Expert Systems With Applications 213 (2023) 119227

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Complexity measures and features for times series classification

Francisco J. Baldán *, José M. Benítez

Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), DICITS Lab, iMUDS, University of Granada, 18071, Granada, Spain

ARTICLE INFO

Keywords:
Classification
Complexity measures
Time series features
Interpretability

ABSTRACT

Time series classification is a growing problem in different disciplines due to the progressive digitalization of the world. The best state-of-the-art algorithms focus on performance, seeking the best possible results, leaving interpretability at a second level, if any. Furthermore, interpretable proposals are far from providing competitive results. In this work, focused on time series classification, we propose a new representation of time series based on a robust and complete set of features. This new representation allows extracting more meaningful information on the underlying time series structure to develop effective classifiers whose results are much easier to interpret than current state-of-the-art models. The proposed feature set allows using the traditional vector-based classification algorithms in time series problems, significantly increasing the number of techniques available for this type of problem. To evaluate the performance of our proposal, we have used the state-of-the-art repository of time series classification, UCR, composed of 112 datasets. The experimental results show that through this representation, more interpretable classifiers can be obtained which are competitive. More specifically, they obtain no statistically significant differences from the second and third-best models of the state-of-the-art. Apart from competitive results in accuracy, our proposal is able to improve the model interpretability based on the set of features proposed.

1. Introduction

At present, large amounts of information are recorded from a wide variety of fields. There is a growing need to analyze and classify these data to obtain useful information, for example, to identify different patterns of electricity consumption in order to adapt prices to consumers (Markovič, Gosak, Grubelnik, Marhl, & Virtič, 2019), to identify cardiac anomalies characteristics of a pathology (Chauhan, Vig, & Ahmad, 2019) or search for anomalies in starlight curves (Twomey, Chen, Diethe, & Flach, 2019).

The field of time series classification (TSC) (Bagnall, Lines, Bostrom, Large, & Keogh, 2017) has historically been dominated by proposals that offer good classification results but are hardly interpretable. The interpretability of a model allows us to follow the logic and understand the process realized to obtain the final prediction (Adadi & Berrada, 2018; Arrieta et al., 2020). In our field, we identify a model as more interpretable than others when the relationship between the input variables and the output variable is clearer (Schlegel, Arnout, El-Assady, Oelke, & Keim, 2019). Additionally, this relationship must be able to provide additional information about the problem. Without this information, the interpretability of the results is very limited.

On the one hand, we have simple approaches that achieve good average results in the different types of problems as One Nearest Neighbor with Dynamic Time Warping (1NN+DTW) (Berndt & Clifford, 1994; Ratanamahatana & Keogh, 2004). This algorithm tells us how similar the time series are to each other, but it does not allow us to extract additional information from the problem. On the other hand, we have proposals with high complexity that achieve the best possible results as The Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) (Bagnall, Flynn, Large, Lines, & Middlehurst, 2020). HIVE-COTE is composed of several classifiers of different domains distributed in five large modules. Each module provides a probability estimate for each class and obtains a weighting proportional to the accuracy achieved over the training set. HIVE-COTE combines these estimates in a second layer and obtains the predicted class from the highest weight over all the modules. The HIVE-COTE proposal provides the best results, but its interpretability is very low, and its high computational cost prevents its application in large datasets.

In the literature, we can find other types of proposals focused on extracting a large number of features from time series (Fulcher, 2018; Fulcher, Little, & Jones, 2013). The main idea of these features

* Corresponding author.
E-mail addresses: fjbaldan@decsai.ugr.es (F.J. Baldán), J.M.Benitez@decsai.ugr.es (J.M. Benítez).

https://doi.org/10.1016/j.eswa.2022.119227
Received 16 April 2022; Received in revised form 19 October 2022; Accepted 3 November 2022
Available online 9 November 2022

is any type of mathematical operation applicable over a time series that provide valuable information. These proposals aim to look for an underlying structure that represents the behavior of a time series. These studies are ambitious but difficult to interpret over a specific problem due to the high number of features present. Moreover, these studies focus on the unsupervised learning environment. Some proposals select the main features of a time series from the theoretical point of view that could explain the origin of their behavior (Kang, Hyndman, Li, et al., 2018). The objective of the previous work is to generate synthetics time series that represent real problem behaviors, so its main target is far from the problem of TSC. On the other hand, CAnonical Time-series CHaracteristics (Catch22) (Lubba et al., 2019) proposes a set of 22 features that have been selected based on the classification results obtained on a large set of datasets. For this proposal, a large number of features and their possible combinations have been tested, measuring the classification results obtained. The main criterion for selecting the features is to provide the best possible results, although the execution time and, in some cases, their interpretability is also taken into account.

The interpretability of the results has so far been ignored by these proposals due to the high number of initial features, their heterogeneous origin, the experimental features selection approach, and the complexity of applied models.

In this work, we propose a new representation for time series based on a set of complexity measures and representative time series features, which are selected based on their theoretical basis and interpretability. The main advantages of this proposal are:

- The proposed representation allows express time series in a different domain, structured as a vector, thus enabling the use of traditional vector-based algorithms on TSC problems. Because of this, our proposal significantly increases the number of available techniques to address time series classification problems.
- The time series feature selection based on the features' theoretical basis provides us robust results over unseen new time series problems, contrary to other proposals based on selecting the best features for a given set of datasets through intensive experimentation.
- The focus of this representation is to improve the interpretability of the built classifiers, particularly those that already excel at easy understanding, such as tree-based ones. The selected well-known time series features increase the interpretability of the processed time series and the interpretable models created. They provide additional interpretability that can be mandatory in critical applications.
- The proposed transformation is especially useful in applications with heterogeneous time series, e.g., of different lengths or non-stationary. Those behaviors prevent the application of a high number of the main proposals of the state-of-the-art.
- The computational complexity of our proposal is low. It can be simplified to the sum of the features' highest complexity, which is $time_series_length^2$, and the computational complexity of the classification model used. The latter is low if we compare it with the complexity of the main algorithms of the state-of-the-art, which are quadratic at least.

The performance of our proposal has been tested on a set of 112 datasets from the UCR repository (Dau et al., 2019). We have applied the most popular and widely used classification algorithms based on trees that allow interpretable results. Our proposal is publicly available as an R package in the online repository.[1]

The rest of the work is organized as follows: Section 2 introduces the state-of-the-art in TSC: distance-based methods, feature-based methods, and deep learning methods. In Section 3, we describe in-depth our proposal. Section 4 shows the experimental design used, the results obtained, and the interpretability of these results. Finally, Section 5 concludes the paper.

2. Related work

There are several ways to group the TSC algorithms. In this work, we group them by the type of data used for each algorithm and its internal operation. In this way, we have three principal groups with their corresponding subgroups. The first group is composed of the distance-based proposals (Section 2.1), which are strongly related to calculations of similarity and distance between different time series or subsequences of the time series themselves. The second group is composed of the deep learning proposals (Section 2.2), where data entry and processing depend entirely on each proposal. The last group would be made up of feature-based proposals (Section 2.3), which are based on the calculation of certain parameters of the time series that transform the original data. After this transformation, traditional classification algorithms are applied to the new dataset.

2.1. Distance-based classification

Patterns searched for in TSC problems may have their origin in different domains. For this reason, there are different types of approaches depending on multiple factors. There are currently six main approaches for dealing with this kind of TSC problems (Bagnall et al., 2017), grouped by the type of discriminatory features that the technique attempts to find:

- Proposals that use all the values of the time series: are linked to the use of similarity measures and different types of distance. The reference algorithm of this group is 1NN+DTW, which is simple to apply but has high computational complexity. This algorithm is often used as a benchmark in TSC problems.
- Those using phase-dependent intervals: they use small subsets from each time series, rather than using the entire time series. Proposals like Time Series Forest (TSF) (Deng, Runger, Tuv, & Vladimir, 2013) have been proved that extracting features such as mean, variance, or slope from random intervals, and use them as classifier features, works particularly well.
- The independent phases, based on shapelets: the shapelets based ones look for subsequences of the time series that allow differentiating the time series belonging to each class (Mueen, Keogh, & Young, 2011; Rakthanmanon & Keogh, 2013; Ye & Keogh, 2009). They are closely linked to the use of similarity and distance measurements. Recent work on shapelets has shown that they are particularly useful when used as input features to a traditional classification algorithm (Baldán & Benítez, 2019; Bostrom & Bagnall, 2017; Lines, Davis, Hills, & Bagnall, 2012), rather than as part of the classification tree itself.
- Based on dictionaries: useful, when the frequency of a pattern in a time series is determinant to classify it correctly (Lin & Li, 2009; Schäfer, 2015). These algorithms count both the presence or absence of each subsequence in a time series. They create a classifier based on the histograms obtained from these dictionaries. The dictionary creation way is one of the main differences among the proposals of this type.
- Based on models: this approach is mainly oriented to problems with long time series, but with different lengths (Bagnall & Janacek, 2014; Chen, Tang, Tino, & Yao, 2013). These proposals usually fit a model to each time series and measure the similarity between the models. It is an approach that is not sufficiently widespread and is applied to particular problems.

[1] Complexity Measures and Features for Times Series classification. https://github.com/fjbaldan/CMFTS/.

- Combinations or ensembles: this approach works both in time series and traditional classification problems, using the results of different models to make a final decision. In the area of time series, HIVE-COTE (Bagnall et al., 2020) is the best proposal to date. It uses models from different approaches and offers the best accuracy results. On the other hand, it is the approach with the highest computational complexity due to the high number of algorithms it uses and its corresponding computational complexities. Moreover, this large number of algorithms leads to low interpretability of results.

