Data Mining and Knowledge Discovery (2025) 39:27
https://doi.org/10.1007/s10618-025-01098-3

TSRF-Dist: a novel time series distance based on extremely randomized canonical interval forests

Alberto Azzari[1] · Manuele Bicego[1] · Carlo Combi[1] · Andrea Cracco[1] · Pietro Sala[1]

Received: 10 October 2024 / Accepted: 18 March 2025 / Published online: 15 April 2025

Abstract

This paper presents TSRF-Dist, a novel distance between time series based on Random Forests (RFs). We extend to the time-series domain concepts and tools of RF distances, a recent class of robust data-dependent distances defined for vectorial representations, thus proposing the *first* RF distance for time series. The distance is determined by (i) creating an RF to model a set of time series, and (ii) exploiting the trained RF to quantify the similarity between time series. As for the first step, we introduce in this paper the *Extremely Randomized Canonical Interval Forest* (ERCIF), a novel extension of Canonical Interval Forests that can model time series and can be trained without labels. We then exploit three different schemes, following ideas already employed in the vectorial case. The proposed distance, in different variants, has been thoroughly evaluated with 128 datasets from the `UCR Time Series` archive, showing promising results compared with literature alternatives.

Keywords Time series · Distances · Random forests · Extremely randomized trees

Responsible editor: Eamonn Keogh.

Alberto Azzari, Manuele Bicego, Carlo Combi, Andrea Cracco, Pietro Sala have contributed equally to this work.

✉ Alberto Azzari
alberto.azzari@univr.it

Manuele Bicego
manuele.bicego@univr.it

Carlo Combi
carlo.combi@univr.it

Andrea Cracco
andrea.cracco@univr.it

Pietro Sala
pietro.sala@univr.it

[1] Department of Computer Science, University of Verona, Strada le Grazie 15, 37134 Verona, Veneto, Italy

1 Introduction

The analysis of time series represents a fundamental process in Pattern Recognition and Data Mining (Mishra et al. 2022; Zhao et al. 2019; Ruiz et al. 2021), crucial in many different application scenarios, like signal processing (Lu et al. 2023), medical data analysis (Morid et al. 2023; Shehab et al. 2022), weather forecasting (Karevan and Suykens 2020), human activity recognition (Chen et al. 2021) and bioinformatics (Mitra and MacLean 2021), just to cite a few. In this context, a task that has received great attention in the literature is the definition of a proper distance between time series, being often the basis of more complex operations such as classification or clustering. Several measures have been proposed in the past (Holder et al. 2024; Wang et al. 2013; Esling and Agon 2012), characterized by different features in terms of employed concepts, obtained performances, and computational requirements—see Sect. 2 for a brief overview. In this paper, we contribute to this field by proposing a novel distance between time series based on Random Forests (RFs) (Breiman 2001; Criminisi et al. 2012). These models, which represent ensembles of decision trees (Quinlan 1993), are typically employed in supervised tasks like regression or classification. In recent years, however, a great interest has arisen in the exploitation of Random Forests also in unsupervised contexts, such as outlier detection (Liu et al. 2008, 2012), clustering (Moosmann et al. 2006; Shotton et al. 2008; Yan et al. 2013; Bicego 2019; Shi and Horvath 2006; Zhu et al. 2014; Bicego et al. 2023), and, crucially, distance computation. Breiman, in his seminal work (Breiman 2001), was the first to suggest the possibility of deriving powerful data-dependent similarity measures from Random Forests. After his intuition, many other RF distances have been proposed (see, e.g., Shi and Horvath 2006; Zhu et al. 2014; Ting et al. 2016; Aryal et al. 2020; Ting et al. 2019; Aryal et al. 2014, 2017; Bicego et al. 2023), shown to be effective in classification, clustering, outlier detection and other tasks. All these distances are defined on vectors, i.e., they permit the derivation of a distance measure between objects described with a *vectorial* representation. Among these, Proximity Forest (Lucas et al. 2019) extends RF-based approaches to time series classification by leveraging distance-based splits, while Proximity Isolation Forest (Mensi et al. 2023) extends the isolation mechanism to define a proximity-based anomaly detection method applicable to time series. In this paper, we take one step forward and export concepts and tools for RF distances to the time series domain, proposing the first RF distance for time series, which we call TSRF-Dist.

Following the classical scheme for the definition of vectorial RF distances, the TSRF-Dist distance is derived in two steps: first, we create a Random Forest starting from a training set of time series; then, given two time series, we compute their distance through the trained Forest. More in detail, in the first step, we define the *Extremely Randomized Canonical Interval Forest* (ERCIF), a Random Forest which can model time series and can be trained without labels. Our model starts from the Canonical Interval Forest (Middlehurst et al. 2020), an RF for time series defined for classification, and extends it in different directions. The main change is to adapt

it to the unsupervised case with a training scheme that follows the Extremely Randomized Trees (ERT) paradigm (Geurts et al. 2006), in which the tests in the nodes of the trees are defined with an extreme degree of randomness. ERT-based schemes have shown to be appropriate to compute RF distances (Bicego and Cicalese 2023), also in tasks in which labels are available (Ting et al. 2018). Given the trained ERCIF, the distance between two time series is defined by simply exploiting one of the schemes already defined for the vectorial case. For example, following the definition of Breiman Breiman (2001), the distance is computed by letting the two time series traverse all the trees of the ERCIF, and counting how many times they fall in different leaves. The main intuition is that if two time series are clearly different, they will answer the tests in the nodes of the trees in different ways, thus ending in different leaves. In this paper, we investigated three different distance schemes: the classic Breiman scheme, the more recent scheme introduced in Zhu et al. (2014), which exploits path overlaps, and the very recent scheme (Bicego et al. 2023), which is based on the Tversky ratio model.

According to the scenario depicted, the main novelty of the proposed TSRF-Dist may be summarized as in the following:

- TSRF-Dist exploits the ERT paradigm, shown to exhibit good theoretical properties for the derivation of RF distances (Bicego and Cicalese 2023), to provide an RF-based distance for time series.
- TSRF-Dist leverages on ERCIF, which is unsupervised, while the other models proposed in the literature are supervised, being designed for classification. Thus, ERCIF can be trained without labels and, thus, it can also be exploited in unsupervised contexts such as time series clustering or anomaly detection.

Due to its unsupervised nature, the proposed distance has been thoroughly investigated in the clustering scenario with the 128 datasets from the `UCR Time Series` archive (Dau et al. 2019), testing different variants and different parametrizations with an average-link Agglomerative Clustering algorithm. We also compared our distance to different literature alternatives, employing all those measures investigated in the recent review (Holder et al. 2024) on clustering. The obtained results are promising, showing that our TSRF-Dist can represent a valid alternative to classic as well as advanced distances between time series. We also developed a highly optimized `Rust` implementation, which will be made freely available to the community.

The rest of the paper is organized as follows: in Sect. 2, we summarize the related work, whereas the proposed distance is introduced in Sect. 3. Section 4 describes the empirical evaluation, and, finally, in Sect. 5, we draw some conclusions, and envisage future perspectives.

2 Related work

Measuring the distance between time series represents a crucial step in Pattern Recognition and Data Mining, often being the basis for other tasks such as classification, clustering, and others (Holder et al. 2024; Wang et al. 2013; Esling

and Agon 2012). Many measures have been proposed in the past: excellent surveys and comparisons can be found in the literature like the very recent review for clustering (Holder et al. 2024) or the one for classification (Abanda et al. 2019). In this section, we briefly summarize the time series distances to which our method is compared, as discussed in Sect. 4.4, most of them being also considered recently in Holder et al. (2024).

The first distance metric that often comes to mind when analyzing quantitative data is the Euclidean distance. However, in the context of time-series analysis, this metric has demonstrated sub-optimal performance as a benchmark (Middlehurst et al. 2021). Notably, it requires the two time series to have the same length, making it primarily suitable for use in benchmarks like the `UCR Time Series` archive, which contains time series of fixed length.

A simple variant we considered (called by us CATCH22Eucl)—both to address the fixed-length constraint and to evaluate the claim that improvements stem not from the use of random trees but from CATCH22–was to compute the complete set of CATCH22 features on the normalized time series and then calculate the Euclidean distance between them.

Dynamic Time Warping (DTW) is arguably the most extensively researched distance measure for time-series analysis (Ratanamahatana and Keogh 2005). It has led to the development of numerous variants (Herrmann and Webb 2023), each aiming to address specific limitations of the original formulation (i.e. CDTW, where the alignment is constrained to a specified length), such as its non-metric properties (Jain 2019). In essence, DTW calculates the optimal alignment between two time series by allowing for the repetition of points within each series; such operation is called "warping". The optimal solution is determined through dynamic programming, with the value of the distance represented by the cost of the solution.

As previously mentioned, many extensions of DTW have been proposed over the years. For example, in Derivative Dynamic Time Warping (DDTW) (Keogh and Pazzani 2001), each point in the compared time series is transformed into the mean of the slopes with its adjacent points, both preceding and following. Subsequently, the transformed series undergoes DTW alignment. Another interesting extension is the Weighted Dynamic Time Warping (WDTW) (Jeong et al. 2011), which, unlike DDTW, modifies the DTW framework by penalizing long warping paths instead of altering the time series itself. By integrating a penalty function that scales with the length of the path, WDTW aims to mitigate the generation of unnecessarily long warping paths by the DTW algorithm. The severity of the penalty is adjustable, controlled by a parameter that determines its growth with increasing path length. Combining the advantages of DDTW and WDTW, Weighted Derivative Dynamic Time Warping (WDDTW) applies the penalty function to the slope-transformed series, enhancing both alignment accuracy and path length control. Finally, the very recent Amercing DTW (ADTW) introduces a novel approach that further refines the alignment process by adding a fixed additive cost, aiming to improve accuracy in diverse application scenarios (Herrmann and Webb 2023). This method applies a constant penalty to each warping

operation, regardless of the warping distance, thereby balancing the trade-off between warping and non-warping alignments.

Another very famous measure is the Longest Common Subsequence (LCSS) (Górecki 2014), which represents an adaptation of the method for identifying the maximal sequence of sequential (though not necessarily contiguous) elements shared between two discrete sequences of symbols. In the discrete case, this objective is accomplished by performing deletion operations on both sequences until they match exactly. In such a case, the distance is quantified by the number of deletions required. For application to time series data, LCSS is adapted by softening the strict requirement for exact symbol matches to a criterion based on the proximity of real-valued elements. This adaptation involves the introduction of a parameter, namely ϵ, which sets the threshold for deeming two real numbers sufficiently similar. Deletions are then executed until both series are of equal length and corresponding elements across these series fall within the ϵ threshold of each other. Analogous to Dynamic Time Warping (DTW), the LCSS problem for time series can also be efficiently resolved using dynamic programming techniques.

