Ingeniaritza Konputazionala eta
Sistema Adimentsuak Unibertsitate Masterra

Máster Universitario en Ingeniería Computacional
y Sistemas Inteligentes

Konputazio Zientziak eta Adimen Artifiziala Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Master Tesia
Tesis de Máster

An Analysis of the Relevance of Temporal Information in Time Series Classification

Ainhize Barrainkua Aguirre

Zuzendaritza
Dirección

Jose Antonio Lozano Alonso

Basque Center for Applied Mathematics (BCAM)

Intelligent Systems Group (ISG), Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU

Usue Mori Carrascal

Intelligent Systems Group (ISG), Department of Computer Science and Artificial Intelligence

Department of Applied Mathematics, Statistics and Operational Research

University of the Basque Country UPV/EHU

Abstract

In recent years, the interest in time series has increased considerably due to the vast amount of such data collected in a variety of fields. Time series are a particular type of data: they are sets of ordered observations. The analysis of databases composed of time series requires the consideration of the nature of the instances, where there exist a temporal correlation among the observations. A considerable variety of algorithms that take heed of such characteristic of the instances have been developed to represent, index, cluster and classify time series. Particularly, this work focuses on the classification task. Firstly, a review is performed about the specific time series classifiers, to give an insight of their workflow as well as how they capture the temporal information. The different procedures of those classifiers endow them with different abilities to catch the intrinsic temporal information of the instances for classification. Moreover, this work carries out an experiment based on empirical distributions to estimate the sensitivity of specific time series classifiers to the temporal order of the observations. Besides, although in general specific classifiers have been used to classify time series, in some time series classification problems, non-specific classification algorithms have shown to be competitive with the specific ones. Thus the relevance of the temporal order for classification varies for different time series classification problems. The present work aims to develop an analysis based on empirical distributions for estimating the relevance of the temporal ordering in a given time series classification problem, as well as studying the sensitivity of the specific time series classifiers to the temporal correlation of the observations.

Contents

1 Introduction

Time series (TS) are very present in our every day life, starting from the media to the different scientific, technological and financial research areas [1]. They are widely used in statistics, signal processing, econometrics, finances, weather forecasting, medical studies, control engineering etc [2]. When certain data describe the evolution of one or several attributes (variables) in a certain time range, or other ordered dimension, it is said to form a time series. Thus, a time series is a sequence of ordered observations.

There exist a huge variety of algorithms to represent [3], index [4], cluster [5] and classify [6]–[8] time series [9]. Particularly, this work studies the time series classification (TSC) task: given a database where each time series has associated a particular class, the aim is to predict the class of a newly given series.

When dealing with TSC problems for the first time, the first intuitive approach would be to consider the values at each time stamp as belonging to a single attribute and perform the classification in the conventional way. Nonetheless, such approach has shown to fail in several problems, as relevant signals are not necessarily aligned and the information about the temporal correlation of the observations is not considered. Thus considering the temporal order of the instances seems to be crucial in the classification of some problems related to time series. Henceforth, the classification of time series differs from the classical classification learning problems, where the instances are described by attributes without an specific natural ordering.

The particularity regarding the instances represented by a time series, has lead to the assumption that they must be studied by specific methods that take into account the temporal correlation of the attributes. Those methods base the classification process on the intrinsic temporal information of the observations. There exist a huge variety of such specific methods [6]–[8]: algorithms where the classification is based on the distance between complete raw time series, classifiers that transform the original series to a symbolic feature space then train the classifier on the transformed instances, ensembles of simple classifiers, classifiers with decision tree architectures, deep learning approaches, etc. Those specific methods, use different workflows to read the temporal information given by the order of the observations, that will later be used for classification. Depending on their procedure, their ability to learn the intrinsic temporal structures of the instances varies.

As TS classifiers base their predictions on the temporal information given by the instances of the datasets, if the timestamps of the observations are ordered at random and the original temporal information of the instances is lost, the performance of the classifiers should deteriorate. Consequently, when specific TS classifiers are applied to such altered problems, as they base the classification in an artificial temporal order, their performance is expected to vary negatively. That is, even if the recorded values of the observations are the same, when the order in which they are recorded has been changed, the ability of the classifier to assign a class label to an unlabeled time series will not longer be the same.

When the temporal order of the observations is shuffled in a TSC problem, significant shapes, subsequences, structures, patterns etc. are lost. That is, the intrinsic temporal information of the problem is lost. Nevertheless, the alteration of the order of the observations will not have the same repercussion on the performances of all the classifiers, as they consider different procedures to capture the temporal information for classification. Therefore, depending on their internal strategy, the performances of some algorithms might be more robust than that of the others against the loss of the original temporal information of the instances.

On the other hand, the performance of the conventional non specific classifiers will not vary in front of an alteration of the temporal order, as they do not take into account the order in which the values are recorded.

Therefore, it is assumed that as the non-specific methods do not take into account the so mentioned temporal information, cannot outperform, or at least be competitive with the specific methods whenever they are applied to TSC problems.

When the temporal order of the observations is shuffled, losing a considerable part of the intrinsic temporal information should deteriorate the performance of specific algorithms, as such information is considered to be discriminatory for classification. Nonetheless, if in a given TSC classification problem, creating new artificial temporal orderings by randomly altering the order of the series leads to better performances of such algorithms, the temporal order of the instances will not be considered as discriminatory for the classification.

Estimating the relevance of the temporal information for a given TSC problem is of great interest, as in the cases where the order of the instances is not discriminatory for classification conventional non-specific classifiers could also be considered.

One of the principal objectives of this work is to study the sensitivity that the different specific TS classifiers show related to the alteration of the temporal order of the instances; as well as their ability to detect the intrinsic temporal information of the instances. Moreover, this work also aims to examine the relevance of the order of the observations on a TS classification process; that is, to conclude how discriminatory the temporal information is for a given TSC problem. The initial idea was to define a statistical analysis by means of a mathematical expression to estimate the relevance of the temporal order in the classification, using terms such as entropy gain or mutual information. Nonetheless, because of the lack of a definition of the probability distribution regarding the attribute representing the temporal information, the study of this work is based on empirical experiments.