Each of these approaches adapts to different types of problems, but they all work on the original values of the time series.

2.2. Deep learning classification

The approach based on deep learning has gained popularity recently, providing very interesting proposals in the field of TSC (Fawaz, Forestier, Weber, Idoumghar, & Muller, 2019; Wang, Yan, & Oates, 2017). We can distinguish between two main groups within this approach: Generative Models and Discriminative Models.

- In the Generative Models, there is usually a previous step of unsupervised training to the learning phase of the classifier. Depending on the approach, two subgroups can be differentiated: Auto Encoders and Echo State Networks. In the case of Auto Encoders, there are a large number of proposals, for example, to model the time series before the classifier is included in an unsupervised pre-training phase such as Stacked Denoising Auto-Encoders (SDAEs) (Bengio, Yao, Alain, & Vincent, 2013). In the case of the Echo State Networks, these networks were used to reconstruct time series and use the representation learned in the space reservoir for classification. They were also used to define a kernel on the learned representations and apply an MLP or SVM as a classifier.
- In the case of Discriminative Models, these are a classifier or regressor that learns the mapping between the input values of the time series and returns the probability distribution over the class variable of the problem. In this case, we can differentiate two subgroups: Feature Engineering and End-to-End. The typical case of use of Feature Engineering is the transformation of the time series into images, using different techniques such as recurrence plots (Hatami, Gavet, & Debayle, 2018) and Markov transition fields (Wang & Oates, 2015), and introduce that information in a deep learning discriminating classifier (Nweke, Teh, Al-Garadi, & Alo, 2018). In contrast, the End-to-End approach incorporates feature learning while adjusting the discriminative classifier.

If we look at the TSC problem, we see that the Convolutional Neural Networks (CNNs) are the most used architectures, mainly due to their robustness and their relatively short training time, compared to other types of networks. One of the best-known architectures is the Residual Networks (ResNets) (Wang et al., 2017). This proposal adds linear shortcuts for the convolutional layers, potentially improving the accuracy of the model.

2.3. Feature-based classification

The feature-based approach is focused on the time series dataset transformation, obtaining a new dataset composed of different features that explain the behavior of the original time series (Fulcher, 2018). The feature-based approach offers multiple advantages over the distance-based approach for dealing with time-series classification problems. This approach allows analysis of time series on different time domains and lengths, being more widely applicable because the stationarity properties of the series are not always required. In addition, this approach allows us to use the standard classification methods that have been developed for non-time series data. In this approach, we can find two different approximations:

- The first one is based on the use of a reduced set of features with a strong theoretical basis that is easily interpretable. In addition to applying traditional learning algorithms to the problem, this approach offers the possibility to analyze the extracted parameters and to obtain additional information.

  Based on this approach, we can find proposals that, with a minimum of four initial features such as mean, typical deviation, skewness, and kurtosis, are able to face the problem of the classification synthetic control chart patterns used in the statistical process control (Nanopoulos, Alcock, & Manolopoulos, 2001). There are also proposals, focused on the improvement of accuracy, based on the creation of an ensemble for classification, composed of trained classifiers on different representations of the time series (Bagnall, Davis, Hills, & Lines, 2012): power spectrum, autocorrelation function, and a principal components space. The final classification is obtained from a weighted voting scheme.

  In this approach, we also find proposals that aim to generate synthetic time series with a given behavior as close as possible to a real time series (Kang et al., 2018). This work contains a selection of the main features of a time series. Its objective is to use them to generate time series with a real behavior with these controllable parameters.
- The second approach focuses on applying a large number of different operations to obtain a great set of descriptive parameters of the time series analyzed. In this approach, the selection of the features of interest resides in the learning algorithm used on the transformed dataset. Having a much greater set of features than the first approach allows us to capture a higher number of behaviors of interest, improving the results of the algorithms applied afterward. But it is hard to extract useful information because there are a large number of features to analyze. In addition, it is possible that a large part of the selected features is not as explanatory as the features with a strong theoretical base such as trend, seasonality, among others.

  In this area, we can find different proposals. One of the most representative proposals starts from almost 9000 features, coming from different fields, being of special importance the way to select the variables of interest (Fulcher & Jones, 2014). The authors seek to obtain the best classification results by working on the transformed dataset. This proposal opted for the selection of the combination of variables that offers the best classification results, using the following procedure:

  In the first place, the proposal selects the variable that obtains the best classification result by itself. Then, one by one, it combines the previously selected variable with the rest of the variables, and the variable that offers the best results is selected as the second variable. This set of two variables is then combined with each of the other variables and evaluated. This process is repeated until the stop criterion is met. However, this proposal entails a high computational complexity due to the large number of combinations available.

  As we can see, the interpretability of the results has not been among the main objectives of these types of proposals.

In summary, we can see that the feature-based approach in the TSC field is the one that offers the greatest number of possibilities to expand the number of directly applicable algorithms and to obtain interpretable results. This approach allows us to use all the interpretable classification algorithms available for the vector-based traditional data. The proposals that achieve better results are focused on obtaining the best set of features experimentally on multiple datasets. The interpretability of those proposals is seriously hindered because they start out of a huge set of candidates and they are different for different problems, lacking a general setting. On the other hand, proposals that offer

Table 1
Complexity measures selected.

Char.	Name	Description	Ref.
C_1	lempel_ziv	LempelZiv (LZA)	Lempel and Ziv (1976)
C_2	approximation_entropy	Approximation Entropy	Pincus (1991)
C_3	sample_entropy	Sample Entropy (DK Lake in Matlab)	Richman and Moorman (2000)
C_4	permutation_entropy	Permutation Entropy (tsExpKit)	Bandt and Pompe (2002)
C_5	shannon_entropy_CS	Chao–Shen Entropy Estimator	Chao and Shen (2003)
C_6	shannon_entropy_SG	Schurmann–Grassberger Entropy Estimator	Schürmann and Grassberger (1996)
C_7	spectral_entropy	Spectral Entropy	Zhang, Yang, and Huang (2008)
C_8	nforbidden	Number of forbidden patterns	Amigó (2010)
C_9	kurtosis	Kurtosis, the "tailedness" of the probability distribution	DeCarlo (1997)
C_{10}	skewness	Skewness, asymmetry of the probability distribution	Čisar and Čisar (2010)

closed feature sets have not been able to provide competitive results. For these reasons, it is necessary to develop a robust set of measures, which does not depend on the problem faced, that provides competitive results and improves the interpretability of the used models.

3. Time series complexity measures and features for classification

In this section, our proposal to obtain interpretable time series classifiers through a new representation based on complexity measures and features is presented.

The complexity of a time series represents the interrelationship that exists among its different elements. It is logical to assume that time series with similar complexity share characteristics between them, like a similar trend or a similar seasonal component. In this way, we can consider this complexity as another feature, and we can use different complexity measures to compare time series with each other.

The features of a time series explain certain behaviors typical of the time series itself. The features that traditionally have been used in the process of analysis of a time series such as seasonality, trend, and stationarity, among others, are well documented (Baldán, Ramírez-Gallego, Bergmeir, Herrera, & Benítez, 2018; Kang et al., 2018). These types of features can describe the behavior of a time series effectively. There are other types of features that provide small pieces of information about the behavior of a time series, such as mean, maximum value, minimum value, variance, among others. These are features that may be especially useful depending on the problem. For example, in a classification problem where time series of different classes have significant differences in their value ranges, the mean can be very helpful.

This work presents a novel time series representation based on an ensemble of complexity measures and features of time series, aimed at solving problems of classification of time series by applying traditional classification algorithms. It also aims to improve the interpretability of models. The features selected, complexity measures and time series features, are based on information theory and seek to provide deeper knowledge about the underlying structure of the represented time series. The first group of features, based on measures of complexity, is included in Table 1.

In addition to the features mentioned above, we have added a set of time series features. They have been selected based on their theoretical basis, taking into account also their historical importance in the field of time series and its interpretability (Kang et al., 2018). Those features have proved their robustness and usability, being the basis of a time series generator that is able to produce time series with specific behaviors that are quite similar to the real ones. This second group of features is listed in Table 2.

Table 2
Time series features selected.