A further widely applied set of distances for time series is based on Edit Distance. For example, the Edit Distance on Real Sequences (EDR) (Chen et al. 2005) permits not only deletions but also insertions of elements, and it allows for the (penalized) matching of real values whose difference exceeds a specified threshold ϵ. Consequently, EDR can produce both an alignment and a common subsequence, in contrast to LCSS, which primarily focuses on finding a common subsequence. Another example is the Edit Distance with Real Penalties (ERP) (Chen and Ng 2004), which may be seen as the gap-penalizing variant of EDR, similar to what WDTW represents within DTW frameworks. ERP employs a penalty function triggered by the formation of gaps, which may originate from either insertions or deletions in sequence alignments. In line with the penalty logic of WDTW, the severity of the penalties adopted by ERP correlates with the size of the gap, thus applying a stronger deterrent against the creation of larger gaps. The Moving Split Merge (MSM) distance (Stefan et al. 2013) advances the ERP framework by introducing a refined penalty mechanism for handling gaps with greater precision. In MSM, the treatment of insertions (splits) and deletions (merges) is context-dependent. A constant penalty is applied to splits if the inserted value, presumed to be a copy of the current existing value in the time series, is encapsulated by two adjacent values in the comparison time series. Should this inserted value not satisfy such a condition, its penalty is calibrated based on its distance from the farther adjacent value. Merges operate under a similar principle, where a constant penalty is applied if a deleted value is with the two corresponding consecutive values in the comparison time series, with the penalty intensifying exactly as the split case if the value does not satisfy such a condition.

Finally, Time Warp Edit (TWE) seamlessly combines the edit distance feature prevalent in EDR frameworks with the warping capabilities central to Dynamic Time Warping DTW methodologies (Marteau 2009). TWE imposes a warping penalty related to the length of the warping path during match operations, ensuring that the cost reflects the extent of the warping path needed for alignment. Conversely,

when alignment necessitates insertions or deletions, TWE enforces edit penalties for these operations, following a mechanism similar to that found in ERP.

Within the clustering scenario, the context of our experimental evaluation, K-Shape is a notable time series clustering algorithm that leverages a specific distance measure called the Shape-Based Distance (SBD) (Paparrizos and Gravano 2015). K-Shape aligns time series by their shapes and clusters them based on their similarity. The SBD measure is particularly effective for capturing the shape characteristics of time series, making K-Shape a powerful tool for clustering tasks where shape similarity is crucial.

In addition to these approaches, we also evaluated MPDIST, a distance measure designed specifically for time-series data. Unlike Euclidean distance, MPDIST is robust to differences in time-series length and accounts for local variations, making it particularly well-suited for real-world applications where time-series alignment or normalization may vary. Its ability to capture similarity across subsequences without requiring strict global alignment provided a complementary perspective to our analysis.

3 The proposed distance

In this section, we propose the distance TSRF-Dist. As typically done in the field of vectorial Random Forest distances, TSRF-Dist is obtained by following two steps:

1. In the first step, a Random Forest for time series is built, starting from a set of training time series. Since no labels are available, classical classification Forests for time series as those in Deng et al. (2013), Middlehurst et al. (2020) cannot be used. We introduce here a novel RF, which we call *Extremely Randomized Canonical Interval Forest* (ERCIF), which i) can model time series, and ii) can be trained without labels.
2. In the second step, given two time series, the TSRF-Dist is defined through the learned ERCIF, in particular by exploiting the information obtained from the paths the two time series are taking in each tree of the forest.

3.1 Step 1: The extremely randomized canonical interval tree (ERCIT) and forest (ERCIF)

The Extremely Randomized Canonical Interval Forest (ERCIF) represents an ensemble of Extremely Randomized Canonical Interval Trees (ERCITs). An ERCIT is a proper binary tree where each node has exactly zero—leaf—or two—internal nodes—children nodes. Each internal node is equipped with a test. According to the test result, a time series is sent to the left or to the right child. More formally, a test t on a time series $\mathbf{x} = x_1, \ldots, x_m$ is defined by a tuple $t = (f, v, i)$ which works as follows:

- It extracts from the time series $\mathbf{x}$ the values corresponding to the interval defined by i. If we assume that all the time series have the same length, we can define i in terms of absolute positions: $i = [i_l, i_u]$, where i_l marks the start of the interval, and i_u the end. If we do not assume that time series have fixed length, we can define the interval in terms of a start and percentages on the length of the time series (e.g., from 20 to 25% of the time series). Let us denote as $\mathbf{x}^{(i)}$ the extracted interval.
- It measures from $\mathbf{x}^{(i)}$ the feature f, which represents one of the possible measures we have defined (see below). Let us denote as $\mathbf{x}_f^{(i)}$ the computed feature.
- If $\mathbf{x}_f^{(i)} < v$ the time series $\mathbf{x}$ follows the left branch, otherwise it is sent to the right branch.

Features Feature f is selected among a set of 25 different measures computable from a time series: in particular, for the first 22 we employ the CATCH22 features, which represent a well-known collection of 22 descriptive features designed for time series analysis (Lubba et al. 2019), aiming at providing a concise and informative subset of the 7658 features in the HCTSA toolbox (Fulcher and Jones 2017). For the sake of completeness, we provide a brief summary of the CATCH22 features in Table 4. This is in line with what is done in the *Canonical Interval Forest* (Middlehurst et al. 2020), one of the most recent RF for the classification of time series. The last three features represent the mean, the standard deviation, and the slope of the least squares regression line, as defined in the *Time Series Forest* Classifier of Deng et al. Deng et al. (2013).

Training of an ERCIT We follow a classical top-down, recursive strategy to build an Extremely Randomized Canonical Interval Tree (ERCIT). We start with the whole training set $D = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ composed by n time series, and choose a test $t = (f, v, i)$ for the root. The test splits D into D_L and D_R, according to the answer the time series in D give to it; sets D_L and D_R are then used to determine the tests in the left and right children of the root, going ahead with the recursion until the tree reaches a height limit or a node contains a minimum number of samples.

To determine the test to be adopted in a given node, we resort here to the paradigm employed within the Extremely Randomized Trees (ERT-Geurts et al. 2006): instead of searching for the "best" question [like the Gini Criterion in classification trees (Breiman et al. 1984)], within the ERT paradigm, the tests are chosen in an extremely randomized way. This paradigm has been shown to be surprisingly useful for classification (Geurts et al. 2006), anomaly detection (Liu et al. 2012), clustering (Bicego and Escolano 2020; Moosmann et al. 2006; Shotton et al. 2008), and crucially, for the derivation of vectorial RF distances (Ting et al. 2016, 2019; Aryal et al. 2014, 2017, 2020). It represents an unsupervised, simple, and efficient way to derive a forest, able to define distances that often outperform more sophisticated schemes, even in supervised cases, as shown in Breiman (2000), citedavies2014random. A theoretical characterization of the good behaviour of ERT for RF distances has also been recently shown in Bicego and Cicalese (2023). More in detail, in our ERCIT, given a set $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of

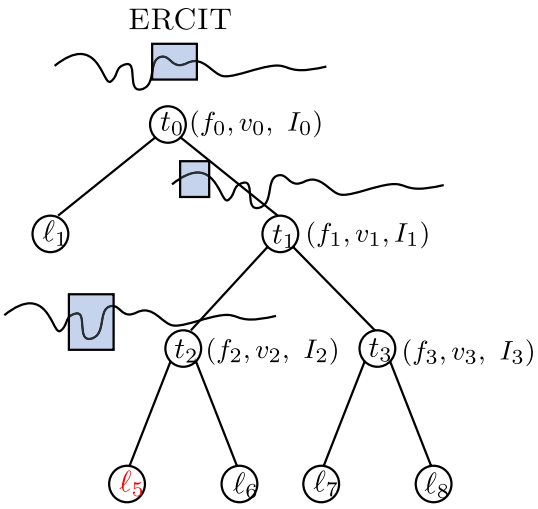

Fig. 1 An Extremely Randomized Canonical Interval Tree. In each node i, the decision is taken as follows: (i) we extract the interval I_i from the time series (highlighted by the shaded box); (ii) on I_i we compute the feature f_i; (iii) if $f_i < v_i$, the time series is sent to the left else to the right, till we reach a leaf (in red)

time series reaching a node, the test $t = (f, v, i)$ of a node is determined in the following way:

- Interval i is defined by selecting an interval from the set I. The set I is computed by randomly choosing starting points and random lengths.[1]
- Feature f is randomly selected from a random subset of 25 pre-defined features.
- A split threshold v in the range $(f_{min} - f_{max})$ is randomly selected, which represents the domain of feature f within the objects at that node, namely:

$$f_{min} = \min_{j=1\cdots n} \mathbf{x}_{jf}^{(i)}, \quad f_{max} = \max_{j=1,\cdots n} \mathbf{x}_{jf}^{(i)},$$

where $\mathbf{x}_{jf}^{(i)}$ represents the value of feature f computed on the interval i extracted from the time series $\mathbf{x}_j$.

Figure 1 illustrates the tree-building process for time series, where at each node, a random interval and features are selected. Based on the time series' response to the test, the sample goes to the left or right.

Algorithm 1 contains the pseudo-code of the complete training procedure for the ERCIT.

[1] Please note that to compute some of the CATCH22 features, the interval should have a minimal length of 20.

Algorithm 1 ERCIT Fitting: buildERCIT

Data: Training Set D, CATCH22 Feature Subset F, Interval Subset I
Result: Fitted Tree.
if $|D| < 2$ **then**
  return leaf{size : $|D|$}
end
else
  $D_L \leftarrow \emptyset$
  $D_R \leftarrow \emptyset$
  $i \in_R I$ `/* select a random interval from` I `*/`
  $f \in_R F$ `/* select a random feature from` F `*/`
  Let $D_f^{(i)}$ be the set composed by computing $\mathbf{x}_f^{(i)}$ for every $\mathbf{x} \in D$
  $v \in_R D_f^{(i)}$ `/* Randomly select` v `from` $D_f^{(i)}$ `*/`
  `/* Test Split */`
  for $j \leftarrow 0..|D|$ **do**
    if $\mathbf{x}_{jf}^{(i)} < v$ **then**
      $D_L \leftarrow D_L \cup \{\mathbf{x}_j\}$
    end
    else
      $D_R \leftarrow D_R \cup \{\mathbf{x}_j\}$
    end
  end
  return node$\left\{ \begin{array}{l} left : buildERCIT(D_L), \\ right : buildERCIT(D_R), \\ split_feature : f, split_value : v, interval : i \end{array} \right\}$
end

From Trees to Forest The *Extremely Randomized Canonical Interval Forest* (ERCIF -see Algorithm 2) represents an ensemble of ERCITs. Following the classic recipe of Random Forests (Breiman 2001), each ERCIT is created, with the procedure described previously, starting from a random bootstrapping of the training set.