The rest of the work is organized as follows: Section 2 provides a summary of the existing specific time series classification algorithms, and their procedures to capture the temporal information. In Section 3 the sensitivity that specific TS classifiers have in relation to the temporal order of the instances is studied. Section 4 examines the relevance of the intrinsic temporal information for different TSC problems. Section 5 analyses the correlations between different characteristics regarding the TSC problems and specific TS classifiers. Finally, Section 6 sums up the main ideas concluded from this work.

2 Review on Specific Time Series Classifiers

A *time series* is a sequence of ordered observations represented by the value of the measurement and the timestamp it has been observed in [10]:

$$TS = \{(t_i, x_i),\ \ i = 1, ..., N\}$$

where N is the length of the time series. The measured values (x_i) can be univariate or multivariate, depending on the number of attributes measured, taking either discrete or real values.

An unordered collection of such time series forms a *time series database*. The time series that compose a database can have varying lengths.

There is a wide research area dedicated to time series, where diverse analysis are carried out to extract useful information from the temporal data. The most popular objective has been forecasting future values of time series [1]. Nonetheless, considerable work has been done regarding the classification, representation, clustering, segmentation, etc. of time series [9], [11]. In particular, this dissertation addresses the time series classification task.

Time series classification is a supervised data mining task where, given a training time series dataset $\mathbf{TS} = \{TS_1, TS_2, ..., TS_n\}$ and the set of class labels associated to each of them $\mathbf{C} = \{C_1, C_2, ..., C_n\}$, a time series classifier is built. The latter aims to predict the class of a newly given time series (see Figure 1), from a set of predefined classes characterized by the set of observations for training. The classification of time series differs from the classical classification learning problems: time series classifiers must consider the temporal correlation of attributes, while in classical supervised problems the instances are described by attributes without an specific natural ordering.

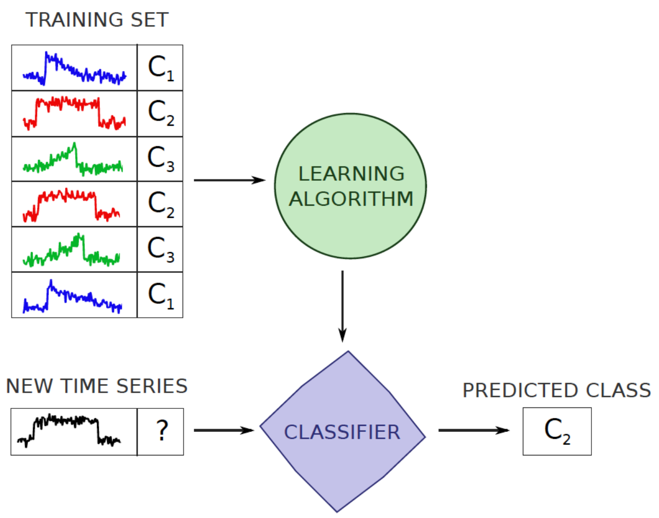


Figure 1: Schematic representation of the time series classification process taken from [10].

There exists a huge variety of specific TS classification algorithms which reckon with the particular nature of the observations, where the attributes are temporally correlated. Some classifiers compute the similarity between raw time series and perform the classification with a distance based machine learning (ML) algorithm. Other approaches extract feature vectors from the real-values time series, then map the original series to the new feature space and perform the classification based on such new space by a conventional

ML classifier. There are also classifiers that follow tree-based structures, ensembles of base classifiers whose individual outcomes are combined to perform a single classification and deep-learning approaches.

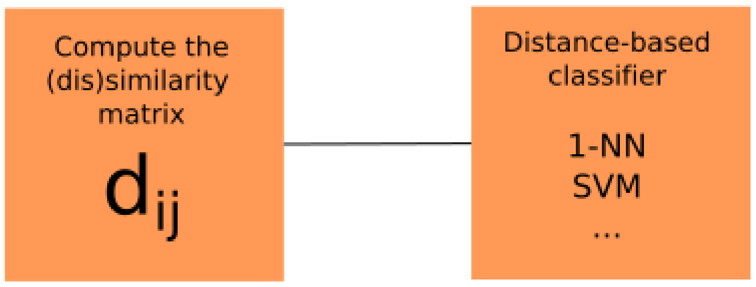
In fact, this section divides the classifiers in 5 principal groups (see Figure 2). In the first one, the algorithms compute a (dis)similarity matrix using a given TS distance measure, and then perform the classification with a distance-based classifier. The workflow of the algorithms in the second group, is to extract features from real-valued time series, and then perform the classification by a distance-based or a feature-based algorithm. Several approaches of this group also perform an internal feature selection process. The third group, presents the so called ensemble algorithms, which as its name suggests, are ensembles of base algorithms, whose outputs are combined for a final single prediction. In the fourth group, those algorithms that follow decision tree architectures are found. Some approaches use the distance between complete real-values time series (as those in the first group) as a splitting criteria, while in other approaches, extracted features are used to branch the data (features extracted by techniques explained in the section related to the second group of algorithms). Lastly, deep learning architectures designed to deal with temporal data are explained.

Note that an algorithm can belong to different groups, as their characteristics may match with several workflows at the same time.

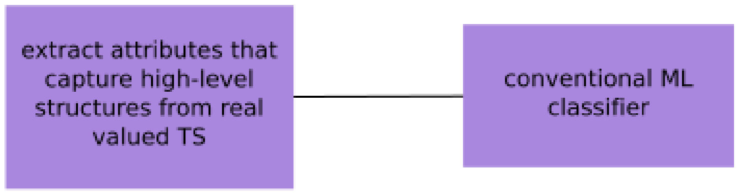
The different existing time series classifiers, follow very distinct procedures to construct models that are able to assign a class label to a query time series. That is, they focus on very different characteristics when performing the classification, learning the intrinsic temporal information of the instances in different ways. In consequence, the alteration of the temporal order of the observations will have different repercussions in the algorithms. In some cases, the performance of the classifier will be very robust against re-ordering of the time stamps of the observations, while other algorithms may show significant variations in their performances when the order is randomly altered, meaning they have high sensitivity to the order in which the observations are recorded.