Char.	Name	Description
C_{11}	x_acf1	First autocorrelation coefficient
C_{12}	x_acf10	Sum of squares of the first 10 autocorrelation coefficients
C_{13}	diff1_acf1	Differenced series first autocorrelation coefficients
C_{14}	diff1_acf10	Differenced series sum of squares of the first 10 autocorrelation coefficients
C_{15}	diff2_acf1	Twice differenced series first autocorrelation coefficients
C_{16}	diff2_acf10	Twice differenced series sum of squares of the first 10 autocorrelation coefficients
C_{17}	max_kl_shift	Maximum shift in Kullback–Leibler divergence between two consecutive windows
C_{18}	time_kl_shift	Instant of time in which the Maximum shift in Kullback–Leibler divergence between two consecutive windows is located
C_{19}	outlierinclude_mdrmd	Calculates the median of the medians of the values, while adding more outliers
C_{20}	max_level_shift	Maximum mean shift between two consecutive windows
C_{21}	time_level_shift	Instant of time in which the maximum mean shift between two consecutive windows is located
C_{22}	ac_9	Autocorrelation at lag 9
C_{23}	crossing_points	The number of times a time series crosses the median line
C_{24}	max_var_shift	Maximum variance shift between two consecutive windows
C_{25}	time_var_shift	Instant of time in which the maximum variance shift between two consecutive windows is located
C_{26}	nonlinearity	Modified statistic from Teräsvirta's test
C_{27}	embed2_incircle	Proportion of points inside a given circular boundary in a 2-d embedding space
C_{28}	spreadrandomlocal_meantaul	Mean of the first zero-crossings of the autocorrelation function in each segment of the 100 time-series segments of length l selected at random from the original time series
C_{29}	flat_spots	Maximum run length within any single interval obtained from the ten equal-sized intervals of the sample space of a time series
C_{30}	x_pacf5	Sum of squares of the first 5 partial autocorrelation coefficients
C_{31}	diff1x_pacf5	Differenced series sum of squares of the first 5 partial autocorrelation coefficients
C_{32}	diff2x_pacf5	Twice differenced series sum of squares of the first 5 partial autocorrelation coefficients
C_{33}	firstmin_ac	Time of first minimum in the autocorrelation function
C_{34}	std1st_der	Standard deviation of the first derivative of the time series
C_{35}	stability	Stability variance of the means
C_{36}	firstzero_ac	First zero crossing of the autocorrelation function
C_{37}	trev_num	The numerator of the trev function, a normalized nonlinear autocorrelation, with the time lag set to 1
C_{38}	alpha	Smoothing parameter for the level-alpha of Holt's linear trend method
C_{39}	beta	Smoothing parameter for the trend-beta of Holt's linear trend method
C_{40}	nperiods	Number of seasonal periods (1 for no seasonal data)
C_{41}	seasonal_period	Seasonal periods (1 for no seasonal data)
C_{42}	trend	Strength of trend
C_{43}	spike	Spikiness variance of the leave-one-out variances of the remainder component
C_{44}	linearity	Linearity calculated based on the coefficients of an orthogonal quadratic regression
C_{45}	curvature	Curvature calculated based on the coefficients of an orthogonal quadratic regression

(*continued on next page*)

Table 2 (*continued*).

C_{46}	e_acf1	First autocorrelation coefficient of the remainder component
C_{47}	e_acf10	Sum of the first then squared autocorrelation coefficients
C_{48}	walker_propcross	Fraction of time series length that walker crosses time series
C_{49}	hurst	Long-memory coefficient
C_{50}	unitroot_kpss	Statistic for the KPSS unit root test with linear trend and lag one
C_{51}	histogram_mode	Calculates the mode of the data vector using histograms with 10 bins (It is possible to select a different number of bins)
C_{52}	unitroot_pp	Statistic for the PP unit root test with constant trend and lag one
C_{53}	localsimple_taures	First zero crossing of the autocorrelation function of the residuals from a predictor that uses the past trainLength values of the time series to predict its next value
C_{54}	lumpiness	Lumpiness variance of the variance
C_{55}	motiftwo_entro3	Entropy of words in the binary alphabet of length 3. The binary alphabet is obtained as follows: Time-series values above its mean are given 1, and those below the mean are 0

The possible interrelation between the different selected operations has also been analyzed, eliminating those that present high correlation values.

The final features set proposed is composed of the union of the 10 complexity measures and the 45 time series features included in Tables 1 and 2, respectively. The objective of using such features is to obtain an alternative and more informative representation of a time series. This representation allows us to use traditional classification algorithms and improve the interpretability of models. In this way, we can explain the classification based on features that describe the behavior of the processed time series if the classification algorithm applied offers an interpretable model. We can obtain information beyond the simple visual behavior of a time series.

It is important to note that all the time series do not depend on the 55 features proposed. On the one hand, these features represent the typical and well-known time series behaviors, but a time series does not have to contain all the behaviors. On the other hand, it is also possible that there are behaviors present in the time series that have not yet been formally explained or described. Because of this, the proposed set of measures cannot clearly capture them, although it is possible that some features partially reflected them. Finally, we note that the proposed transformation is not bijective. Because of this, after the application of the transformation, we cannot retrieve the original time series from the extracted feature set.

The theoretical explanation of each of the measures has not been included in this paper due to space constraints. For the convenience of the reader, they are available online in the web resource[2] associated with this work.

Our proposal consists of representing the time series as a vector of features, converting the time series dataset into a conventional vector dataset. Then, we can apply a regular vector classification algorithm. Fig. 1 shows, in a graphic form, the overall process of calculating time series features. The pseudocode in Algorithm 1 shows in-depth how our proposal works.

Our proposal begins with an individual and independent processing of each time series (line 1). The selected set of features is calculated for each time series, obtaining a set of results with as many values as features applied to the time series, namely, a vector. By processing the whole set of input time series, we calculate a matrix of values with as many columns as applied features and as many rows as processed time series. This matrix is a representation of the input time series, free of any time dependency, based on the parameters obtained when applying the operations mentioned above. As there is no time dependency in the new dataset, it is possible to use any traditional classification algorithm on this new dataset.

Although most of the proposed features are specially designed to be applied over time series, in some cases, these features may not be defined for some specific time series. In these cases, undesirable values are produced, and we must process them. In the first place, we differentiate between the cases in which we obtain infinite values and those we do not. For this reason, the results obtained are filtered, detecting the cases of noninfinite values and transforming to the same value (lines 2–5) for subsequent elimination or imputation. On the training set, we check for each column (operation applied) that the amount of these values is less than 20% of the total. In other cases, the column is removed from both the training set and the test set (lines 6–11). Infinite values are identified as positive or negative and replaced by the maximum or minimum value of the corresponding column, respectively, ignoring the infinite values in these calculations (lines 12–22). Imputation of missing values based on the mean is then applied

Algorithm 1 Main procedure

```
Input:
    train: train dataframe with (Ts_class, Ts_values)
    test: test dataframe with (Ts_class, Ts_values)
    model: selected model to be processed
Output:
    output_data: a triplet that contains the fitted models,
    the vectors with the predicted labels and the
    accuracies obtained
    f_train: features train dataframe
    f_test: features test dataframe
 1: f_train, f_test ← calc_cmfts((train.Ts_values, test.Ts_values), all)
 2: for each value in (f_train, f_test) do
 3:     if (is.na(value) || is.nan(value)) then value ← NA
 4:     end if
 5: end for
 6: for each column in f_train do
 7:     if (count.na(column) ≥ (length(column)*0.2)) then
 8:         f_train ← f_train[ , -column.index]
 9:         f_test ← f_test[ , -column.index]
10:     end if
11: end for
12: for each column in f_train do
13:     for each value in (f_train[ , column.index], f_test[ , column.index]) do
14:         if (is.infinite(value)) then
15:             if (value ≥ 0) then
16:                 value ← max(f_train[ , column.index], ignore.inf)
17:             else
18:                 value ← min(f_train[ , column.index], ignore.inf)
19:             end if
20:         end if
21:     end for
22: end for
23: for each column in (f_train, f_test) do
24:     column ← impute.Mean(column)
25: end for
26: output_data ← NULL
27: fit ← model.train(f_train, train.Ts_class)
28: pred ← fit.predict(f_test)
29: acc ← accuracy(pred, test.Ts_class)
30: output_data.add(fit, pred, acc)
31: return (output_data, f_train, f_test)
```

[2] Complexity Measures and Features for Times Series classification. http://dicits.ugr.es/papers/CMFTS/.

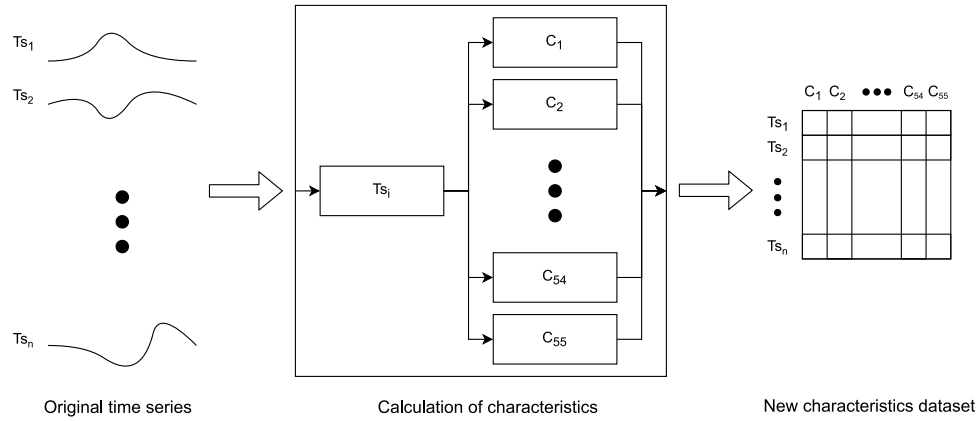


Fig. 1. Feature calculation workflow.

Table 3
Computational complexity of used models. Notation: h, height of a tree; m, number of attributes; n, number of samples; c, number of classes t, number of randomized trees; k number of attributes randomly included at each node.

Models	Time complexity
C5.0 with Boosting	$O(h \cdot m \cdot (n \cdot c + n \cdot \log(n)))$ (Khadiev, Mannapov, & Safina, 2019)
Random Forest	$O(0.632 \cdot n \cdot t \cdot k \cdot \log(0.632 \cdot n))$ (Baldán & Benítez, 2019)
Support Vector Machine	$O(n^3)$ (Abdiansah & Wardoyo, 2015)
1-Nearest Neighbor	$O(n \cdot v \cdot f)$

to each column (lines 23–25), eliminating any presence of unwanted values in the datasets.