To control the computational requirements for the training of ERCIT, while keeping a high level of randomization, we adapt to our case some of the mechanisms typically employed for classification trees. For example, in a CART (Classification and Regression Tree) (Breiman et al. 1984), it is common that the optimal feature in a node is selected among only a subset of all possible features, which is randomly chosen and kept fixed over the whole tree construction. In our case, at the beginning of the ERCIT training, we randomly select a set of n_i intervals and a set of n_f features; then, during the recursive training, feature f and interval I in each test $t = (f, v, I)$ are randomly chosen within these pre-selected sets. By efficiently pre-computing all the selected features on all the selected intervals for all the training time series, we reduce the overload of the recursive training procedure. At the same time, since this procedure is repeated for every tree in the forest, different trees work on different intervals and use different features, maintaining a high degree of randomization,

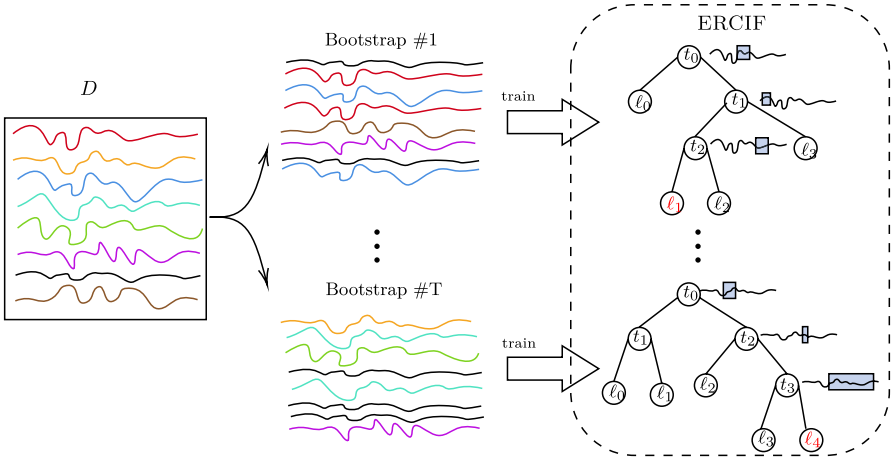

Fig. 2 Training phase of the ERCIF model. The original unlabelled dataset is bootstrapped T times, and each bootstrap sample is used to train a single ERCIT

which is typical in ERT-inspired schemes. Figure 2 offers a visual depiction of the training procedure for the presented forest-based model.

Let us conclude the section by stressing the novelty of the proposed ERCIF. While sharing some of the mechanisms with the other Random Forests for time series proposed in the literature, like the computation of features from subparts of the time series, as done in Middlehurst et al. (2020), Deng et al. (2013), Karlsson et al. (2016), ERCIF has two distinctive traits: (i) differently than the other models, which are designed for classification, it is completely unsupervised, since it can be trained without labels—this opens the possibility to exploit ERCIF also in unsupervised contexts such as time series clustering or anomaly detection; (ii) the model is used to define an RF distance, and exploits the ERT paradigm, which was shown to exhibit good theoretical properties for derivation of RF distances (Bicego and Cicalese 2023).

Algorithm 2 ERCIF Construction: buildERCIF

Data: Training Set D, Number of trees n_t, Set of features F.
Result: Fitted Forest.
$Forest \leftarrow []$
for $i \leftarrow 0$ **to** n_t **do**
  $D_i \leftarrow bootstrap(D)$
  `/* generate random intervals set */`
  $I_i \leftarrow sample\_intervals(D)$
  $Forest[i] \leftarrow build\text{ERCIT}(D_i, F, I_i)$
end
return *Forest*

3.2 Step 2. Computing the distance

Given the trained ERCIF and two time series, the TSRF-Dist is then computed following the typical schemes used for vectorial RF distances (Shi and Horvath 2006; Zhu et al. 2014; Ting et al. 2016; Aryal et al. 2020; Ting et al. 2019; Aryal et al. 2014, 2017; Bicego et al. 2023). The main idea, which is also behind all RF distances, is the following. In an ERCIT, a node contains a test on a specific feature extracted from a specific interval of the time series. If the two time series provide the same answer to this test, they share a similarity in that feature and interval; otherwise, if they respond differently, they do not. This can be generalized at the tree level since a time series follows a path in the tree, i.e., it has to answer many tests. By aggregating and comparing these answers, we can define a similarity measure in a tree, which can finally be aggregated at the forest level (graphical example in Fig. 3).

For example, in his seminal work (Breiman 2001), Breiman defined a tree-based similarity measure that compares two objects: if the objects fall in different leaves, their similarity is 0; if they fall in the same leaf, their similarity is 1. This tree-based measure is then extended to a forest-based similarity by averaging the 0/1 similarities across all trees in the forest. As a result, the forest-based similarity between two objects is computed as the proportion of trees in which the two

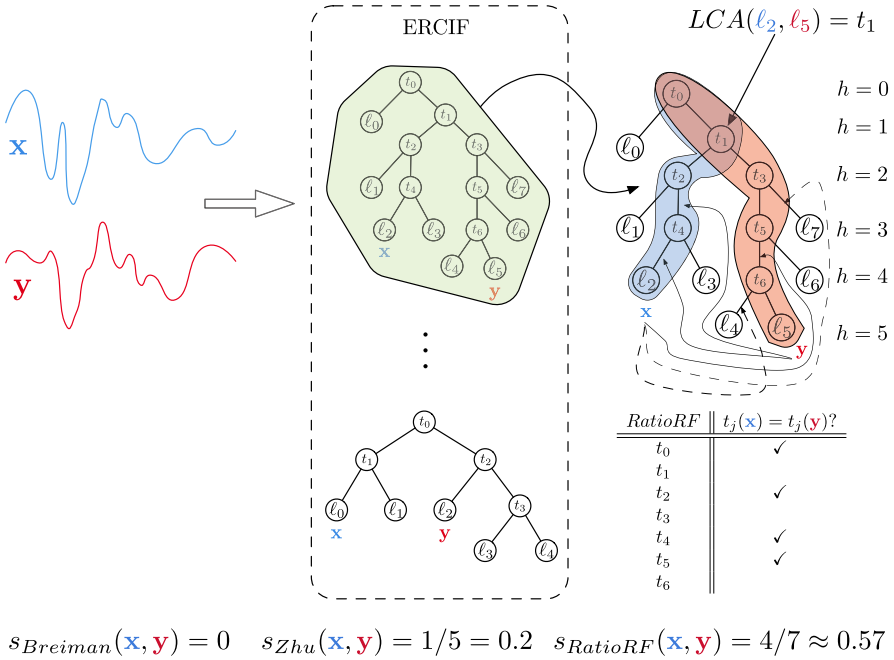

Fig. 3 Visual representation of the computation of the TSRF-Dist in Breiman, Zhu, and RatioRF versions

objects end up in the same leaf, effectively measuring how often they provide identical answers to all tests along their paths through the trees.

Here we consider three different distance measures: in all cases, given two time series $\mathbf{x}$, $\mathbf{y}$, we (i) define a tree-similarity $s(\mathbf{x}, \mathbf{y})$ between them; (ii) aggregate such similarity at the forest level by averaging over the different tree similarities; (iii) convert the similarity into a distance by subtracting this similarity value from 1. More precisely, given an ERCIF composed by T ERCITs, and given two time series $\mathbf{x}$, $\mathbf{y}$, the TSRF-Dist $d^{TS\text{-}RF}(\mathbf{x}, \mathbf{y})$ is defined as:

$$d^{TS\text{-}RF}(\mathbf{x}, \mathbf{y}) = 1 - \frac{1}{T} \sum_{i=1}^{T} s^i(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $s^i(\mathbf{x}, \mathbf{y})$ corresponds to the chosen similarity measure, evaluated on the $i-$th ERCIT. Thus, the resulting distance measure varies according to the chosen definition of tree similarity. In this paper, we consider three distinct similarity measures, each entailing a different distance measure:

- The first measure adopts the variant introduced by Breiman (2001), described above. The corresponding *Breiman* tree similarity $s(\mathbf{x}, \mathbf{y})$ with respect to a given ERCIT is defined as:

$$s_{Breiman}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 \ if \quad \ell(\mathbf{x}) = \ell(\mathbf{y}), \\ 0 \ otherwise. \end{cases}, \tag{2}$$

  where $\ell(\mathbf{x})$ and $\ell(\mathbf{y})$ represent the leaves, in the considered ERCIT, where the time series $\mathbf{x}$ and $\ell(\mathbf{y})$ fall, respectively.
- The second measure adopts the ClustRF-Strct definition proposed in Zhu et al. (2014), here simply referred to as *Zhu* from the name of the first author. The *Zhu* measure is defined as the depth of the lowest common ancestor (LCA) of the two leaves where the two time series fall: the deeper this node, the larger the number of tests to which the two time series answered in the same way. Formally,

$$s_{Zhu}(\mathbf{x}, \mathbf{y}) = \frac{h(LCA(\ell(\mathbf{x}), \ell(\mathbf{y})))}{max(h(\ell(\mathbf{x})), h(\ell(\mathbf{y})))}, \tag{3}$$

  where $h(u)$ represents the depth of a node u, and $LCA(u, v)$ represents the lowest common ancestor between nodes u and v.
- The third measure employs the *RatioRF* definition, the most recent in this field (Bicego et al. 2023), which has shown to outperform many alternatives in the clustering scenario. *RatioRF* defines a similarity that follows the axiomatic definition given by the Tversky's Ratio model (Tversky 1977). The main idea is to consider, among all the tests in the paths of the two time series $\mathbf{x}$ and $\mathbf{y}$, how many times $\mathbf{x}$ and $\mathbf{y}$ provide the same answer. Formally, if we define the set $P(\mathbf{x})$ of tests in the path the time series $\mathbf{x}$ is following in the given ERCIT and denoting as $t(\mathbf{x})$ the answer provided by the time series $\mathbf{x}$ to a given test t (and similarly for time series $\mathbf{y}$), the RatioRF similarity is defined as

$$s_{RatioRF}(\mathbf{x}, \mathbf{y}) = \frac{|\{t \in P(\mathbf{x}) \cup P(\mathbf{y}) \mid t(\mathbf{x}) = t(\mathbf{y})\}|}{|\{P(\mathbf{x}) \cup P(\mathbf{y})\}|}, \tag{4}$$

We conclude this section with an example to illustrate the computation of the TSRF-Dist distances. Figure 3 demonstrates how these distances are computed. In this process, two time series are fed to each ERCIT of the trained ERCIF, and their root-to-leaf paths are compared. As shown in the right part of Fig. 3, considering the ERCIT in the green area, the blue time series $\mathbf{x}$ reaches ℓ_2 at depth 4, while the red time series $\mathbf{y}$ terminates in leaf ℓ_5. The node t_1 serves as their Lowest Common Ancestor (LCA) at depth 5. Based on these paths, we can compute the three distances. The *Breiman* distance, being a binary measure, simply checks if the two time series end up in the same leaf node. In this case, since $\mathbf{x}$ and $\mathbf{y}$ land in different leaves (ℓ_2 and ℓ_5 respectively), the *Breiman* similarity for this tree, $s_{Breiman}(\mathbf{x}, \mathbf{y})$, is 0.