In all the cases except deep learning approaches, the sensitivity of the algorithm to the temporal information it is not subject to the classifier itself, as they are conventional ML classifiers, but to the information given to such classifier instead. That is, the temporal information is saved in the specific distance measures or extracted attributes. Therefore, the ability of a given algorithm to detect the temporal information of a TSC problem or the performance variation of such algorithm when the original temporal order of the instances is lost, depends on the specific TS distance measures or extracted features.

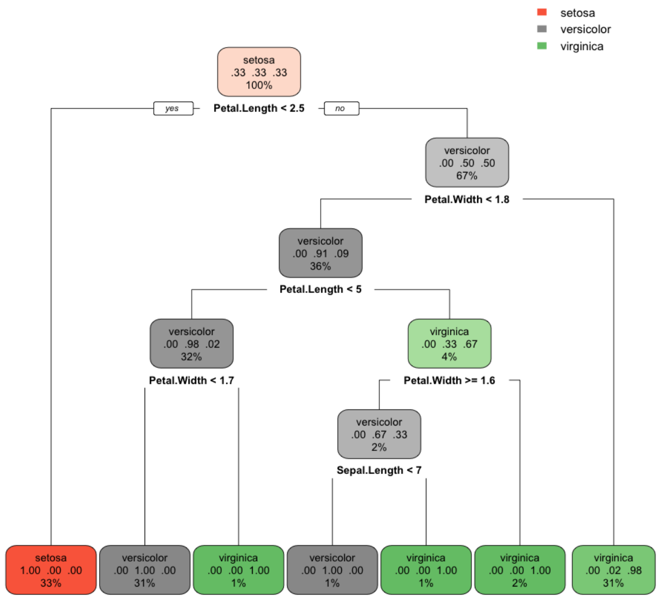
The review of this section does not cover every existing specific algorithm to perform the classification of TS, but it explains the most basic approaches (those in which more complex classifiers are based) and some compounded algorithms that are considered in the empirical study later in this work.


Compute the (dis)similarity matrix
d_{ij}
Distance-based classifier
1-NN
SVM
...

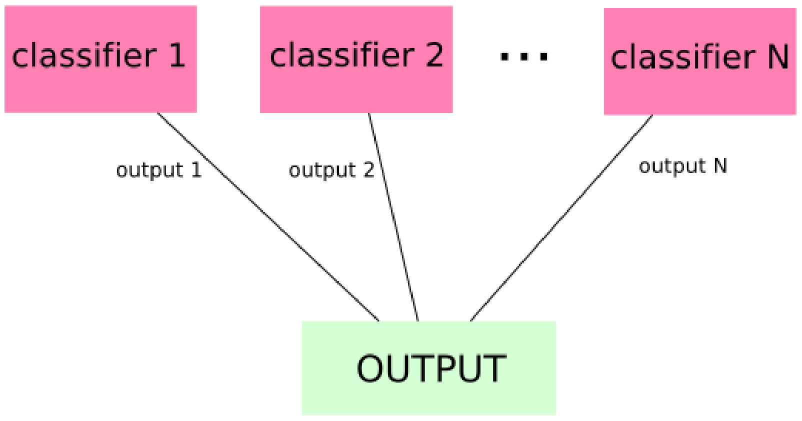
(a)


extract attributes that capture high-level structures from real valued TS
conventional ML classifier

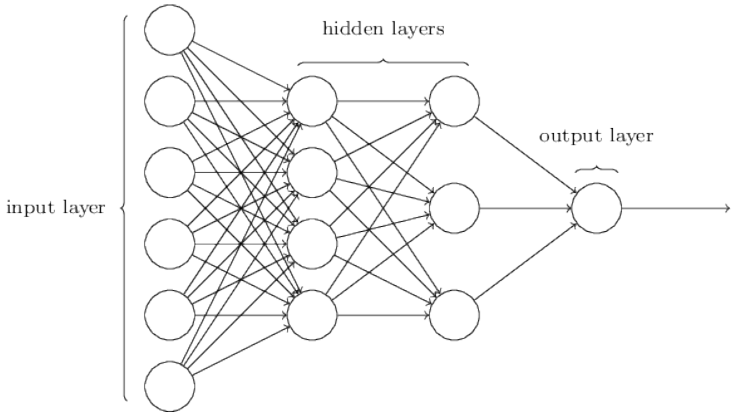
(b)


setosa
versicolor
virginica
setosa .33 .33 .33 100%
yes
Petal.Length < 2.5
no
versicolor .00 .50 .50 67%
Petal.Width < 1.8
versicolor .00 .91 .09 36%
Petal.Length < 5
versicolor .00 .98 .02 32%
Petal.Width < 1.7
virginica .00 .33 .67 4%
Petal.Width >= 1.6
versicolor .00 .67 .33 2%
Sepal.Length < 7
setosa 1.00 .00 .00 33%
versicolor .00 1.00 .00 31%
virginica .00 .00 1.00 1%
versicolor .00 1.00 .00 1%
virginica .00 .00 1.00 1%
virginica .00 .00 1.00 2%
virginica .00 .02 .98 31%

(c)



(d)



(e)

Figure 2: The general workflows of the different specific TS classifiers.

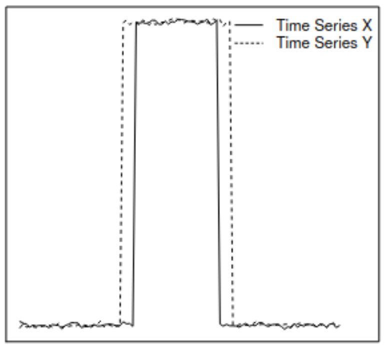
2.1 Similarity Between Whole Series: Shape-Based Similarities

The classifiers grouped in this section, consider shape-based similarities between complete time series, concluded from the individual point values. The particular nature of time series require specific similarity measures. By this time, there exists a considerable variety of distance measures to compute the similarity between two whole series. Those distance measures are used to construct the (dis)similarity matrix by considering all the pair combinations between the series on the training set. Then a distance based classifier is trained based on such matrix. The classification of unlabeled instances relies on the distance of the query to all the time series of the training set. The classification is almost exclusively performed by the nearest neighbour (1-NN) classifier. Nonetheless, other approaches can be used; such as, k-Nearest Neighbour (k-NN) classifier or support vector machines (SVM) [12]. The following lines of this section present some of the existing similarity measures to calculate the similarity of two whole time series, in which the (dis)similarity matrix is based.