Since one of our objectives is to obtain interpretable results, we recommend selecting techniques that produce interpretable models, such as algorithms based on trees: C5.0, C5.0 with boosting (Quinlan, 1993), Rpart (Therneau, Atkinson, et al., 1997) and Ctree (Hothorn, Hornik, & Zeileis, 2015). We initialize a variable that contains the results obtained for the processed model (line 26). In the final part, we calculate the selected model, make the corresponding prediction, and calculate the accuracy. Finally, all these data are stored (lines 27–30). Our proposal returns these data together with the training and test sets with the new calculated features (line 31).

At this point, it is necessary to proceed to the analysis of the trees obtained in search of an interpretable result that, in many cases, is difficult to appreciate in the original time series.

Finally, we consider the computational complexity of our proposal, stated in O notation, which is composed of the sum of two components: the calculation of the proposed features and the model used. The calculation of each feature is independent of the rest, so the total computational complexity of the features will be defined by the feature with the highest computational complexity. In this case, the features denominated *approximation_entropy* and *sample_entropy* have the highest computational complexity $O(l^2)$ (Manis, 2008; Manis, Aktaruzzaman, & Sassi, 2018), where l is the length of the time series to be processed. Since any traditional model can be applied to our proposal, in Table 3, we have included the computational complexities of the most competitive models used in this work.

The efficiency of the algorithms for processing time series usually depends on the lengths of time series and the number of time series to process. By using the feature representation proposed, we bound the complexity of the algorithms since the variable length of the time series is changed by a constant value, which is 55 features in this proposal. In addition, the number of used features is not dependent on the time series length. For this reason, this transformation leads to more efficient runs for the longer time series, especially if we compare it with the main state-of-the-art proposals such as HIVE-COTE or Shapelet Transform, which have a computational complexity of $O(n^2m^4)$.

Once our proposal has been exposed, two main limitations can be identified in it:

- The first limitation is related to problems composed of long-length time series in which the distinguishable pattern is contained in a very short subsequence. Our proposal extracts features from the whole time series. Due to this, it is difficult for it to extract information about very small patterns in time. The proposed transformation can deal with this problem by adding a sliding window mechanism that allows us to extract the desired features from shorter subsequences.
- The second limitation is related to the no bijective behavior of the proposed transformation, so we cannot recover the original time series from the transformed feature set. Although it is not necessary for the proposed classification model, it may be of interest to specific applications. Achieving a set of features that allows a bijective transformation between the time domain representation and the features while maintaining the interpretability of the proposed set is a long-term goal to keep in mind.

The proposed representation is especially efficient in cases where long time series are processed because it reduces them to a representation with a limited length, making subsequent procedures more efficient.

4. Empirical study

In this section, we evaluate the performance of our proposal. To do this, we first show the experimental design carried out (Section 4.1). Next, the performance results of our proposal against the main algorithms and on the state-of-the-art reference datasets are shown (Section 4.2.1). Finally, the interpretability analysis on two different applications and over the reference 112 datasets are performed (Section 4.2.2).

4.1. Experimental design

In this section, we show the measures used to evaluate the performance of our proposal, the datasets processed, the classification models selected and the hardware used in the experimentation.

The source code of our proposal and experimentation has been developed in R 3.4.4 and can be found in the online repository.[3]

4.1.1. Performance measures

We have chosen accuracy as a basic measure of performance. Accuracy is calculated as the number of correctly classified instances in the test set divided by the total number of cases in the test set. We use the average rank to compare the performance of the different models against each other. Having over a large number of datasets, very different from each other, a relative performance measure like the rank is one of the best options to make the desired comparison. Since the results can vary greatly from one dataset to another we have chosen to use the Critical Difference diagram (CD) (Demšar, 2006). CD allows making a comparison of results, between the different models, from a statistical point of view. In this diagram, the models linked by a bold line can be considered to have no statistically significant differences in their results at a given confidence level α. In this paper, we have chosen a 95% confidence level setting an α of 0.05. We have used the R *scmamp* package to calculate average rank and the CD. In addition, we include the Win/Loss/Tie ratio to be able to observe in a direct quantitative way the performance of each model in comparison with the rest.

4.1.2. Datasets

The used datasets have been extracted from the UCR repository (Dau et al., 2019), which is the reference repository in the field of TSC. It is composed of 128 datasets. The authors of the repository have run the main algorithms of the state-of-the-art of TSC on 112 of the 128 datasets. They eliminated 15 datasets because of containing time series of different lengths and the Fungi dataset because it contains only one instance per class in the training data. Given the great number of algorithms run on these datasets, we can consider the 112 selected datasets as the state-of-the-art in TSC datasets.

4.1.3. Models

The main tree classification algorithms have been selected based on their interpretability: C5.0, C5.0 with boosting (C5.0B) (Quinlan, 1993), Rpart (Therneau et al., 1997), and Ctree (Hothorn et al., 2015). 1NN+ED, 1NN+DTW(w=100), and 1NN+DTW(w_learned) applied over the original time series have been included as benchmark methods since they are the benchmark TSC methods. Although, they do not provide interpretable results beyond how similar two time series are to each other. The new representation of time series that we propose in this work offers an additional information about these series that can also be used by less interpretable algorithms to improve the obtained results. For this purpose, classification algorithms with greater complexity and better accuracy performance have been selected like RF (Breiman, 2001), and SVM (Cortes & Vapnik, 1995). We have also added 1NN+ED applied to the proposed features as a benchmark method. We name the models based on the features proposed in this work following the CMFTS+Model pattern, for example, CMFTS+RF, CMFTS+C5.0, etc.

In order to evaluate our proposal, we have selected only the main algorithms of the state-of-the-art that have been run on the 112 datasets previously mentioned. The algorithms selected are: HIVE-COTE, STC, ResNet, WEASEL, BOSS, cBOSS, c-RISE, TSF, and Catch22. We do not include the model FEARS because there are not public results over the 112 selected datasets, and we were not able to reproduce the results of the original work.

Table 4
Comparative results of the proposed feature-based models (CMFTS) and the TSC benchmark models. The best results are stressed in bold.

Model	Average acc.	Average rank	W/L/T ratio
CMFTS+C5.0	0.724	6.442	3/109/2
CMFTS+C5.0B	0.766	4.263	12/100/3
CMFTS+Rpart	0.682	7.071	4/108/1
CMFTS+Ctree	0.652	7.683	4/108/2
CMFTS+RF	**0.807**	**2.567**	**48/64/4**
CMFTS+SVM	0.764	4.21	14/98/5
CMFTS+1NN-ED	0.737	5.996	8/104/4
1NN-ED	0.694	6.388	9/103/9
1NN-DTW (learned_w)	0.752	4.71	23/89/11
1NN-DTW (w=100)	0.73	5.67	16/96/5

4.1.4. Hardware

For our experiments, we have used a server with the following characteristics: 4 × Intel(R) Xeon(R) CPU E5-4620 0 @ 2.20 GHz processors, 8 cores per processor with HyperThreading, 10 TB HDD, 512 GB RAM. We have used the following software configuration: Ubuntu 18.04, R 3.6.3.

4.2. Results

In this section, we show and evaluate the results obtained by our proposal both in terms of performance (Section 4.2.1) and interpretability (Section 4.2.2). Since the complete empirical results are too extensive to include in the paper, we have put just a summary. The complete set is available at web resource[4] associated to this work.

4.2.1. Performance results

Table 4 shows the results obtained for the 112 datasets processed. We show the average accuracy, average rank, and Win/Loss/Tie Ratio, for all the feature-based learning models (CMFTS) proposed in this paper and the benchmark models in TSC.

If we look at the results of the average rank, Table 4, we see that the CMFTS+RF model obtains the best results, followed by CMFTS+SVM, CMFTS+C5.0B, and 1NN-DTW (learned_w). This shows that more complex models such as RF, C5.0B, SVM, and 1NN-DTW (learned_w) offer better results than more simple models such as C5.0, Rpart, and Ctree. This behavior is also visible in the Win/Loss/Tie Ratio, where CMFTS+RF is the best model, with 48 wins, followed by 1NN-DTW (learned_w) with 23 wins. The third, fourth and fifth places are taken by 1NNN-DTW (w=100) (16 wins), CMFTS+SVM (14 wins), and CMFTS+C5.0B (12 wins), respectively.

In order to make a statistically robust comparison between the different models, we used the CD shown in Fig. 2, with a confidence level of 95%. The CD diagram shows that there is no statistical relationship between CMFTS+RF and the other models, being CMFTS+RF the model most interesting of the tested set. We also see how there are no statistically significant differences between the CMFTS+SVM, CMFTS+C5.0B, and 1NN-DTW (learned_w) models, being the CMFTS+C5.0B model the one with a higher degree of interpretability. Those results allow us to aspire to have interpretable models with competitive results. Finally, we see how CMFTS+1NN-ED slightly improves the results of its direct competitor 1NN-ED and the remaining of the tree-based models (C5.0, Rpart, and Ctree). But the differences are not significant from a statistical point of view.

Once the best models of our proposal have been identified, we will compare them with the best models of the state-of-the-art. The best models of our proposal selected for this comparison are CMFTS+RF, CMFTS+SVM, CMFTS+C5.0B, and CMFTS+1NN-ED. CMFTS+RF and

[3] Complexity Measures and Features for Times Series classification. https://github.com/fjbaldan/CMFTS/.

[4] Complexity Measures and Features for Times Series classification. http://dicits.ugr.es/papers/CMFTS/.