The *Zhu* similarity, on the other hand, considers the depth of the *LCA* relative to the maximum depth reached by either time series. Here, the *LCA* (t_1) is at depth 1, while the maximum depth between ℓ_2 and ℓ_5 is 5. Thus, the Zhu similarity is quantified as $s_{Zhu}(\mathbf{x}, \mathbf{y}) = 1/5 = 0.2$, representing the ratio of the *LCA* depth to the maximum depth between the leaves determined by the two samples. Lastly, the *RatioRF* similarity examines the proportion of common decisions along the paths of both time series. In our example, we assume that paths of $\mathbf{x}$ and $\mathbf{y}$ agree on 4 decisions (at nodes t_0, t_2, t_4, and t_5, as reported in the table in Fig. 3), out of a total of 7 decisions encountered. This results in a *RatioRF* similarity of $s_{RatioRF}(\mathbf{x}, \mathbf{y}) = 4/7 \approx 0.57$. To compute the final TSRF-Dist measure, each of these three similarities would be averaged across all trees in the forest.

4 Experimental evaluation

This Section deals with the evaluation of the proposed distances. In particular, after introducing in Sect. 4.1 our benchmark dataset, the `UCR Time Series` archive (Dau et al. 2019), we present the experimental setup in Sect. 4.2. We then compare the different variants of TSRF-Dist in Sect. 4.3; we compare our approach with literature alternatives in Sect. 4.4, closing with Sect. 4.5, where we provide some insights on the implementation details, specifically on the performance related to the execution time.

4.1 Benchmark dataset: the UCR time series archive

The `UCR Time Series` archive is a comprehensive collection of time series data that served as the foundation for a thorough evaluation of our proposed TSRF-Dist method. Our evaluation encompasses 128 datasets from the `UCR Time Series` archive (Dau et al. 2019). To provide a high-level overview of the types of data collected in the `UCR Time Series` archive, we categorized its datasets into three

groups: Multimedia, Sensors/Devices, and Biomedical & Others. Table 1 provides a summary, detailing the number of datasets and associated UCR Time Series tags for each category.

Being each UCR Time Series dataset originally organized as a supervised dataset, it is provided pre-divided into training and test files. However, since our focus is on the unsupervised task of clustering, we merged these files into a single one for each UCR Time Series dataset. The original labels have been retained and used as external criteria for evaluating clustering performance, which we discuss further in Sect. 4.2.

After merging, each dataset is characterized by two key parameters: the number of samples and the length of the time series. Our assessment focuses on two key aspects: effectiveness, measured by the clustering performance, and efficiency, measured by the computational performance.

Figure 4 presents a scatter plot of the parameters, i.e., the number of samples and the length of the time series, for all datasets in the UCR Time Series archive. In this plot, both blue dots and red stars represent datasets. Experiments on clustering performance (Sects. 4.2, 4.3, and 4.4) have been conducted on all datasets, i.e., corresponding to both dots and stars, represented in Fig. 4. However, for our computing-time benchmark experiments (Sect. 4.5), we used a selected subset of datasets, depicted as red stars in Fig. 4. This selection ensures coverage of various combinations of time series length and number of samples, providing a comprehensive view of the computational performance of our method across different configurations, each one identified by specific (number of samples, length) pairs.

For a more comprehensive overview of the datasets, including specific categories, names, and cardinalities, readers can refer to https://timeseriesclassification.com/dataset.php.

4.2 Experimental details

The experimental evaluation has been conducted in comparison with established time series distances from the literature, providing a comprehensive view of the main features of TSRF-Dist. To compare TSRF-Dist with existing time series distances from the literature on the UCR Time Series datasets, we implemented the experimental workflow presented in Fig. 5. As commonly done with time series distances (Holder et al. 2024), the evaluation is performed in a clustering setting. We employed Agglomerative Clustering (Ran et al. 2023), using average linkage as the linkage method, which starts with each object forming its

Table 1 Details of each category

Category	#Datasets	UCR Tags
Multimedia	34	Image, Audio
Sensors/Devices	43	Spectro, Sensor, Device, Traffic
BioMed & Others	51	ECG, Hemodynamics, EOG, EPG, Simulated, Other, HAR, Motion

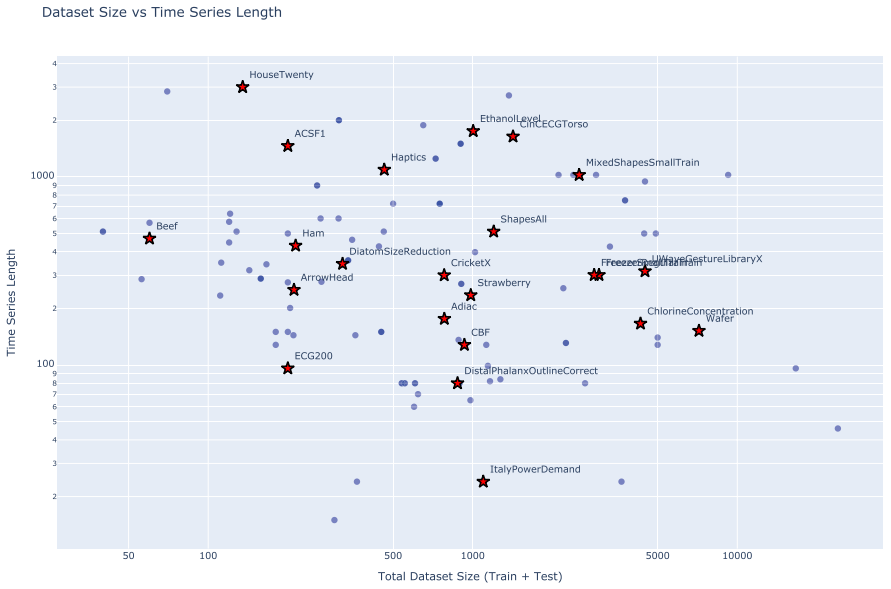

Fig. 4 Scatter plot of dataset size vs time series length for the datasets in the `UCR Time Series` archive. The x-axis shows the total dataset size (train + test) on a logarithmic scale, and the y-axis shows the time series length on a logarithmic scale

own cluster and then iteratively merges clusters based on their similarity to form larger clusters. In our case the process terminates when the number of clusters equals the number of classes of the problem. As the final step in our evaluation process, we compute the *Adjusted Rand Index* (ARI) (Hubert and Arabie 1985). ARI is a corrected-for-chance version of the Rand index that compares two partitions of a set. It is commonly used in two scenarios: (1) to measure the agreement between two different clusterings of the same dataset and (2) to assess the similarity between clustering results and known ground truth labels. In our study, we focus on the latter application. ARI ranges from -1 to 1, where 1 indicates perfect agreement between the partitions, 0 represents random labelling, and negative values indicate less agreement than expected by chance. Specifically, in our case, we use ARI to quantitatively assess how well our clustering results match the known class structure of each dataset. Figure 5 depicts the formula for computing the ARI, as well as a graphical description of our entire experimental process, including the clustering and evaluation steps. This approach allows us to account for the possibility of agreement occurring by chance while providing a robust metric for comparing the performance of different time series distance measures across `UCR Time Series` datasets. Our experimental setup involves two distinct configurations of the ERCIF model: **H**eavy and **L**ight. The **H**eavy configuration closely resembles that employed in the paper introducing the Canonical Interval Forest (Middlehurst et al. 2020), while the **L**ight configuration explores the possibility of reducing computational

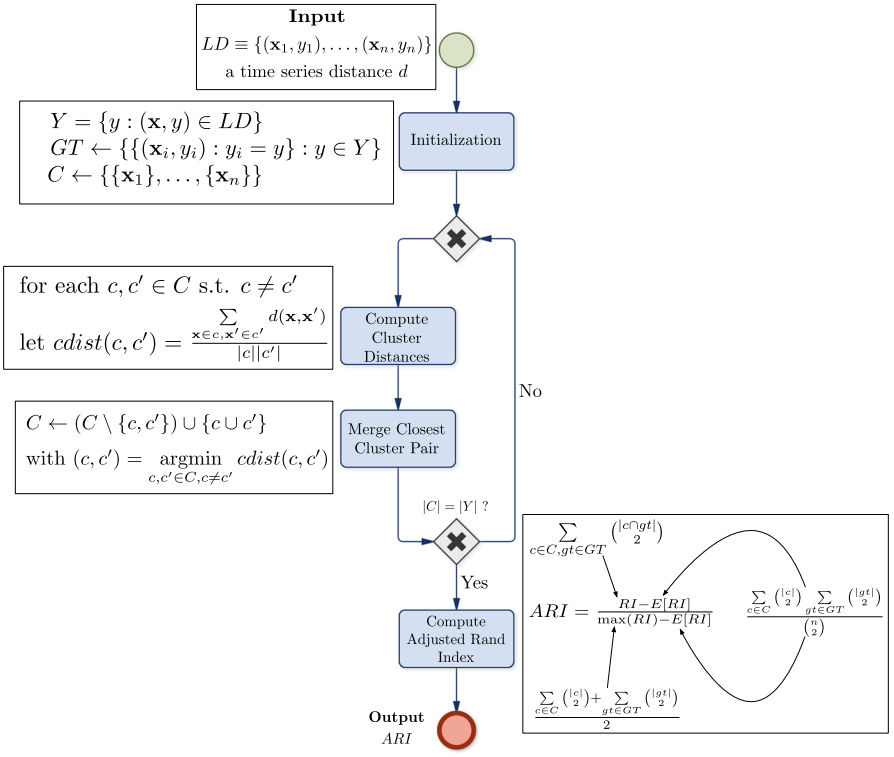

Fig. 5 Our experimental workflow for evaluating clustering by a given time series distance d against a dataset with a given external criteria

requirements. In particular, in the **H**eavy we set the number of ERCITs $T = 500$, the number of features $n_f = 8$ and the number of intervals $n_i = \sqrt{N}$, where N is the sequence length. In the **L**ight configuration, we set $T = 200$, $n_f = 8$ and $n_i = \log_2 N$. For both configurations, we computed ARIs over datasets with the three distances described in Sect. 3, namely *Breiman*, *Zhu*, and *RatioRF* distances, for a total of six variants of TSRF-Dist.