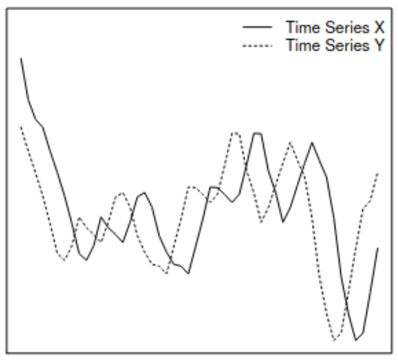
The **Euclidean Distance** (ED) is a shape based distance measure that computes the pointwise distance between two time series. Given two time series $X = \{x_0, x_1, ...x_{N-1}\}$ and $Y = \{y_0, y_1, ...y_{N-1}\}$ the ED between them is computed as follows:

$$ED(X,Y) = \sqrt{\sum_{i=0}^{N-1} (x_i - y_i)^2} \tag{1}$$

Even if the ED is one of the most used measures in data mining, several researchers have outlined it might not be an adequate measure for time series [3], as it is only applicable to series of same length and is considerably susceptible to noise and outliers. Moreover, as it compares the point placed on the same timestamp, it does not represent correctly the distance between two time series in the case of warped or shifted series (see Figure 3), and the alteration of the order of the instances will not affect its performance, as the sum of the distance of the points recorded at the same timestamp remains the same. Even if the order or the points is permuted, the sum will converge to the same value, thus it does not consider the temporal correlation of the observations.



(a)



(b)

Figure 3: Two time series (a) with different local warpings and (b) shifted from each other.

To overcome the problems derived from the ED rigid measure, several elastic distance measures have been proposed. The most simple and popular benchmark elastic distance measure is the **Dynamic Time Warping** (DTW) [13]. Its main purpose is to find the optimal alignment between two series $X = \{x_0, x_1, ...x_{N-1}\}$ and $Y = \{y_0, y_1, ...y_{M-1}\}$ (see Figure 4), by searching the minimal path in a pointwise distance matrix (D) that defines a mapping between them. Each of the elements of the matrix D is the ED between two points (x_i, y_j). The search for the optimal path has several constraints [14]: the path must start in position $D(0,0)$ and end in $D(N-1, M-1)$, the path is forced to continue through adjacent cells and it cannot move backwards in the position of the matrix. Moreover, in some cases an extra temporal restriction is added to limit the number of vertical or horizontal steps that the path can take consecutively, reducing the computational cost by avoiding the match of points that are far from each other [3]. The DTW distance measure allows the calculation of the distance of two time series with different lengths.

The DTW distance measure is sensitive to the temporal order of the instances. Therefore, when the temporal information of a given TSC problem is altered, the alignment between two time series will change, and consequently, so will the distance between them. Therefore, the distance value will no longer be representative of their closeness.

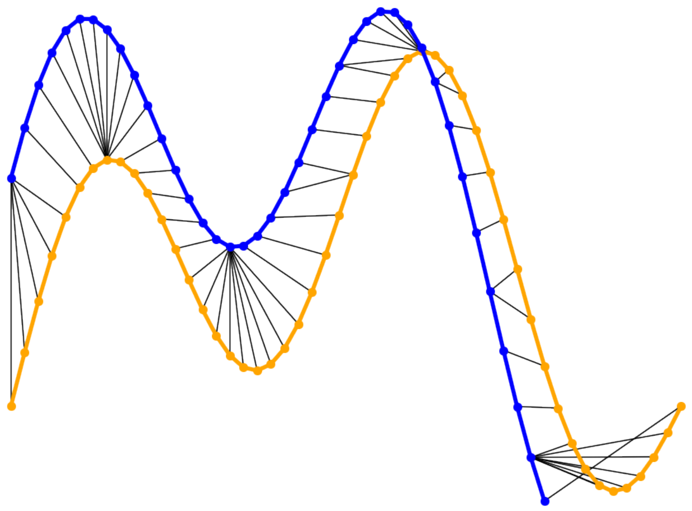


Figure 4: Example of the computation of the DTW distance.

Based on the DTW elastic distance measure, Jong *et al.* [15] describe the **Weighted DTW** (WDTW) measure, which adds a multiplicative weight penalty that depends on the distance between points on the warping path. That is, the elements of the pointwise distance matrix D are multiplied by a weight that depends on their distance on the time domain, applying a bigger penalization to those pairs that are far from each other.

Moreover, Górecki and Luczak [16], [17] proposed the **Derivative DTW** (DD$_{DTW}$), which refers to a weighting combination between the raw series and the first order differences. That is, they find the DTW distance between two series and their corresponding two difference series, and combine them with a weighting parameter α:

$$DD_{DTW} = \alpha \cdot DTW(X, Y) + (1 - \alpha) \cdot DTW(diff(X), diff(Y)) \tag{2}$$

where the first order differences of the time series (*diff*) are defined as:

$$x'_i = x_i - x_{i+1} \quad i = 1, ..., m - 1 \tag{3}$$

The **Move-Split-Merge** (MSM) distance was proposed by Stefan *et al.* [18], where the similarity is calculated using a set of operations that transform a series into a target series. In the move operation, a value is substituted by another; the split operation inserts an identical copy of a value after itself and the merge operation deletes a value if it is followed by an identical value (see figure 5). This algorithm is based on the edit distance, that was originally proposed to calculate the similarity between two sequences of strings based on the minimum edit operations (delete, insert and replace) required to transform one sequence into the other, and adapted to work with sequences of real numbers. Thus, the MSM distance between two time series $X = \{x_0, x_1, ...x_{N-1}\}$ and $Y = \{y_0, y_1, ...y_{M-1}\}$ is the cost of the lowest-cost transformation to convert the series X into the series Y. When working with real numbers, the distance between the points in a time series is either 0 or 1. If two points (x_i, y_j) are closer to each other than a specified threshold value (ϵ), they are considered equal, and their distance is considered to be 0. In the opposite case, where their distance in absolute sense is bigger than ϵ they are considered different and their distance 1.