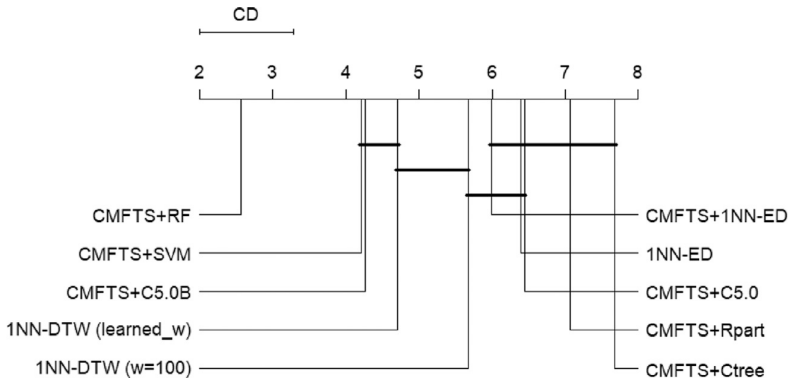


Fig. 2. Critical Difference diagram between the proposed feature-based models (CMFTS) and the TSC benchmark models, confidence level of 95%.

Table 5
Comparative results of the proposed feature-based models (CMFTS) and the TSC state-of-the-art models. The best results are stressed in bold.

Model	Average acc.	Average rank	W/L/T ratio
CMFTS+RF	0.807	7.531	10/102/5
CMFTS+SVM	0.764	9.871	5/107/2
CMFTS+C5.0B	0.766	10.321	1/111/0
CMFTS+1NN-ED	0.737	12.116	4/108/4
BOSS	0.815	7.58	12/100/12
Catch22	0.769	10.353	3/109/2
cBOSS	0.818	7.29	15/97/13
c-RISE	0.79	8.156	7/105/5
HIVE-COTE	**0.864**	**3.17**	**42/70/17**
ResNet	0.82	5.866	33/79/9
STC	0.845	5.308	18/94/9
TSF	0.786	7.741	9/103/7
WEASEL	0.834	5.603	18/94/12
1NN-ED	0.694	12.955	1/111/1
1NN-DTW (learned_w)	0.752	10.473	4/108/3
1NN-DTW (w=100)	0.73	11.665	8/104/5

CMFTS+SVM are the models that obtain the best results, although their interpretability is reduced. CMFTS+C5.0B is the most interpretable model with the best results if we compare it with the rest of the tree-based models. CMFTS+1NN-ED is a simple model that we can use as a benchmark. As in the previous case, for a first analysis, we use a table with the results of average accuracy, average rank, and Win/Loss/Tie Ratio, Table 5. In addition, to carry out an analysis from a statistical point of view we use the CD, Fig. 3.

In Table 5, we see the HIVE-COTE algorithm has the best results in average rank, average accuracy, and win/loss/tie ratio. This algorithm should be used whenever possible. STC is the second method with the lowest average rank and higher average accuracy, but the third in the win/loss/tie ratio. STC can obtain good results in a great number of cases, but not the best results. This behavior indicates that STC offers competitive and robust results in different fields. WEASEL has a behavior very similar to STC. It is the third method in the average rank results, and it has a win/loss/tie ratio and average accuracy results lower but very close to the STC results. For the same win/loss ratio values, WEASEL obtains a higher number of ties than STC. Both methods offer a good start point. ResNet is the fourth method in average rank, but the second one on the win/loss/tie ratio. This behavior indicates that it works better in certain cases, obtaining the best results in a higher number of cases in comparison with STC and WEASEL. In another way, ResNet has worse average performance. If we analyze our proposals, we could observe that CMFTS+RF offers the best results on the average rank, win/loss/tie ratio, and average accuracy.

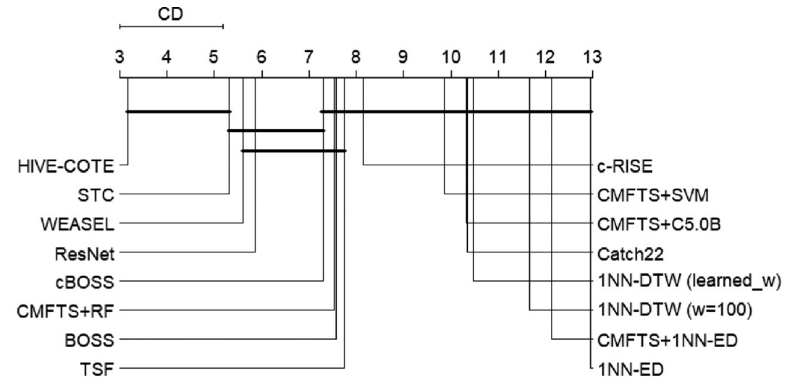
If we analyze Fig. 3, we can observe statistical relationships between our proposal CMFTS+RF and the algorithms HIVE-COTE, STC, and WEASEL, with some conditions. In Fig. 3(a), there are four principal subgroups of proposals without statistical differences between their results over the 112 selected datasets. In this case, HIVE-COTE and STC compose the group with the best results. We can observe that the last group is composed of twelve proposals, which is an interesting behavior. We see how CMFTS proposals are included in this group, but CMFTS+RF is included in another group where its results do not differ statistically from those obtained by WEASEL. If we increase the minimum number of instances per dataset, the observed subgroups can vary significantly. Normally, the feature-based approach performs worse in datasets with a low number of instances. In Fig. 3(b), using datasets with 100 instances or more, we have five different subgroups of models. Now, the best group is composed of HIVE-COTE, STC, and WEASEL. In this case, we see how the results of our best proposal, CMFTS+RF, have not statistical differences with STC and WEASEL models, which are included in the first group. In Fig. 3(c), using datasets with 500 instances or more, we see how the results of CMFTS+RF have not statistical differences with the best model, HIVE-COTE, since CMFTS+RF has been included in the first group. In this case, we can see how WEASEL is the second best model. Those results support the idea that the number of instances affects the results of the feature-based methods.

4.2.2. Interpretability

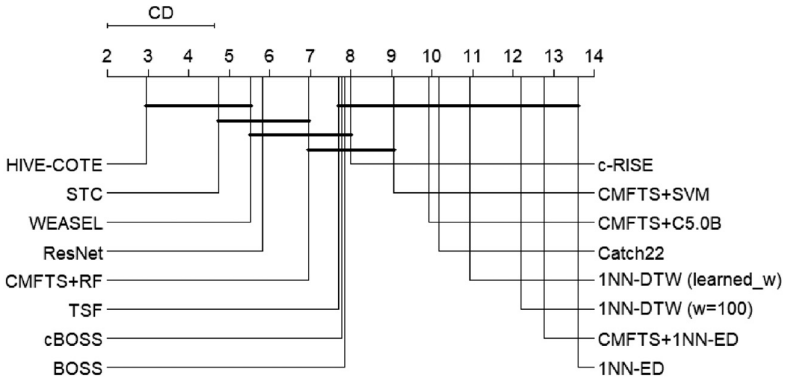
In this section, we analyze the results' interpretability of our proposal over the GunPoint and ECG200 datasets. We also see the advantages of our proposal in terms of the robustness of the results over the entire UCR repository.

Beginning with the *GunPoint* dataset, in Fig. 4(a), we show an example of each of the classes present in this problem. It is a problem that differentiates whether a person has a weapon in his hands or not. The time series that compose this problem comes from the center of mass of the right hand of the person holding or not a weapon. Visually it is appreciated that, in the case of having a gun, the peak present in this temporal series is more pronounced than in the case of not having it.

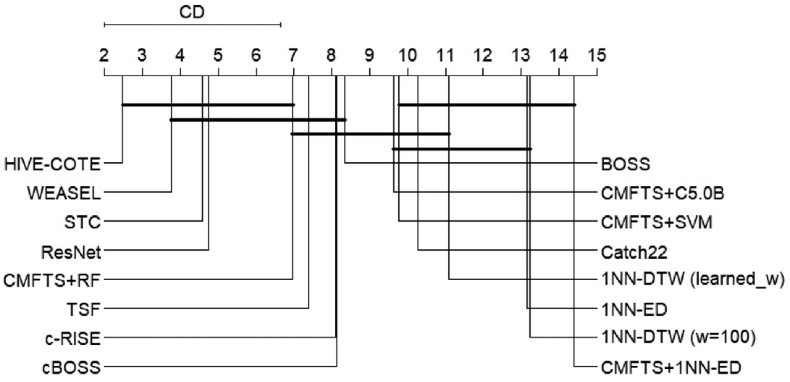
In Fig. 4(b), we see the first classification tree C5.0 obtained by our proposal, CMFTS+C5B. In this tree, we observe how two features like the *stability*, as the variance of the means obtained from tiled windows, and the *shannon_entropy_sg*, with Bayesian estimates of the bin frequencies using the Dirichlet-multinomial pseudo-counting model, can differentiate a large part of the cases that belong to a class. At a glance, we can see that the time series of Class 1 contains more variations in its values than Class 2. Thus we can relate Class 2 with time series values more stables. If we analyze the tree in Fig. 4(b), we can see that high values for the stability feature (node 3) classify the time series as belonging to class 2. On the other hand, high values of the *shannon_entropy_sg* classify the time series as belonging to class 1 (node 1), relating a higher variability in its values to this class. This



(a) Full UCR repository, 112 datasets.



(b) Datasets with 100 or more instances, 76 datasets.