To assess the impact of randomization on the performance of the ERCIF model, we iterated the entire workflow described in Fig. 5 ten times for each dataset and variant. This repetition allows us to measure how the inherent randomization affects the performance of our proposed method. We then report the average ARI across these iterations, providing a more robust evaluation of the performance of the model. Additionally, we applied the same workflow to all datasets in the `UCR Time Series` archive using the time series distances introduced in Sect. 2. Due to the deterministic nature of these distances, a single iteration of the workflow was sufficient for each dataset. The results of these comparisons between TSRF-Dist and the time series distances introduced in Sect. 2 are presented and analyzed in Sect. 4.4.

Table 2 ARI-based comparison between the Light (L) and Heavy (H) forest configuration. The third column represents the *p*-value of a Wilcoxon signed-rank test, between L and H

	L	H	*p*-value
Multimedia	0.3022	0.3072	0.0147
Sensors/Devices	0.2113	0.2084	0.4328
BioMed & Others	0.2608	0.2653	0.0009

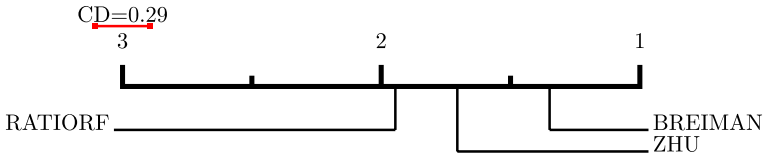

Fig. 6 Critical difference diagram comparing the ARIs of distances employed over the UCR archive datasets

4.3 Comparison between different variants

As a first analysis, we performed a comparison of the different variants of TSRF-Dist. First, we analyze the difference between the two configurations **L**ight and **H**eavy. In particular, for each dataset, we computed the average among *Breiman*, *Zhu*, and *RatioRF* of the ARI of the two configurations. The results, averaged over the datasets of the different domains, are presented in Table 2. The last column indicates the p-value of a Wilcoxon signed-rank test (Wilcoxon 1945), used to check whether the null hypothesis that two related paired samples are drawn from the same distribution can be rejected. For all statistical tests, the significance level has been set to 0.05.

As evident from the table, the **H**eavy configuration performs slightly better than the **L**ight one across all the categories. The statistical significance of these differences indicates that the improved performance of the **H**eavy configuration is due to chance for Sensors/Devices. On the other hand, the BioMed. & Others and Multimedia categories do exhibit this statistical difference, but it is also worth noting that the differences in the mean ARI are quite limited. Despite its higher computational demands, the superior outcomes of the **H**eavy configuration suggest that it is worth considering such configuration for applications where its enhanced performance is critical.

Our second analysis focuses on the various distances used, which is crucial for understanding performance variations across different scenarios. We computed the average ARI for the *Breiman*, *Zhu*, and *RatioRF* variants across the two ERCIF configurations and presented the results in a critical difference diagram (Fig. 6). To have an idea of the statistical significance of the comparison, we performed a Friedman test, followed by a post-hoc pairwise test for multiple comparisons of mean rank sums (Nemenyi's test) (Demšar 2006). The Friedman test checks if there are overall significant differences among the ranks of the analyzed methods in different datasets; if the test is passed, post-hoc Nemenyi mean rank sums assess pairwise differences. Finally, a critical difference diagram is derived, in which the mean ranks

across datasets represent the different methods. In all critical difference diagrams throughout this paper, rankings with a statistically significant difference are connected by red lines. Figure 6 clearly shows that the *Breiman* variant performs the best; additionally, the differences between the other distances are all statistically significant.

4.4 Comparison with state of the art

In this part, we compare the proposed approach with alternative literature distances developed for time series in the last years (hereinafter, we will call them also *elastic distances*, according to a widely accepted terminology). In particular, we used as competitors the distances described in Sect. 2, namely the Euclidean, CATCH22Eucl, ERP, LCSS, DTW, CDTW, DDTW, WDTW, WDDTW, ADTW, MSM, TWE, SBD and MPDIST distances. A subset of these distances have been considered in a recent review on clustering with time series distances (Holder et al. 2024)—we included the Euclidean distance as a baseline. In our experiments, for standard elastic distances, we used the implementation included in the `Aeon` Python library https://www.aeon-toolkit.org, which provides a comprehensive set of tools to perform effectively time series distances computation and comparison; moreover we used the average link Agglomerative clustering provided by `sklearn` [63]. Obtained ARIs, computed using the workflow depicted in Fig. 5 and averaged over the three different categories of Table 1, are displayed in Table 3, where we added to the last row the best result of our TSRF-Dist (with the Breiman variant and the best among **L**ight and **H**eavy configurations). As we can see in Table 3, the proposed TSRF-Dist outperforms, on average, over the three groups, all the competitors.

Table 3 Average ARIs comparison between TSRF-Dist and all other time series distances

	Multimedia	Sensors/devices	BioMed & others
EUCLIDEAN	0.1213	0.0848	0.1553
CATCH22EUCL	0.0463	0.0340	0.0619
ERP	0.1632	0.1326	0.2283
LCSS	0.1036	0.0771	0.1585
DTW	0.1722	0.0960	0.1982
CDTW	0.1802	0.0940	0.2075
DDTW	0.1172	0.0588	0.0444
WDTW	0.1803	0.1061	0.2176
WDDTW	0.1309	0.0454	0.0521
ADTW	0.1602	0.1018	0.2237
MSM	0.1848	0.0960	0.1842
TWE	0.1940	0.1006	0.1812
SBD	0.1258	0.0932	0.1365
MPDIST	0.0471	0.0492	0.0327
TSRF-Dist	0.3612	0.2711	0.3145

Figure 7 depicts the obtained critical diagram. The complete ARI results, including standard deviations, for TSRF-Dist and all other compared methods, from which the critical difference diagrams in Fig. 7 are derived, are reported in Table 5 in Appendix B.

Figure 7 is derived using the mean ARI over 10 distinct random initializations for TSRF-Dist, providing a direct comparison with other distance measures. The results are extremely promising. Indeed, we can observe that TSRF-Dist represents the method that ranks best on all the datasets used; moreover, the differences to ERP (the second most accurate method) are statistically significant, thus confirming that the proposed approach represents a valid alternative to the standard as well as advanced time series distances found in the literature.

4.5 Computational performance analysis

In this section, we provide details on the implementation and computational performance of TSRF-Dist. We developed a highly optimized implementation in `Rust`, leveraging parallelization to achieve efficient processing of time series data. Our implementation focuses on memory safety while maximizing computational speed through the effective use of multi-core architectures. For the sake of comparison, we benchmark the performance of TSRF-Dist against the computation of elastic distances provided by the well-known `Aeon` library (Middlehurst et al. 2024). This comparison allows us to assess the computational efficiency of our approach relative to established methods in the field. It is important to note that the reported benchmark times for TSRF-Dist on a given `UCR Time Series` dataset encompass the entire process, which consists of two main steps: (i) training the ERCIF over the whole dataset, and (ii) computing the pairwise *Breiman* distance using the trained ERCIF between all pairs of samples in the dataset. In contrast, for the elastic distances, the benchmark only includes the computation of pairwise distances between all time series in the given dataset, as these methods do not require a preliminary training step.

Leveraging the power of `Rayon`, a parallelization library for `Rust`, the TSRF-Dist achieves remarkable efficiency in processing time series data. Parallelization is done during forest construction, where each tree is built and processed

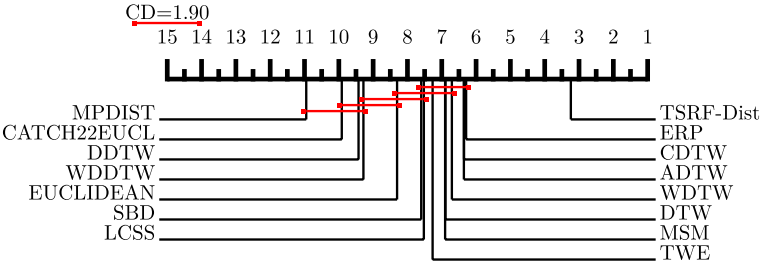

Fig. 7 Critical Difference Diagram comparing TSRF-Dist with other time series distances. Comparison using mean ARI values for TSRF-Dist

in a separate thread. This method significantly accelerates the fitting process while ensuring memory is used efficiently. Additionally, the calculation of distances benefits from implicit parallelization in this paradigm. Each pair of objects has their descent through the paths of a designated tree in the forest computed by a dedicated thread. The `Rust` language prioritizes memory safety, ensuring a robust and secure implementation, which is essential for managing real-world time series data. To illustrate how performance improves with the number of cores utilized, Fig. 8 presents a bar plot detailing the time required for constructing the forest under ERCIF light configuration parametrization and subsequently computing the distance matrices for *Breiman* distances. The plot compares these times (y-axis) against the number of cores employed (x-axis) on the `Adiac` dataset in the `UCR Time Series` archive.

As evident from the plot, the execution time of TSRF-Dist decreases significantly as the number of cores increases, demonstrating excellent scalability. The performance improvement is particularly pronounced up to 8 cores. Beyond 8 cores, we observe additional, albeit smaller, speedups. This scalability showcases the effectiveness of our parallelization strategy, enabling our implementation of TSRF-Dist to efficiently utilize modern multi-core processors and potentially handle larger datasets or more complex analyses within reasonable time frames.

The graphs in Fig. 9 show execution times for five elastic distances (ERP, WDTW, ADTW, DTW, MSM) and the two variants of the ERCIF-based Breiman distance, namely *Breiman_L* (Light configuration) and *Breiman_H* (Heavy configuration). Datasets are sorted by average execution time across all methods. The y-axis uses a logarithmic scale to accommodate the wide range of execution times. Note that *Breiman_L* and *Breiman_H* consistently show lower execution times compared to the elastic distances, particularly for computationally intensive datasets.