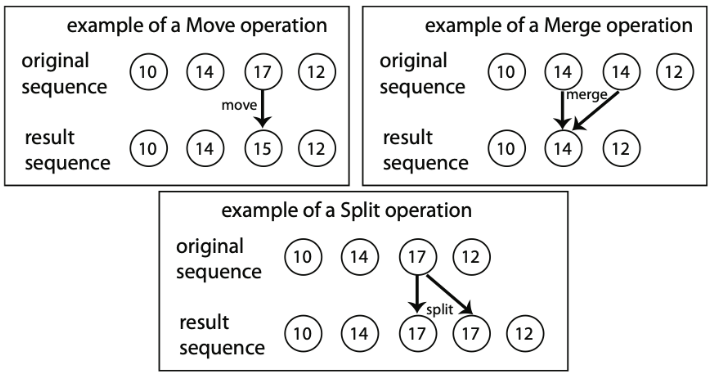


Figure 5: Examples of the move, split and merge operations, image taken from [18].

In the case of this adapted edit distance, when the temporal order of the instances is altered, converting one series into another target series will require from different transformations. Thus the value of the distance between two time series will change. Because of that, time series that were originally similar to each other might not longer be, and vice versa.

Another time domain distance measure is the **Longest Common Subsequence** (LCSS) [19], [20]. There are several approaches to calculate this non-metric based distance measure [19], which is based on the solution of the LCSS problem in pattern matching. This method uses pointwise similarity and it has a great ability to ignore noise and distortion values. Although, it was originally designed to measure the similarity of two sequences of characters (see Figure 6), it has been extended to consider real-valued time series, by using a distance threshold (ϵ). The latter will determine the maximum difference allowed between a pair of values for them to be considered a match.

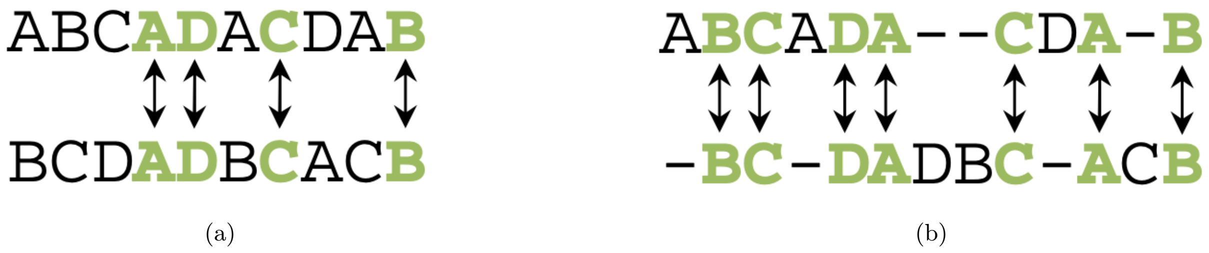


Figure 6: An example of the common subsequences of two sequences of character, (a) without inserting gaps between the characters (non-elastic) and (b) inserting gaps (elastic). When the gaps are inserted allowing elasticity, the number of common subsequences grows, as well as the size of some of such subsequences. Image taken from [21].

The LCSS distance between two series $X = \{x_0, x_1, ...x_{N-1}\}$ and $Y = \{y_0, y_1, ...y_{N-1}\}$ is calculated as follows:

$$d_{LCSS}(X,Y) = 1 - \frac{LCSS(X,Y)}{N} \tag{4}$$

where, $LCSS(X,Y)$ is the length of the LCSS between the two series, and N the length of those series. Moreover, several methods have been proposed to improve the conventional method to compute the LCSS [20] distance. One of those improvements consists on considering an elastic method to compute the LCSS by inserting gaps between the characters (i.e. pointwise values of the time series), as explained in Figure 6(b), to find the greatest number of matching pairs for an optimal alignment. The latter, provides elasticity to the measure and allows the search for the optimal alignment between two series. In this case, the distance between two time series of different length can be computed.

In the case of the LCSS distance, depending on whether elasticity is considered or not, the value of the distance between two time series will be affected differently when the temporal order of the instances is altered. In the case where no gaps are inserted, the pointwise values that in the original dataset were aligned, will continue to be aligned, but in another timestamp value. Whether they were considered similar, will also remain unchanged. Nonetheless, the number of consecutive matching points will vary as the order of the observations is altered. In the case of the elastic approximation, first gaps are inserted to find the optimal alignment, thus the values that were in the same timestamps in the original dataset might not coincide. Then, as in the non-elastic case, the number of consecutive matching points may vary, changing the value of the distance between two time series.

Table 1 sums up the main characteristics of the presented algorithms as well as their advantages and drawbacks.

In the distance-based classifiers where a rigid distance measure, such as ED, is used, the value of the distance between two TS is independent of the temporal order of the instances. Thus its performance will not vary whenever the timestamps are re-ordered randomly. Moreover, in all the distance-based classifiers where the used distance measure is elastic, shuffling the timestamps of the observations does affect the performance of the classifier. The values of the distances between two time series will change, as there will exist a new optimal alignment which will no longer be representative of the original intrinsic temporal information of the dataset. The situation is similar when the edit distance is used in combination with the distance based classifier.

distance measure	type	different length	sens. to temporal order	advantages	drawbacks
Euclidean	rigid	no	no		wrong distance calculation for disaligned TS
DTW	elastic	yes	yes	alignment of time series	only considers the Y-axis value of datapoints, does not account for the relative importance regarding the phase difference between a reference point and a testing point, unintuitive alignments in presence of singularities, large requirements of time and memory
WDTW	elastic	yes	yes	penalizes points with higher phase difference, prevent minimum distance distortion caused by outliers	large requirements of time and memory
DDTW	elastic	yes	yes	more intuitive alignments in presence of singularities, superior alignments (information about the shape given by the derivatives)	large requirements of time and memory
MSM	edit	yes	yes	efficient, invariant to the choice of origin	
LCSS	rigid / elastic (gaps)	no / yes	yes	ignores noise and distortion	

Table 1: Summary of the described distance measures for time series. For each measure it is specified the measure type, whether it accepts datasets composed by instances with varying lengths, whether its performance is changes when the temporal order of the instances is altered and their main advantages and drawbacks.