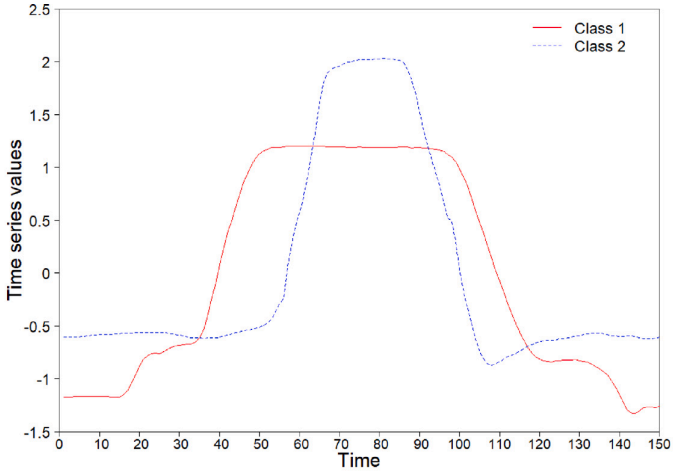
(c) Datasets with 500 or more instances, 25 datasets.

Fig. 3. Critical Difference diagrams between the proposed feature-based models (CMFTS) and the TSC state-of-the-art models, confidence level of 95%. Different scenarios.

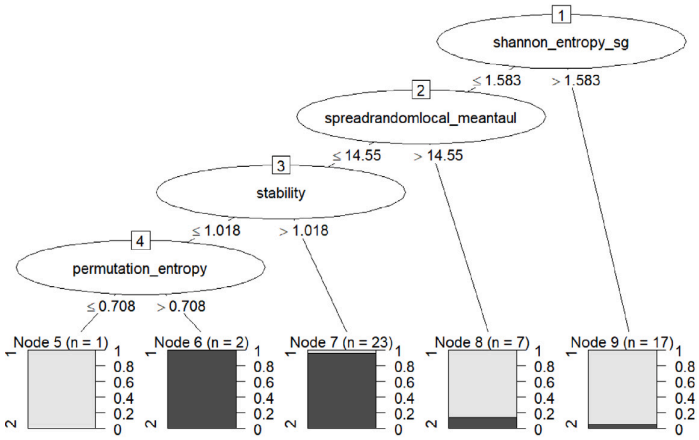
interpretability of the results is provided by the union of the knowledge of the problem, the interpretable classifier used, and the proposed features, which improve the interpretability of this classifier. If we compare these results with Fig. 4(c), where the values of some instants of time are the ones that determine if a case belongs to different classes, we can see how our proposal offers a robust behavior to problems as simple as the desynchronization of the temporal series.

Next, in Fig. 5, we analyze the interpretability results on the problem *ECG200*. Fig. 5(a) shows an example of each of the classes that compose this TSC problem. This problem tries to differentiate between two types of heartbeats: a normal heartbeat and a Myocardial Infarction. For this purpose, the electrical activity associated with a heartbeat has been recorded.

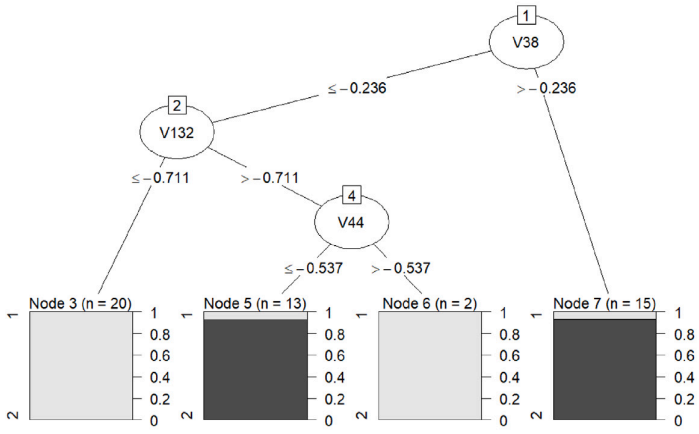
In Fig. 5(b), we see that the feature *spike* can differentiate the time series from class 1 easily. If we compare both classes in Fig. 5(a), we can see that the time series corresponding to class 1 has a large number of peaks over time and even contains a large peak that allows us to differentiate it from class −1 (node 4). On the other hand, we can appreciate more clearly the *curvature* in the time series of class −1 (node 1), where we do not have multiple spikes over time. Class



(a) GunPoint classes example.



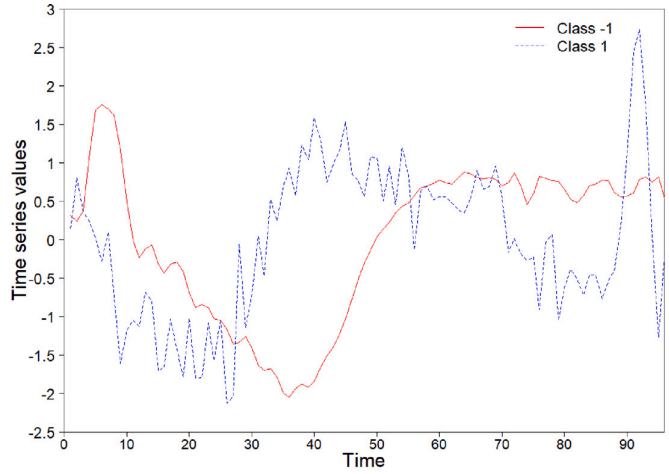
(b) GunPoint example, first C5.0B tree with time series measures.



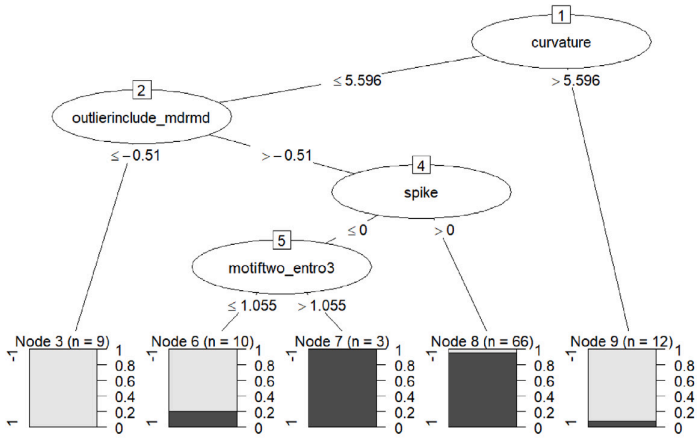
(c) GunPoint example, first C5.0B tree with time series original values.

Fig. 4. Interpretability GunPoint dataset example.

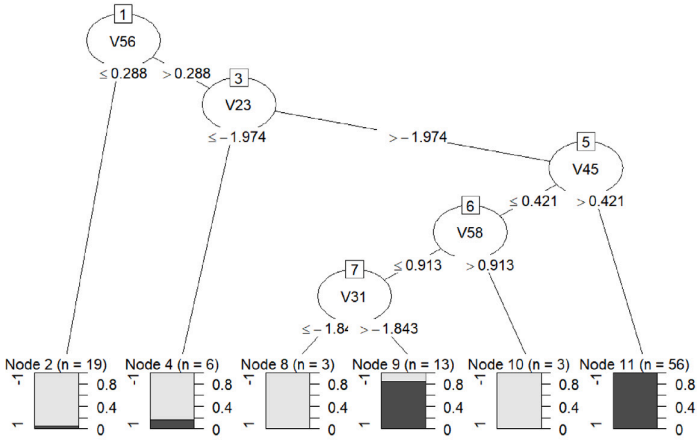
−1 shows a cleaner and more stable behavior, without significant variations in short periods. This behavior coincides with the expressed by the *outlierinclude_mdrmd* and *motiftwo_entro3* features. Higher variability and extreme values will make the *outlierinclude_mdrmd* feature higher, so low values would correspond to time series characteristics of class −1 (node 2). The same happens with the feature *motiftwo_entro3*, which would be related to entropy and would identify the most stable time series, with lower entropy, as belonging to class −1 (node 5). In



(a) ECG200 classes example.



(b) ECG200 example, first C5.0B tree with time series measures.



(c) ECG200 example, first C5.0B tree with time series original values.