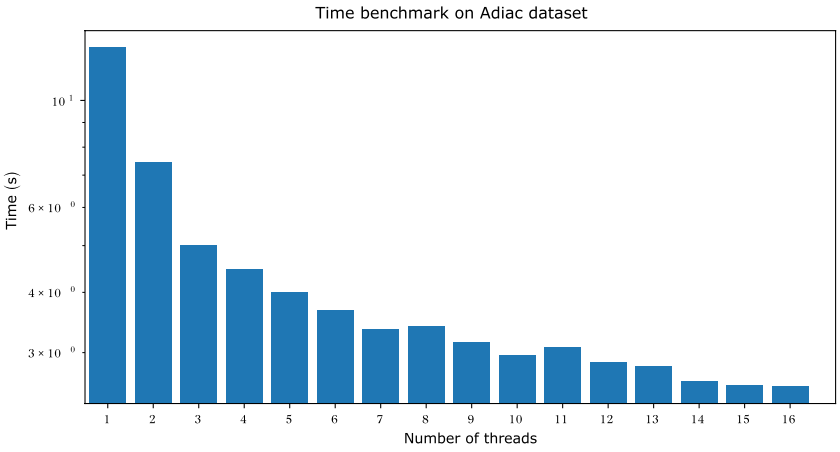

Fig. 8 Log-scale Bar plot showing the performances of the implementation with different numbers of cores (Intel(R) Core(TM) i9-10980HK)

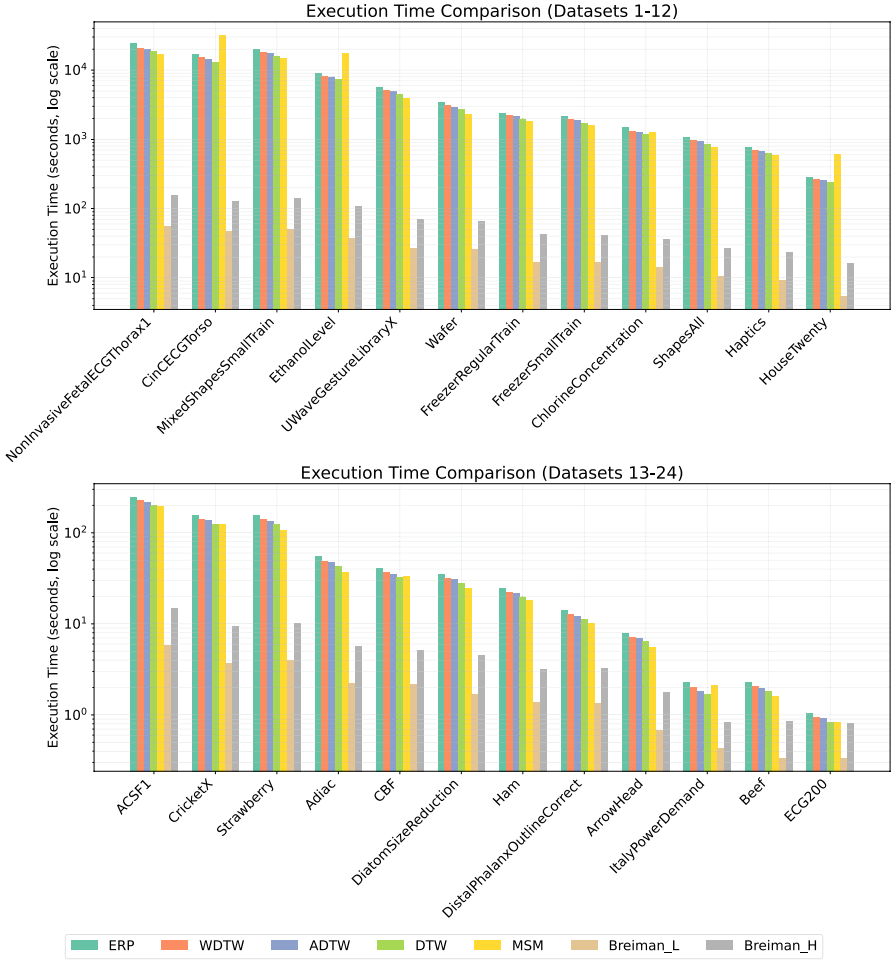

Fig. 9 Comparison of execution times for various distance measures across 24 time series datasets from `UCR Time Series` archive

For information about the time series length and the number of samples in each dataset, the reader should refer to Fig. 4.

Experiment Reproducibility The URL to the implementation is openly accessible at https://github.com/albertoazzari/forust. Additionally, the data employed in this study can be accessed via the `UCR Time Series` archive at https://www.cs.ucr.edu/%7Eeamonn/time_series_data_2018/, guaranteeing the full reproducibility of our results. Due to the Intel(R) Core(TM) i9-10980HK, which is composed of 8 cores and 16 threads, we can see from the plot in Fig 8 that the ERCIF scales linearly up to the eight cores and then can get minor speedups also from hyper-threading.

5 Discussion and conclusions

In this paper, we introduced a novel distance between time series based on Random Forests (RFs) called TSRF-Dist. The distance extends concepts and tools of vectorial RF distances to the time-series domain, thus representing the *first* RF distance for time series. To define the distance, we introduced a novel Random Forest, called *Extremely Randomized Canonical Interval Forest (*ERCIF*)*, that can model time series and can be trained without labels. On top of this, we derived three different distance variants. The proposed approach has been thoroughly evaluated with the 128 datasets in the `UCR Time Series` archive, showing promising results compared to literature alternatives.

CATCH22 summary

The Table 4 provides a simplified overview of the 22 features from the CATCH22 feature set, organized by category. Each feature is associated with specific statistical methods used to analyze time series data. The categories include: Distribution, Simple Temporal Statistics, Linear Autocorrelation, Nonlinear Autocorrelation, Successive Differences, Fluctuation Analysis and Others.

Datasets details and adjusted rand index (ARI) results

In this section, we report the details related to experiments performed on the `UCR Time Series` archive datasets for all the time series distances considered in this work. In Table 5, we report the ARI for these distances. For TSRF-Dist, we report the mean ARI with its corresponding standard deviation in parentheses.

Table 4 Simplified descriptions of the 22 CATCH22 features grouped by category (Lubba et al. 2019)

Category	Description
Distribution	• Mode of z-scored distribution (5-bin histogram)
	• Mode of z-scored distribution (10-bin histogram)
Simple temporal statistics	• Longest period of consecutive values above the mean
	• Time intervals between successive extreme events above the mean
	• Time intervals between successive extreme events below the mean
Linear autocorrelation	• First $1/e$ crossing of autocorrelation function
	• First minimum of autocorrelation function
	• Total power in lowest fifth of frequencies in the Fourier power spectrum
	• Centroid of the Fourier power spectrum
	• Mean error from a rolling 3-sample mean forecasting
Nonlinear autocorrelation	• Time-reversibility statistic, $\langle (x_{t+1} - x_t)^3 \rangle_t$
	• Automutual information, $m = 2$, $\tau = 5$
	• First minimum of the automutual information function
Successive differences	• Proportion of successive differences exceeding 0.04σ (Mietus 2002)
	• Longest period of successive incremental decreases
	• Shannon entropy of two successive letters in equiprobable 3-letter symbolization
	• Change in correlation length after iterative differencing
	• Exponential fit to successive distances in 2-d embedding space
Fluctuation analysis	• Proportion of slower timescale fluctuations that scales with DFA (Detrended Fluctuational Analysis) (50% sampling)
	• Proportion of slower timescale fluctuations that scales with linearly rescaled range fits
Others	• Trace of covariance of transition matrix between symbols in 3-letter alphabet
	• Periodicity measure, as defined in Wang et al. (2007)

Table 5 Complete ARIs results from our analysis of TSRF-Dist and the top 5 lature alternatives (with higher mean ARIs over all datasets), including the standard deviations

	TSRF-Dist	ERP	WDTW	ADTW	CDTW	TWE
ACSF1	0.4287 (0.0257)	0.0233	0.0051	0.0069	0.0099	0.0252
Adiac	0.4498 (0.0144)	0.0259	0.0795	0.0231	0.0797	0.0224
AllGWX	0.1253 (0.0109)	0.0137	0.0123	0.0131	0.0074	0.0033
AllGWY	0.0955 (0.0086)	0.0067	0.0119	0.0113	0.0087	0.0021
AllGWZ	0.1077 (0.0112)	0.0099	0.0113	0.0113	0.0085	0.0064
ArrowHead	0.2979 (0.0807)	− 0.0028	− 0.0037	− 0.0000	− 0.0031	0.0042
BME	0.2882 (0.0717)	0.1376	0.1366	0.1404	0.1404	0.1373
Beef	0.1414 (0.0146)	0.0629	0.0348	0.0658	0.0658	0.0305
BeetleFly	0.5299 (0.1949)	0.0052	− 0.0093	0.0722	0.0052	0.3438
BirdChicken	0.0506 (0.0665)	0.0000	− 0.0221	− 0.0150	0.0240	− 0.0221
CBF	0.4964 (0.2020)	0.1419	0.2769	0.7377	0.2288	0.6521
Car	0.1682 (0.0661)	0.1134	0.1058	0.1067	0.1061	0.1146
Chinatown	0.5628 (0.3547)	0.0456	0.0164	0.0164	0.0211	0.0456
ChlorineC	0.0131 (0.0179)	0.0193	0.0056	0.0092	0.0056	0.0101
CinCECGTorso	0.9978 (0.0093)	0.0237	0.2480	0.0116	0.4308	0.0003
Coffee	0.7272 (0.1067)	− 0.0087	0.0158	0.0026	0.0158	0.0026
Computers	0.1705 (0.0472)	0.0049	− 0.0004	− 0.0002	− 0.0005	0.0018
CricketX	0.1609 (0.0139)	0.1010	0.1166	0.1201	0.0857	0.0552
CricketY	0.1984 (0.0177)	0.1444	0.1204	0.1354	0.0781	0.1155
CricketZ	0.1620 (0.0195)	0.0656	0.1122	0.1134	0.1131	0.0746
Crop	0.2427 (0.0353)	0.2418	0.2105	0.2331	0.2291	0.2126
DiatomSR	1.0000 (0.0081)	0.0123	0.0123	0.0123	0.0190	0.0123
DistalPOAG	0.2130 (0.0000)	0.4220	0.4220	0.4185	0.4220	0.4169
DistalPOC	0.0086 (0.0276)	0.0014	0.0014	0.0027	0.0027	0.0014
DistalPhalanxTW	0.3274 (0.0001)	0.6733	0.5882	0.5938	0.6710	0.6610
DodgerLoopDay	0.2007 (0.0449)	0.1786	0.2033	− 0.0002	0.2023	0.2027
DodgerLoopGame	0.0046 (0.0183)	0.0047	0.0010	0.0010	− 0.0009	0.0010
DodgerLW	0.8508 (0.0041)	0.9229	− 0.0074	− 0.0074	− 0.0074	− 0.0074
ECG200	0.2767 (0.0604)	0.0242	0.0098	0.0242	0.0507	0.0098
ECG5000	0.4420 (0.0545)	0.7128	0.7291	0.8222	0.0177	0.0016
ECGFiveDays	0.4449 (0.0238)	0.0000	0.0138	0.0009	0.0033	0.0059
EOGHS	0.2675 (0.0045)	0.1707	0.0889	0.1551	0.1479	0.0004
EOGVS	0.2111 (0.0313)	0.0218	0.0304	0.0511	0.0230	0.0004
Earthquakes	− 0.0330 (0.0202)	− 0.0032	− 0.0889	− 0.0032	− 0.0032	− 0.0033
ElectricDevices	0.3436 (0.0308)	0.1489	− 0.0001	0.1794	0.0031	0.0907
EthanolLevel	0.0050 (0.0298)	0.0011	0.0003	0.0000	0.0020	0.0018
FaceAll	0.6074 (0.0583)	0.0877	0.1105	0.1172	0.0450	0.0860
FaceFour	0.6560 (0.0017)	0.1458	0.1365	0.1458	0.1365	0.6964
FacesUCR	0.6095 (0.0141)	0.0834	0.1105	0.1172	0.0450	0.0781
FiftyWords	0.4635 (0.0248)	0.2680	0.5525	0.5909	0.5329	0.5291
Fish	0.3937 (0.0412)	0.0010	0.0013	0.0053	0.0013	0.0013