2.2 Learning Features: Structural (dis)similarities

The algorithms explained in this section, create discriminatory features that are representative of time series and then, they base the classification on them. That is, these algorithms construct abstractions of the time series in the dataset by learning features that best represent the classes and discriminate between them, and mapping the real valued time series to the new feature space. Thus the classification relies on those representative features. There are several strategies to create such representative variables (see figure 7):

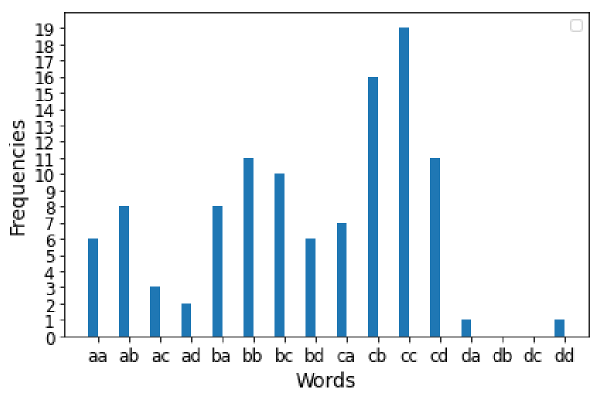
- *Dictionary* based classifiers discretize subsequences of real-valued TS by a symbolic representation to create representative words, which will become the new representative variables. Then, histograms are constructed from those word counts. In some approaches, every TS has an associated histogram, which represents its corresponding word count; in some others, the histograms are associated to each of the classes. In conclusion, pattern frequency features are created, as the words counts are the values for the variables. Some dictionary-based approaches use feature selection methods to select the most discriminative words.

- In *shapelet* based classifiers, a set of subsequences of TS that are highly representative of a class is defined. Those subsequences are called shapelets, and they form the new set of attributes. The values of such attributes for each TS are the distances between the shapelet and the subsequence that represents its best position on the TS. One of the main differences among the shapelet-based algorithms is the procedure they use to find such set of discriminatory shapelets.

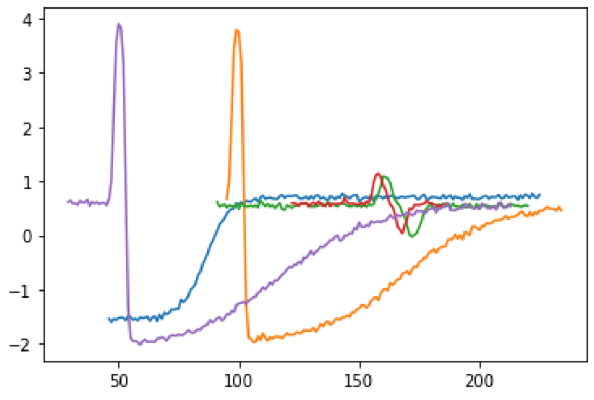
- With an idea similar to shapelet based classifiers, some algorithms use *convolutional kernels* as variables. Those kernels will be representative of different substructural shapes. The real-valued time series will be represented by attributes whose valued will be based on the results of the convolution between the kernel and the TS.

- In *interval* based classifiers, the instances are divided in intervals and summary statistic features are extracted from those intervals (e.g. the mean and the standard deviation of the interval).

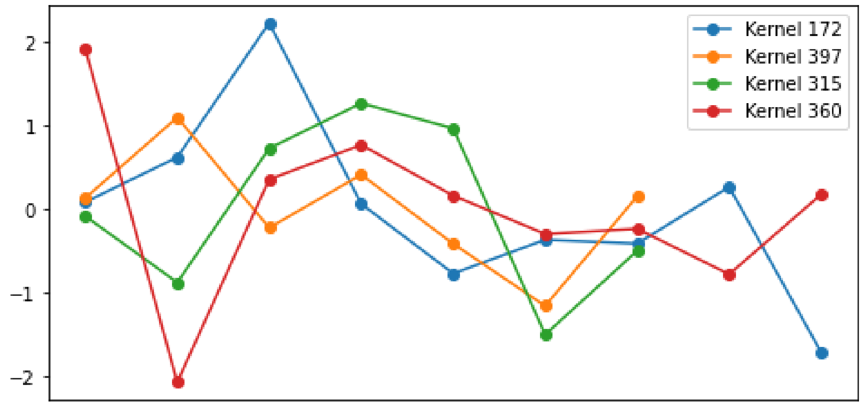
Once the real-values time series are mapped to the new feature space, any distance-based or feature-based classifier can be used for classification. Moreover, those features can be used as a splitting criteria for classifiers based on decision tree structures. The algorithms presented in this section use very distinct classifiers, thus the latter will be specified in the case of each of the algorithms.



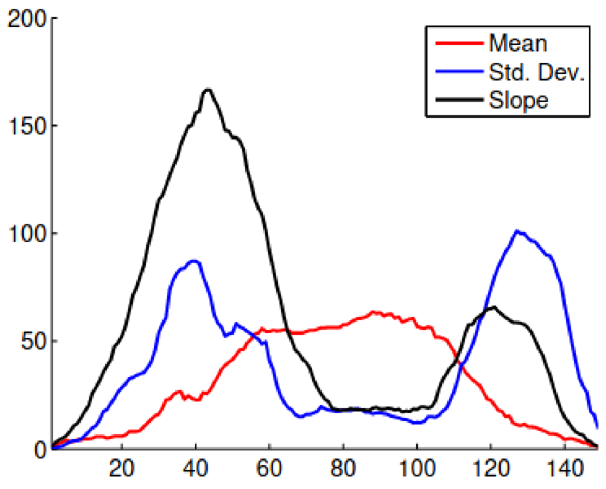
(a) Dictionary



(b) Shapelet



(c) Convolutional kernel



(d) Interval

Figure 7: Graphical summary of the different feature extraction methods.