Fig. 5. Interpretability ECG200 dataset example.

this way, we can associate the results with the interpretable features proposed. First, we found smooth and stable behaviors in the electrical records for the normal heartbeats. Second, the Myocardial Infarction condition has a clear spikiest and non-stable behaviors in the electrical records. Finally, as in the previous dataset, the results obtained by this model on the original values of the time series are dependent on the synchronization between the different time series and do not allow extracting additional information from the problem beyond a specific

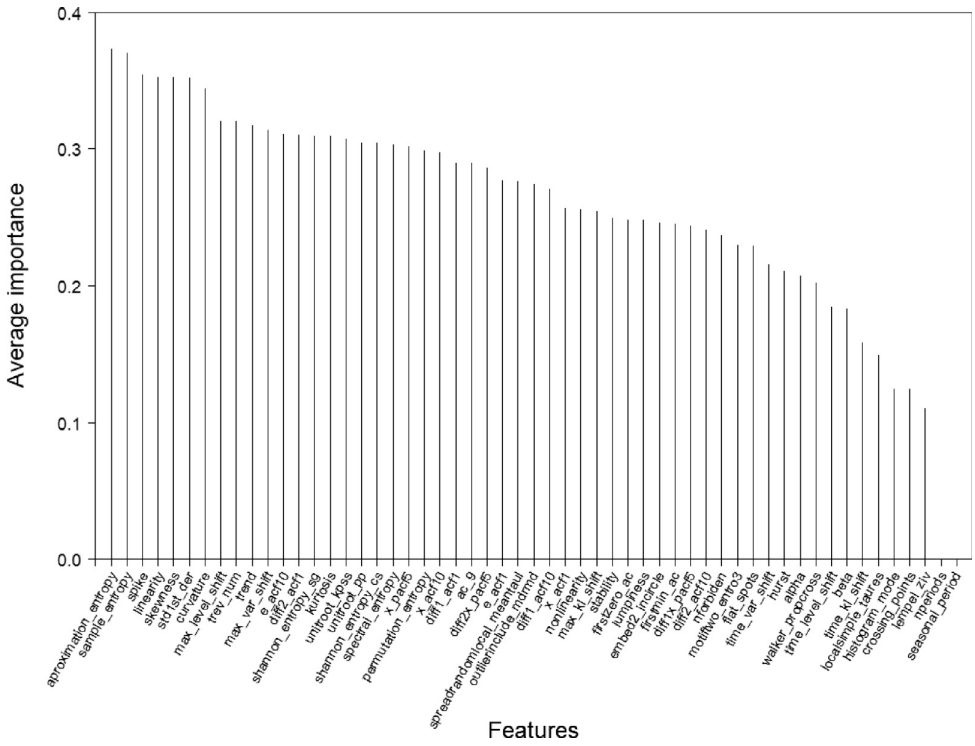


Fig. 6. Average importance of features above all datasets.

value of the time series at a given time. This limitation of the traditional approach can be clearly seen in Fig. 5(c).

Finally, we analyze the conclusions drawn about the complete *UCR repository*. The interpretability of the results is strongly linked to the importance given by each algorithm to each of the input features, whenever it is possible. For this reason, we have selected our best proposal, CMFTS+RF, that measures the importance of each feature through the Gini Index (Ceriani & Verme, 2012). We have analyzed the accumulated importance of each feature over the 112 datasets and the importance of each feature in each dataset.

Fig. 6 shows the mean results of the importance of the features obtained on the 112 datasets used. The features have been sorted in decreasing order based on their average importance to facilitate the analysis. We see how features related to entropy, such as *sample_entropy* and *approximation_entropy* achieve the highest valuation in importance. Interpretable features such as *linearity*, *curvature*, *spike*, *std1st_der*, and *skewness* would occupy the following positions of importance. On the other hand, we can see two features that have zero importance: *nperiods* and *seasonal_period*. Due to the high number of datasets, a no-preprocessing of the data approach has been chosen, specifying a zero frequency for every time series. This causes the calculation of *nperiods* and *seasonal_periods* to always get the same value. In a real case, different parameters can be specified that would allow different values to be obtained in these features. The previous features are especially interesting in the field of time series, so we have decided to keep them in the CMFTS package.

We use a heat map to be able to analyze the importance of each feature on each dataset, Fig. 7. In these cases, we also sorted the features in decreasing order based on their average importance to facilitate comparison with previous importance analyses. As we can see in Fig. 7, there are a lot of differences in the feature importance scores between different datasets. It means that each problem has very several characteristics and behaviors, so we need different features to extract the right information on each dataset. We can differentiate into two big groups of datasets. The first one which we need a small number of features to obtain the desired information. So, our models can obtain good enough results with this small subset of features, even if these results are not the best. The second one which our model uses a lot of features. In this case, it might be because the problem is very complex, and we need a lot of information to obtain good results. Or the features are not good enough to obtain the needed information to resolve the problem, and the model uses a lot of them trying to obtain good results. If we sort the datasets from Fig. 7 in an increasing way based on the accumulated importance of the features, we can observe both groups in an easy way, Fig. 8. At the top of the heat map, we can see the datasets in which our model uses a small set of features. At the bottom, we are able to see the datasets in which our proposal needs to use a lot of features. On the datasets in the order of Fig. 8, if we calculate the difference between the best case of each dataset and our best model (CMFTS+RF), Fig. 9, we see that this difference is lesser in the datasets at the top of Fig. 8. That means that in the cases in which our model uses a small subset of features, it is able to obtain very close results to the best algorithm. Additionally, we can infer a more efficient problem modeling when a reduced set of important features is obtained. This characteristic would indicate that the trees obtained would be centered on a reduced set of features, and simpler trees would be obtained than in the case of using the whole set of available measures. These results reinforce the original idea of this proposal to obtain competitive results with simple and interpretable models.

Once the performance and interpretability results have been analyzed, we can discuss the advances of our proposal related to time series models in different applications. If we compare our proposal with the problems included in the UCR repository, we can see that it is applicable, without additional considerations, over the entire repository. In this work, we only use the 112 datasets with equal-length time series for comparison because the main algorithms of the state-of-the-art have this limitation. We also note the high computational complexity of proposals like HIVE-COTE, $O(n^2m^4)$, which is significantly higher than our proposal (CMFTS). Finally, if we compare the interpretability between HIVE-COTE, deep learning models, and CMFTS, we can see that

Fig. 7. Heat map of the importance of features by dataset.

Fig. 8. Heat map of the importance of features by dataset, sorted by accumulated importance.

the interpretability of HIVE-COTE, which is an ensemble of multiple classifiers, and the deep learning models, which are black boxes, is near 0, whereas the CMFTS proposal provides simple trees or useful information about the variables that compose the problem.

If we analyze the use of CMFTS on new time series applications (Zhou & Liu, 2021), its incorporation is quite straightforward. The original proposal already includes a sliding-window mechanism, so our proposal can be applied without additional considerations. The

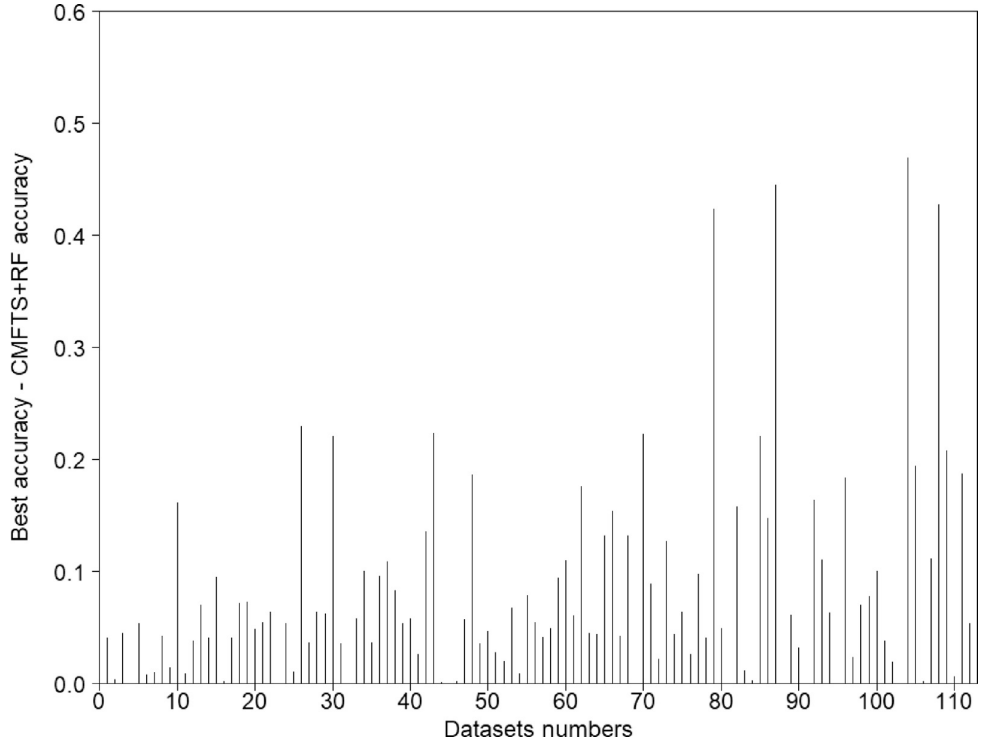


Fig. 9. Accuracy differences between CMFTS+RF and the best algorithm on each dataset. The datasets are sorted like Fig. 8.

equal-length limitation or the high complexity of the main proposals of the time series classification state-of-the-art could increase the needed considerations to their application or make it unviable directly. The additional information about the variables that compose the problem provided by CMFTS can be useful to improve the performance of the algorithm that obtains the best possible results, particularly in a problem involving a convolutional long short-term memory (CLSTM) neural network.

5. Conclusion

In this work, we have presented a set of features, composed of measures of complexity and representative features of time series, capable of extracting important information from the time series on which they are applied. The proposed set of features makes it possible to tackle TSC problems with traditional classification algorithms, allowing them to improve the interpretability of models.

We have published our proposal software to make it accessible and usable for any practitioner or researcher to use. We have published all the results obtained throughout the work to make it fully reproducible. The functioning of our proposal has been tested on 112 datasets obtained from the UCR repository. We have used tree-based classification algorithms due to their high interpretability, and they have been compared with the state-of-the-art TSC algorithms. The results obtained by our proposal have not statistical differences with the third best algorithm of the state-of-the-art of TSC, with a confidence level of 95%. If we focus our analysis on datasets with more than 500 time series, our proposal obtains results statistically indistinguishable from those obtained by the best state-of-the-art algorithm. This result reinforces the original idea that feature-based methods require a larger number of time series to perform correctly.

Extracting features of interest from time series that are robust and interpretable provides more understandable and even better classification results in some cases. Our proposal demonstrates a robust behavior against typical TSC problems by extracting descriptive features from the time series rather than working on the original series itself. In this way, additional interpretability is achieved, which is especially useful in some problems.