Table 5 (continued)

	TSRF-Dist	ERP	WDTW	ADTW	CDTW	TWE
FordA	0.0205 (0.0309)	− 0.0000	0.0000	− 0.0000	0.0000	− 0.0001
FordB	0.0854 (0.0453)	− 0.0001	− 0.0001	0.0000	− 0.0001	− 0.0000
FreezerRT	0.2814 (0.0309)	0.0002	0.0002	0.0002	0.0002	0.0001
FreezerST	0.2860 (0.0427)	0.0002	0.0002	0.0002	0.0002	0.0002
Fungi	0.9271 (0.0394)	0.9568	0.7665	0.7238	0.8380	0.8221
GestureMidAirD1	0.3826 (0.0456)	0.0324	0.1059	0.1406	0.1014	0.0152
GestureMidAirD2	0.3221 (0.0007)	0.1253	0.2308	0.2141	0.2491	0.0122
GestureMidAirD3	0.2430 (0.0183)	0.0270	0.0416	0.0357	0.0295	0.0078
GesturePebbleZ1	0.3767 (0.0142)	0.4038	0.3895	0.1192	0.2666	0.0516
GesturePebbleZ2	0.3890 (0.0091)	0.4038	0.3895	0.1192	0.2666	0.0516
GunPoint	− 0.0050 (0.0536)	0.0308	− 0.0049	0.0308	0.0091	0.0091
GunPointAgeSpan	0.0744 (0.0552)	− 0.0017	− 0.0017	− 0.0017	− 0.0017	− 0.0017
GunPMVF	0.0428 (0.0000)	0.2340	0.2340	0.2340	0.2340	0.2340
GunPOVY	0.0447 (0.0029)	0.2384	0.2384	0.2384	0.2384	0.2384
Ham	0.0147 (0.0023)	0.0007	0.0007	0.0007	0.0007	0.0007
HandOutlines	0.1842 (0.0021)	0.0160	0.0022	0.0011	0.0011	0.0011
Haptics	0.0895 (0.0195)	0.0011	0.0020	0.0023	0.0036	0.0023
Herring	− 0.0040 (0.0884)	− 0.0053	− 0.0053	− 0.0210	− 0.0053	− 0.0122
HouseTwenty	0.3635 (0.0101)	0.0071	0.0034	− 0.0027	0.0201	− 0.0027
InlineSkate	0.0216 (0.0071)	0.0073	0.0084	0.0089	0.0093	0.0096
InsectEPGRT	0.4211 (0.2009)	1.0000	1.0000	1.0000	1.0000	1.0000
InsectEPGST	0.4460 (0.0049)	1.0000	1.0000	1.0000	1.0000	1.0000
InsectWS	0.3714 (0.0425)	0.0161	0.2002	0.2763	0.2949	0.1069
ItalyPD	0.0025 (0.0232)	0.0005	− 0.0004	− 0.0003	0.0221	0.0124
LargeKA	0.0266 (0.0189)	− 0.0000	0.0030	− 0.0000	0.0364	− 0.0000
Lightning2	0.0846 (0.0023)	0.1252	0.1202	0.1202	0.1202	0.1097
Lightning7	0.3610 (0.0180)	0.4275	0.3541	0.4545	0.1938	0.3508
Mallat	0.9724 (0.0552)	0.5925	0.8654	0.7610	0.8569	0.5925
Meat	0.8337 (0.0352)	0.5316	0.5551	0.5551	0.5551	0.5316
MedicalImages	0.0592 (0.0518)	0.0022	0.1102	− 0.0746	− 0.0316	− 0.0574
MelbourneP	0.4288 (0.1080)	0.1341	0.1418	0.1399	0.1436	0.0619
MiddlePOAG	0.4237 (0.0247)	0.4040	0.4040	0.0040	0.4040	0.4084
MiddlePOC	− 0.0054 (0.0171)	− 0.0009	− 0.0009	− 0.0009	− 0.0009	0.0014
MiddlePhalanxTW	0.3208 (0.0023)	0.5175	0.5156	0.5204	0.5128	0.5239
MixedSRT	0.6728 (0.0000)	0.1174	0.1319	0.0165	0.3003	0.0007
MixedSST	0.6544 (0.0531)	0.1149	0.1249	0.0160	0.1256	0.0991
MoteStrain	0.4879 (0.0712)	0.0003	0.0005	0.0005	0.0005	0.0003
NonIFECGT	0.5998 (0.0564)	0.1011	0.0738	0.0892	0.0807	0.0277
NonIFECGT	0.6992 (0.0875)	0.1173	0.1175	0.1303	0.1295	0.0456
OSULeaf	0.3301 (0.0225)	0.0982	0.0523	0.0936	0.0155	0.1402
OliveOil	0.7145 (0.0147)	0.5132	0.4615	0.4615	0.4615	0.5132
PLAID	0.1367 (0.0276)	0.0157	0.0181	0.0238	0.0206	0.0020

Table 5 (continued)

	TSRF-Dist	ERP	WDTW	ADTW	CDTW	TWE
PhalangesOC	− 0.0008 (0.0240)	0.0008	0.0006	0.0006	0.0012	0.0006
Phoneme	0.1334 (0.0138)	0.0750	0.0285	0.0597	0.0359	0.0041
PickupGWZ	0.3410 (0.0000)	0.2514	0.3102	0.3290	0.3222	0.3596
PigAP	0.0629 (0.0070)	0.0807	0.0783	0.0716	0.0719	0.0869
PigArtPressure	0.4075 (0.0280)	0.3136	0.2460	0.2696	0.2583	0.1617
PigCVP	0.0821 (0.0025)	0.0590	0.0421	0.0984	0.0557	0.1158
Plane	0.9544 (0.0276)	0.8419	0.8464	0.7039	0.7002	1.0000
PowerCons	0.0590 (0.0085)	0.0022	0.0000	0.0040	0.0074	0.0156
ProximalPOAG	0.5315 (0.0700)	0.5395	0.5395	0.5411	0.5395	0.5411
ProximalPOC	0.0409 (0.0420)	0.0745	0.0748	0.0050	0.0748	0.0760
ProximalPTW	0.3555 (0.0892)	0.6779	0.6653	0.5864	0.6632	0.5677
RefrigerationD	0.0302 (0.0000)	0.0097	− 0.0000	− 0.0001	− 0.0000	0.0095
Rock	0.2343 (0.0244)	0.2784	0.3678	0.2643	0.1985	0.1679
ScreenType	0.0320 (0.0166)	0.0003	0.0000	0.0000	− 0.0000	0.0005
SemgHGC	0.0818 (0.0552)	− 0.0052	0.0034	0.0034	0.0034	− 0.0050
SemgHMC	0.0932 (0.0199)	0.0009	0.0398	0.0088	0.0444	0.0001
SemgHSC	0.2797 (0.0282)	0.0002	0.0125	0.0009	0.0212	0.0001
ShakeGWZ	0.5230 (0.0181)	0.2810	0.3140	0.2734	0.2009	0.3137
ShapeletSim	0.4763 (0.0548)	0.0000	0.0000	0.0383	− 0.0002	0.0000
ShapesAll	0.4809 (0.0353)	0.0833	0.1011	0.0948	0.1004	0.0969
SmallKA	0.2059 (0.2793)	0.0002	0.0000	0.0008	0.0014	0.0002
SmoothSubspace	0.6797 (0.0040)	0.9410	0.2169	0.8154	0.5361	0.9702
SonyAIBORS	0.6044 (0.0122)	− 0.0085	− 0.0117	− 0.0116	− 0.0117	− 0.0029
SonyAIBORS	0.2977 (0.0203)	0.0025	0.0025	0.0025	0.0012	0.0025
StarLightCurves	0.5160 (0.0778)	0.6764	0.6757	0.6743	0.6763	0.6720
Strawberry	0.0417 (0.0523)	− 0.0186	− 0.0270	− 0.0315	− 0.0186	− 0.0264
SwedishLeaf	0.6550 (0.0019)	0.0360	0.0386	0.0568	0.0392	0.0668
Symbols	0.9519 (0.0360)	0.6757	0.6752	0.6752	0.4818	0.6744
SyntheticC	0.7522 (0.0846)	0.6268	0.6786	0.6654	0.6786	0.6261
ToeS	0.0598 (0.0380)	0.0245	0.0119	0.0015	− 0.0035	0.0477
ToeS	0.0310 (0.0154)	0.0466	0.0699	0.0699	0.0093	0.0466
Trace	0.3880 (0.0393)	0.6649	0.7047	0.6304	0.4225	0.3984
TwoLeadECG	0.0138 (0.0559)	0.0000	0.0000	0.0000	0.0002	0.0002
TwoPatterns	0.0094 (0.0572)	0.9995	0.0001	0.1339	0.0839	0.0226
UMD	0.1649 (0.0566)	0.1561	0.1455	0.1131	0.1715	0.1115
UWGLA	0.6819 (0.0188)	0.1338	0.6375	0.3003	0.4618	0.2183
UWGLX	0.4742 (0.0123)	0.2541	0.3227	0.3399	0.3484	0.2300
UWGLY	0.3010 (0.0150)	0.2395	0.3105	0.2990	0.3301	0.2832
UWGLZ	0.4504 (0.0286)	0.3185	0.2737	0.2673	0.2947	0.2870
Wafer	0.1432 (0.0461)	0.0014	− 0.0001	− 0.0001	− 0.0001	0.0008
Wine	− 0.0004 (0.0340)	0.0050	0.0050	0.0050	0.0050	0.0050
WordSynonyms	0.2740 (0.0215)	0.1430	0.2846	0.2733	0.3665	0.3117

Table 5 (continued)

	TSRF-Dist	ERP	WDTW	ADTW	CDTW	TWE
Worms	0.1767 (0.0002)	0.0594	0.0646	0.1341	0.1191	0.1788
WormsTwoClass	0.0147 (0.0113)	− 0.0042	− 0.0032	− 0.0116	− 0.0148	− 0.0117
Yoga	0.0024 (0.0130)	− 0.0034	− 0.0022	0.0064	− 0.0021	− 0.0001

Acknowledgements The research leading to these results has received funding from the European Union-NextGenerationEU through the Italian Ministry of University and Research under PNRR-M4C2-I1.3 Project PE_00000019 "HEAL ITALIA" to Carlo Combi CUP B33C22001030006. The views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them. We would like to thank the donors and curators of the data to the UCR Time Series Archive.