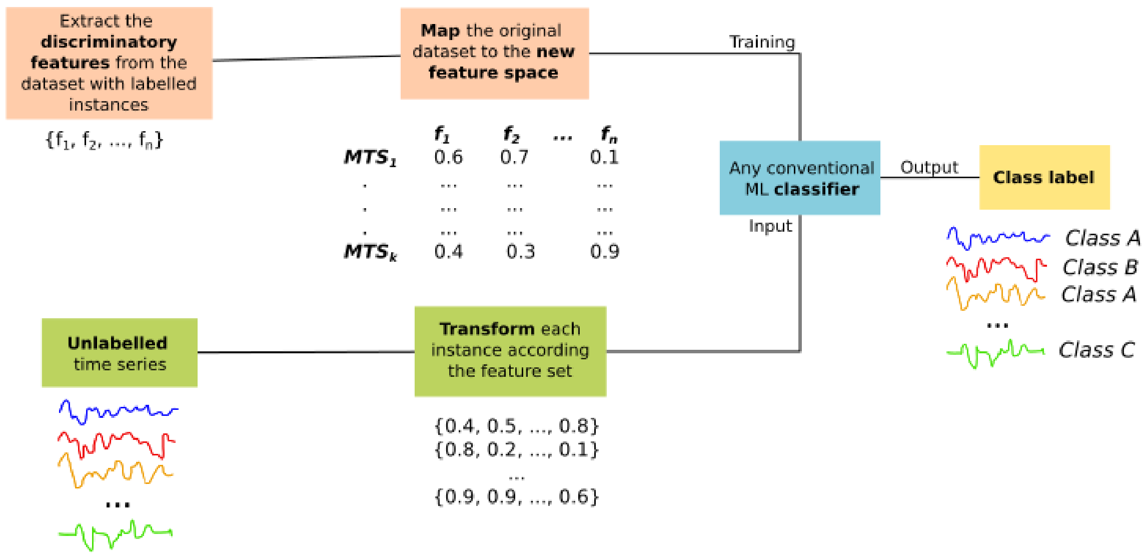

Extract the **discriminatory features** from the dataset with labelled instances
{f1, f2, ..., fn}
Map the original dataset to the **new feature space**
Training
Any conventional ML **classifier**
Output
Class label
Input
f1 f2 ... fn
MTS1 0.6 0.7 0.1
MTSk 0.4 0.3 0.9
Class A
Class B
Class A
Class C
Unlabelled time series
Transform each instance according the feature set
{0.4, 0.5, ..., 0.8}
{0.8, 0.2, ..., 0.1}
{0.9, 0.9, ..., 0.6}

Figure 8: The general workflow of the specific TS classifiers based on feature extraction.

Compared to the algorithms of the previous section, the extracted features provide to base the classification in higher-level structures, rather than point-to-point local comparisons.

It must be outlined, that in the algorithms where the classification is based on extracted features, the classifiers themselves are not sensitive to the temporal order of the instances. The temporal information is recorded on the attributes selected to represent the instances. Therefore, when the temporal order of the instances is altered, the performance will not vary because the classifier is reading the wrong temporal correlation; but because the attributes in which the classification is based do not represents the real temporal correlation of the observations, as they will not longer be representative of the original shapes of the instances.

2.2.1 Dictionary Based Classifiers: Pattern Frequency Features

Dictionary based classifiers use frequency of words as the basis for finding discriminatory features. They transform the series into representative words, reducing its dimensionality. To transform the real valued series, a sliding window of size w goes through the series, producing l values for each window, and discretising those values by assigning a symbol from an alphabet α. The classification is performed calculating the similarity based on the distribution of words. Some approaches, base the classification on the histograms resulting from the word counts, which are representative of particular instances or a given class, depending on the approach. In those cases, a distance-based classifier is used for classification (see Figure 9). Other algorithms, execute a feature selection process and classify the unlabelled instances by feature-based classifiers (see Figure 9).

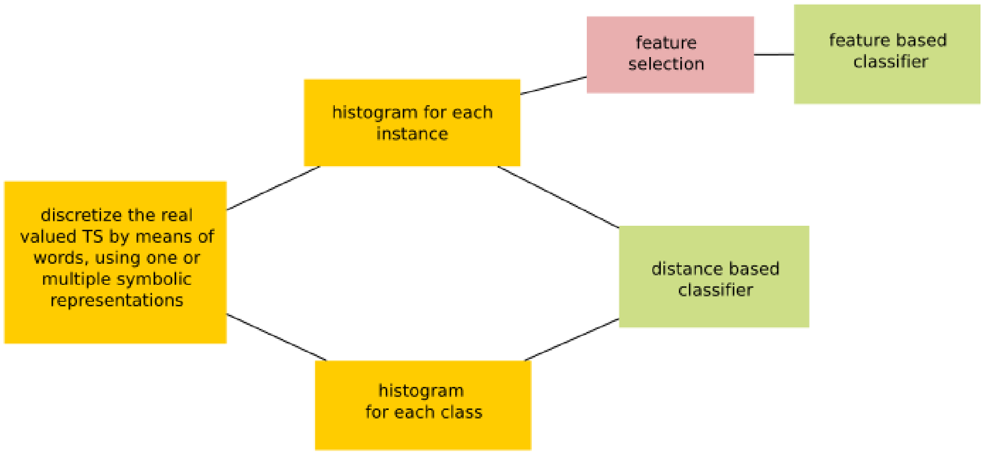

feature selection
feature based classifier
histogram for each instance
discretize the real valued TS by means of words, using one or multiple symbolic representations
distance based classifier
histogram for each class

Figure 9: A general overview of the workflow of different dictionary based classifiers.

The **Bag Of Patterns** (BOP) [22], is a dictionary based classifier that bases the conversion of the series to strings on the Symbolic Aggregate Approximation (SAX) [23] method. The SAX method produces a lower dimensional representation of a time series by transforming the original series into symbolic words (see Figure 10). Once the time series is discretized, the algorithms extracts subsequences of fixed length (specified by the user) by the sliding window technique [24], obtaining a set of strings, each of which corresponds to a subsequence in the time series. In some cases multiple consecutive subsequences are mapped to the same string. To avoid over-counting those trivial matches as true patterns, the *numerosity reduction* [23] technique is applied: only the first occurrence is recorded, and the rest is ignored until a different string is encountered. Once the set of strings regarding each of the time series is defined, the word-sequence matrix M is constructed, which is the "bag of patterns" matrix. The M_{ij} elements of such matrix denote the frequency of the word i in the time series j (see Figure 11). The distribution of words of a series obtained in the M matrix forms a count histogram. To perform the classification, the 1-NN algorithm is used, based on a histogram-based similarity measure. In most cases, the Euclidean distance between the histograms is calculated and then classified by the 1-NN algorithm.

