, and IEEE International Conference on Data Mining (ICDM). His research interests include data mining, data streams, and time series.

GUSTAVO E. A. P. A. BATISTA received the Ph.D. degree in computer science from the University of São Paulo, Brazil, in 2003. In 2007, he joined the University of São Paulo as an Assistant Professor and became an Associate Professor in 2016. In 2018, he joined as an Associate Professor with the University of New South Wales, Australia. He has more than 100 papers in conferences and journals. He has served as Program Committee Member of conferences, such as ACM SIGKDD, IEEE International Conference on Data Mining (ICDM) and IJCAI. He is a member of the Editorial Board of the *Machine Learning* Journal.