Funding Open access funding provided by Università degli Studi di Verona within the CRUI-CARE Agreement.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

References

Abanda A, Mori U, Lozano JA (2019) A review on distance based time series classification. Data Min Knowl Discov 33:378–412

Aryal S, Ting KM, Washio T, Haffari G (2017) Data-dependent dissimilarity measure: an effective alternative to geometric distance measures. Knowl Inf Syst 53:479–506

Aryal S, Ting KM, Washio T, Haffari G (2020) A comparative study of data-dependent approaches without learning in measuring similarities of data objects. Data Min Knowl Discov 34:124–162

Aryal S, Ting KM, Haffari G, Washio T (2014) Mp-dissimilarity: a data dependent dissimilarity measure. In: 2014 IEEE international conference on data mining

Bicego M (2019) K-random forests: a K-means style algorithm for random forest clustering. In: Proc Int Joint Conf Neural Netw (IJCNN2019)

Bicego M, Cicalese F, Mensi A (2023) RatioRF: a novel measure for random forest clustering based on the Tversky's ratio model. IEEE Trans Knowl Data Eng 35:830–841

Bicego M, Cicalese F (2023) On the good behaviour of extremely randomized trees in random forest-distance computation. In: Joint European conference on machine learning and knowledge discovery in databases

Bicego M, Escolano F (2020) On learning random forests for random forest-clustering. In: 25th International conference on pattern recognition

Breiman L (2000) Some infinity theory for predictor ensembles. Technical report, Citeseer
Breiman L (2001) Random forests. Mach Learn 45:5–32
Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees
Chen K, Zhang D, Yao L, Guo B, Yu Z, Liu Y (2021) Deep learning for sensor-based human activity recognition: overview, challenges, and opportunities. ACM Comput Surv 54:1–40
Chen L, Ng RT (2004) On the marriage of lp-norms and edit distance. In: (e)Proceedings of the thirtieth international conference on very large data bases, VLDB 2004, Toronto, Canada, August 31–September 3 2004
Chen L, Özsu MT, Oria V (2005) Robust and fast similarity search for moving object trajectories. In: Proceedings of the ACM SIGMOD international conference on management of data, Baltimore, Maryland, USA, June 14-16, 2005
Criminisi A, Shotton J, Konukoglu E (2012) Decision forests: a unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Found Trends Comput Graph Vis 7:81–227
Dau HA, Bagnall A, Kamgar K, Yeh C-CM, Zhu Y, Gharghabi S, Ratanamahatana CA, Keogh E (2019) The UCR time series archive. IEEE/CAA J Automatica Sinica 6:1293–1305
Davies A, Ghahramani Z (2014) The random forest kernel and other kernels for big data from random partitions. arXiv preprint arXiv:1402.4293
Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
Deng H, Runger GC, Tuv E, Martyanov V (2013) A time series forest for classification and feature extraction. Inf Sci 239:142–153
Esling P, Agon C (2012) Time-series data mining. ACM Comput Surv (CSUR) 45:1–34
Fulcher BD, Jones NS (2017) hctsa: a computational framework for automated time-series phenotyping using massive feature extraction. Cell Syst 5:527–531
Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. Mach Learn 63:3–42
Górecki T (2014) Using derivatives in a longest common subsequence dissimilarity measure for time series classification. Pattern Recognit Lett 45:99–105
Herrmann M, Webb GI (2023) Amercing: an intuitive and effective constraint for dynamic time warping. Pattern Recognit 137:109333
Holder C, Middlehurst M, Bagnall A (2024) A review and evaluation of elastic distance functions for time series clustering. Knowl Inf Syst 66:765–809
Hubert L, Arabie P (1985) Comparing partitions. J Classif
Jain BJ (2019) Making the dynamic time warping distance warping-invariant. Pattern Recognit 94:35–52
Jeong Y, Jeong MK, Omitaomu OA (2011) Weighted dynamic time warping for time series classification. Pattern Recognit 44:2231–2240
Karevan Z, Suykens JAK (2020) Transductive LSTM for time-series prediction: an application to weather forecasting. Neural Netw 125:1–9
Karlsson I, Papapetrou P, Boström H (2016) Generalized random shapelet forests. Data Min Knowl Discov 30:1053–1085
Keogh EJ, Pazzani MJ (2001) Derivative dynamic time warping. In: Proceedings of the first SIAM international conference on data mining, SDM 2001, Chicago, IL, USA, April 5–7, 2001
Liu FT, Ting KM, Zhou Z-H (2008) Isolation forest. In: 2008 Eighth IEEE international conference on data mining
Liu FT, Ting KM, Zhou Z (2012) Isolation-based anomaly detection. ACM Trans Knowl Discov Data 6:1–39
Lu S, Lu J, An K, Wang X, He Q (2023) Edge computing on IoT for machine signal processing and fault diagnosis: a review. IEEE Internet of Things J 10:11093–11116
Lubba CH, Sethi SS, Knaute P, Schultz SR, Fulcher BD, Jones NS (2019) catch22: Canonical time-series characteristics: selected through highly comparative time-series analysis. Data Min Knowl Discov 33:1821–1852
Lucas B, Shifaz A, Pelletier C, O'Neill L, Zaidi N, Goethals B, Petitjean F, Webb GI (2019) Proximity forest: an effective and scalable distance-based classifier for time series. Data Min Knowl Discov 33(3):607–635. https://doi.org/10.1007/s10618-019-00617-3
Marteau P (2009) Time warp edit distance with stiffness adjustment for time series matching. IEEE Trans Pattern Anal Mach Intell 31:306–318
Mensi A, Tax DMJ, Bicego M (2023) Detecting outliers from pairwise proximities: proximity isolation forests. Pattern Recognit 138:109334. https://doi.org/10.1016/J.PATCOG.2023.109334

Middlehurst M, Large J, Flynn M, Lines J, Bostrom A, Bagnall AJ (2021) HIVE-COTE 2.0: a new meta ensemble for time series classification. Mach Learn 110:3211–3243
Middlehurst M, Ismail-Fawaz A, Guillaume A, Holder C, Guijo-Rubio D, Bulatova G, Tsaprounis L, Mentel L, Walter M, Schäfer P, Bagnall AJ (2024) aeon: a python toolkit for learning from time series. CoRR arxiv: 2406.14231
Middlehurst M, Large J, Bagnall A (2020) The canonical interval forest (CIF) classifier for time series classification. In: 2020 IEEE international conference on big data (big data)
Mietus JE (2002) The pNNx files: re-examining a widely used heart rate variability measure. Heart 88:378–380
Mishra K, Basu S, Maulik U (2022) Graft: a graph based time series data mining framework. Eng Appl Artif Intell 110:104695
Mitra R, MacLean AL (2021) Rvagene: generative modeling of gene expression time series data. Bioinformatics 37:3252–3262
Moosmann F, Triggs B, Jurie F (2006) Fast discriminative visual codebooks using randomized clustering forests. In: Advances in neural information processing systems 19
Morid MA, Sheng ORL, Dunbar J (2023) Time series prediction using deep learning methods in healthcare. ACM Trans Manag Inf Syst 14:1–29
Paparrizos J, Gravano L (2015) k-shape: efficient and accurate clustering of time series. SIGMOD Rec
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, VanderPlas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2019) Scikit-learn: Machine learning in python. J Mach Learn Res
Quinlan JR (1993) C4.5: programs for machine learning
Ran X, Xi Y, Lu Y, Wang X, Lu Z (2023) Comprehensive survey on hierarchical clustering algorithms and the recent developments. Artif Intell Rev 56:8219–8264
Ratanamahatana CA, Keogh EJ (2005) Three myths about dynamic time warping data mining. In: Proceedings of the 2005 SIAM international conference on data mining, SDM 2005, Newport Beach, CA, USA, April 21–23, 2005
Ruiz AP, Flynn M, Large J, Middlehurst M, Bagnall A (2021) The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Discov 35:401–449
Shehab M, Abualigah L, Shambour Q, Abu-Hashem MA, Shambour MKY, Alsalibi AI, Gandomi AH (2022) Machine learning in medical applications: a review of state-of-the-art methods. Comput Biol Med 145:105458
Shi T, Horvath S (2006) Unsupervised learning with random forest predictors. J Comput Graph Stat 15:118–138
Shotton J, Johnson M, Cipolla R (2008) Semantic texton forests for image categorization and segmentation. In: IEEE computer society conference on computer vision and pattern recognition (CVPR 2008)
Stefan A, Athitsos V, Das G (2013) The move-split-merge metric for time series. IEEE Trans Knowl Data Eng 25:1425–1438
Ting KM, Zhu Y, Carman M, Zhu Y, Washio T, Zhou Z-H (2019) Lowest probability mass neighbour algorithms: relaxing the metric constraint in distance-based neighbourhood algorithms. Mach Learn
Ting KM, Zhu Y, Carman M, Zhu Y, Zhou Z (2016) Overcoming key weaknesses of distance-based neighbourhood methods using a data dependent dissimilarity measure. In: Proc Int Conf Knowl Discov Data Min
Ting KM, Zhu Y, Zhou Z-H (2018) Isolation kernel and its effect on SVM. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining
Tversky A (1977) Features of similarity. Psychol Rev 84:327
Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Discov 26:275–309
Wang X, Wirth A, Wang L (2007) Structure-based statistical features and multivariate time series clustering. In: Proceedings-IEEE international conference on data mining, ICDM
Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull
Yan D, Chen A, Jordan MI (2013) Cluster forests. Comput Stat Data Anal 66:178–192
Zhao K, Wulder MA, Hu T, Bright R, Wu Q, Qin H, Li Y, Toman E, Mallick B, Zhang X, Brown M (2019) Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: a Bayesian ensemble algorithm. Remote Sens Environ 232:111181

Zhu X, Loy CC, Gong S (2014) Constructing robust affinity graphs for spectral clustering. In: Proc Int Conf Comput Vision Pattern Recognit, CVPR 2014