The time series representation introduced in this paper has been applied to time series classification, but it can be extended to other tasks in time series. This suggests several future lines of research: time series clustering based on the feature-based approach, analysis of new time series features that could improve the performance and interpretability of the proposed set, application on multivariate time series problems or data massive environments, analysis of transfer learning scenarios based on close feature-sets, among others.

CRediT authorship contribution statement

Francisco J. Baldán: Conceptualization, Formal analysis, Data curation, Methodology, Software, Investigation, Writing – original draft. **José M. Benítez:** Conceptualization, Validation, Supervision, Writing – review & editing, Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The link to source code and experimentation is given in the paper.

Acknowledgments

This paper was partially supported by the following grants: TIN2016-81113-R and PID2020-118224RB-I00 from the Spanish Ministry of Economy and Competitiveness, and P12-TIC-2958, P18-TP-5168, and A-TIC-388-UGR-18 from the Andalusian Regional Government, Spain. Francisco J. Baldán holds the FPI grant BES-2017-080137 from the Spanish Ministry of Economy and Competitiveness.

We do warm fully acknowledge insightful comments from Rob. J. Hyndman on previous versions of this paper that without doubt greatly contributed to the enhancement of this paper.

The authors would like to thank Prof. Eamonn Keogh and all the people who have contributed to the UCR TSC archive for their selfless work.

References

Abdiansah, A., & Wardoyo, R. (2015). Time complexity analysis of support vector machines (SVM) in LibSVM. *International Journal Computer and Application*, *128*(3), 28–34.

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE Access*, *6*, 52138–52160.

Amigó, J. (2010). *Permutation complexity in dynamical systems: ordinal patterns, permutation entropy and all that*. Springer Science & Business Media.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., et al. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115.

Bagnall, A., Davis, L., Hills, J., & Lines, J. (2012). Transformation based ensembles for time series classification. In *Proceedings of the 2012 SIAM international conference on data mining* (pp. 307–318). SIAM.

Bagnall, A., Flynn, M., Large, J., Lines, J., & Middlehurst, M. (2020). On the Usage and Performance of The Hierarchical Vote Collective of Transformation-based Ensembles version 1.0 (HIVE-COTE 1.0). arXiv preprint arXiv:2004.06069.

Bagnall, A., & Janacek, G. (2014). A run length transformation for discriminating between auto regressive time series. *Journal of Classification*, *31*(2), 154–178.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606–660.

Baldán, F. J., & Benítez, J. M. (2019). Distributed FastShapelet Transform: a Big Data time series classification algorithm. *Information Sciences*, *496*, 451–463.

Baldán, F. J., Ramírez-Gallego, S., Bergmeir, C., Herrera, F., & Benítez, J. M. (2018). A Forecasting Methodology for Workload Forecasting in Cloud Systems. *IEEE Transactions on Cloud Computing*, *6*(4), 929–941.

Bandt, C., & Pompe, B. (2002). Permutation entropy: a natural complexity measure for time series. *Physical Review Letters*, *88*(17), Article 174102.

Bengio, Y., Yao, L., Alain, G., & Vincent, P. (2013). Generalized denoising auto-encoders as generative models. In *Advances in neural information processing systems* (pp. 899–907).

Berndt, D. J., & Clifford, J. (1994). Using Dynamic Time Warping to Find Patterns in Time Series. In *Proceedings of the 3rd International conference on knowledge discovery and data mining* (pp. 359–370). AAAI Press.

Bostrom, A., & Bagnall, A. (2017). Binary shapelet transform for multiclass time series classification. In *Transactions on large-scale data-and knowledge-centered systems XXXII* (pp. 24–46). Springer.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32.

Ceriani, L., & Verme, P. (2012). The origins of the Gini index: extracts from Variabilità e Mutabilità (1912) by Corrado Gini. *The Journal of Economic Inequality*, *10*(3), 421–443.

Chao, A., & Shen, T.-J. (2003). Nonparametric estimation of Shannon's index of diversity when there are unseen species in sample. *Environmental and Ecological Statistics*, *10*(4), 429–443.

Chauhan, S., Vig, L., & Ahmad, S. (2019). ECG anomaly class identification using LSTM and error profile modeling. *Computers in Biology and Medicine*, *109*, 14–21.

Chen, H., Tang, F., Tino, P., & Yao, X. (2013). Model-based kernel for efficient time series analysis. In *Proceedings of the 19th ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 392–400). ACM.

Čisar, P., & Čisar, S. M. (2010). Skewness and kurtosis in function of selection of network traffic distribution. *Acta Polytechnica Hungarica*, *7*(2), 95–106.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., et al. (2019). The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, *6*(6), 1293–1305.

DeCarlo, L. T. (1997). On the meaning and use of kurtosis. *Psychological Methods*, *2*(3), 292.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*(Jan), 1–30.

Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, *239*, 142–153.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963.

Fulcher, B. D. (2018). Feature-based time-series analysis. In *Feature engineering for machine learning and data analytics* (pp. 87–116). CRC Press.

Fulcher, B. D., & Jones, N. S. (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, *26*(12), 3026–3037.

Fulcher, B. D., Little, M. A., & Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of the Royal Society Interface*, *10*(83), Article 20130048.

Hatami, N., Gavet, Y., & Debayle, J. (2018). Classification of time-series images using deep convolutional neural networks. In *Tenth international conference on machine vision, vol. 10696* (p. 106960Y). International Society for Optics and Photonics.

Hothorn, T., Hornik, K., & Zeileis, A. (2015). ctree: Conditional inference trees. *The Comprehensive R Archive Network*, 1–34.

Kang, Y., Hyndman, R. J., Li, F., et al. (2018). *Efficient generation of time series with diverse and controllable characteristics*: *Technical Report*, Monash University, Department of Econometrics and Business Statistics.

Khadiev, K., Mannapov, I., & Safina, L. (2019). The Quantum Version Of Classification Decision Tree Constructing Algorithm C5. 0. arXiv preprint arXiv:1907.06840.

Lempel, A., & Ziv, J. (1976). On the complexity of finite sequences. *IEEE Transactions on Information Theory*, *22*(1), 75–81.

Lin, J., & Li, Y. (2009). Finding structural similarity in time series data using bag-of-patterns representation. In *International conference on scientific and statistical database management* (pp. 461–477). Springer.

Lines, J., Davis, L. M., Hills, J., & Bagnall, A. (2012). A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 289–297). ACM.

Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., & Jones, N. S. (2019). Catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, *33*(6), 1821–1852.

Manis, G. (2008). Fast computation of approximate entropy. *Computer Methods and Programs in Biomedicine*, *91*(1), 48–54.

Manis, G., Aktaruzzaman, M., & Sassi, R. (2018). Low computational cost for sample entropy. *Entropy*, *20*(1), 61.

Markovič, R., Gosak, M., Grubelnik, V., Marhl, M., & Virtič, P. (2019). Data-driven classification of residential energy consumption patterns by means of functional connectivity networks. *Applied Energy*, *242*, 506–515.

Mueen, A., Keogh, E., & Young, N. (2011). Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 1154–1162). ACM.

Nanopoulos, A., Alcock, R., & Manolopoulos, Y. (2001). Feature-based classification of time-series data. *International Journal of Computer Research*, *10*(3), 49–61.

Nweke, H. F., Teh, Y. W., Al-Garadi, M. A., & Alo, U. R. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, *105*, 233–261.

Pincus, S. M. (1991). Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences*, *88*(6), 2297–2301.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Rakthanmanon, T., & Keogh, E. (2013). Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proceedings of the 2013 SIAM International conference on data mining* (pp. 668–676). SIAM.

Ratanamahatana, C. A., & Keogh, E. (2004). Making time-series classification more accurate using learned constraints. In *Proceedings of the 2004 SIAM international conference on data mining* (pp. 11–22). SIAM.

Richman, J. S., & Moorman, J. R. (2000). Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, *278*(6), H2039–H2049.

Schäfer, P. (2015). The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, *29*(6), 1505–1530.

Schlegel, U., Arnout, H., El-Assady, M., Oelke, D., & Keim, D. A. (2019). Towards a rigorous evaluation of xai methods on time series. In *2019 IEEE/CVF international conference on computer vision workshop* (pp. 4197–4201). IEEE.

Schürmann, T., & Grassberger, P. (1996). Entropy estimation of symbol sequences. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *6*(3), 414–427.

Therneau, T. M., Atkinson, E. J., et al. (1997). An introduction to recursive partitioning using the RPART routines. Technical Report 61. URL http://www.mayo.edu/hsr/techrpt/61.pdf.

Twomey, N., Chen, H., Diethe, T., & Flach, P. (2019). An application of hierarchical Gaussian processes to the detection of anomalies in star light curves. *Neurocomputing*, *342*, 152–163.

Wang, Z., & Oates, T. (2015). Spatially encoding temporal correlations to classify temporal data using convolutional neural networks. arXiv preprint arXiv:1509.07481.

Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks* (pp. 1578–1585). IEEE.

Ye, L., & Keogh, E. (2009). Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 947–956). ACM.

Zhang, A., Yang, B., & Huang, L. (2008). Feature Extraction of EEG Signals Using Power Spectral Entropy. In *2008 International conference on biomedical engineering and informatics, vol. 2* (pp. 435–439).

Zhou, K., & Liu, Y. (2021). Early-stage gas identification using convolutional long short-term neural network with sensor array time series data. *Sensors*, *21*(14), 4826.