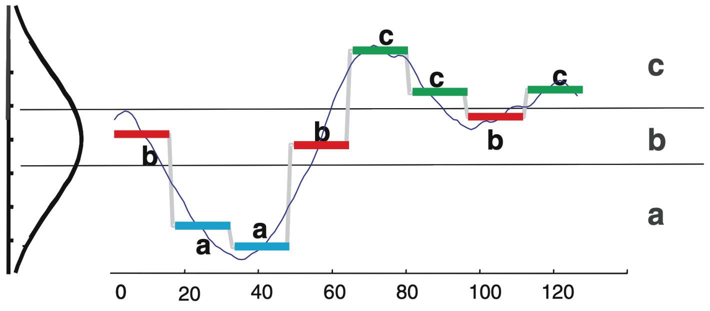
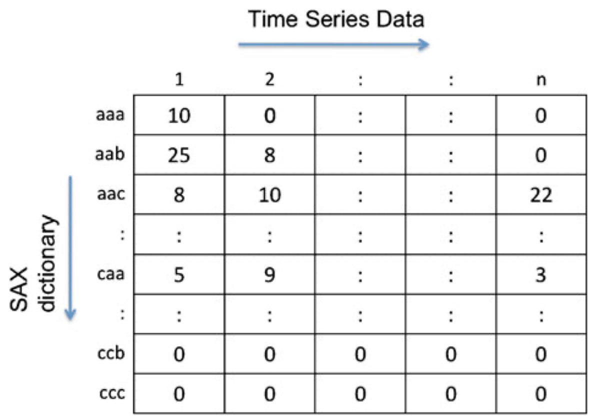


Figure 10: A time series discretized into SAX symbols, image taken from [23].


Time Series Data
SAX dictionary

	1	2	:	:	n
aaa	10	0	:	:	0
aab	25	8	:	:	0
aac	8	10	:	:	22
:	:	:	:	:	:
caa	5	9	:	:	3
:	:	:	:	:	:
ccb	0	0	0	0	0
ccc	0	0	0	0	0

Figure 11: A visual example of the BOP representation for a time series, image taken from [22]. Each row refers to a SAX word and each column denotes one time series from the database.

There is another classifier that combines the SAX representation with the Vector Space Model, called **Symbolic Aggregate Approximation - Vector Space Model** (SAXVSM) [25]. As in the case of the BOP algorithm, using a sliding window subsequence extraction and the SAX method, the labeled time series are transformed into collections of SAX words. This algorithm, builds bags of SAX-generated words representing each of the training classes assembles them into a corpus. The latter, is a sparse *term frequency matrix*. The rows of the matrix represent the set of all the SAX words, and the columns each of the classes of the training set (and not each of the time series of the dataset as in the case of the BOP algorithm). The elements of the matrix represent the observed frequency of a word in a class; thus in this case, the distribution of words is formed over the classes rather than series. In the next step, each of the elements of the matrix undergoes a weighting by the term frequency and inverse document frequency ($tf \cdot idf$), transforming the frequency values into weight coefficients:

$$tf \cdot idf(tf, df) = \begin{cases} log(1+tf) \cdot log\left(\frac{c}{df}\right), & \text{if } df > 0, \\ 0, & \text{otherwise} \end{cases}$$

where c is the number of classes, the term frequency (tf) is the number of appearance of a word in a class, and the inverse document frequency (idf) refers to the number of classes a word appears in. The resulting weight vectors are retained for classification. To perform the classification, the unlabeled time series is transformed into a terms frequency vector by the same sliding window technique and SAX parameters as in the training phase. Then, the cosine similarity between the terms frequency vector and the weight vectors of the training classes are performed. The unlabeled time series is assigned to the class whose vector has obtained the highest similarity value, using the 1-NN classifier. The procedure is summarized in Figure 12.

Another dictionary based algorithm is the **Bag of SFA Symbols** (BOSS) [26], a combination of the noise tolerance of the Symbolic Fourier Approximation (SFA) [27] with the structure based representation of the bag-of-words model (see Figure 13). In this algorithm, firstly, sliding windows of fixed length are extracted for each time series. Then, the SFA transformation is applied to each of the real valued sliding window. The SFA is a symbolic representation of a time series by means of a sequence of symbols, known as SFA *words*. The SFA is composed of two operations. First, the Discrete Fourier Transform (DFT) decomposes a signal into a sum of sinusoidal waves. Each wave is represented by a complex number called a Fourier coefficient:

$$DFT(X) = \{x_0, ..., x_{n-1}\} = (real_0, imag_0, ..., real_{n-1}, imag_{n-1}) \tag{5}$$

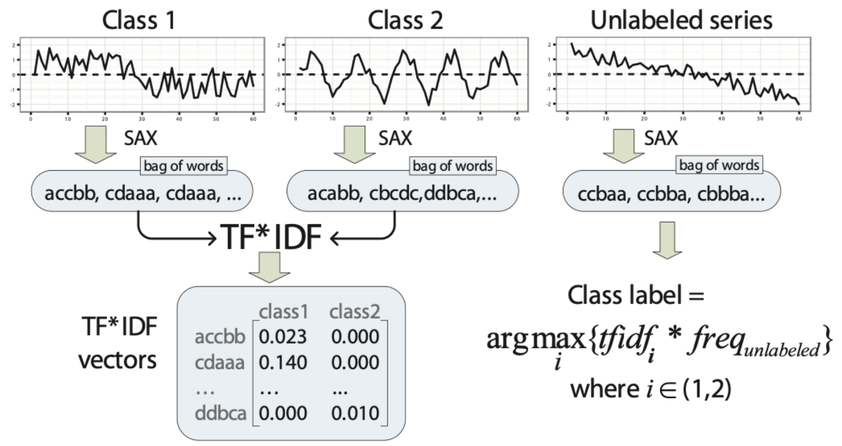


Figure 12: A graphical summary of the SAX-VSM algorithm, image taken from [25].

The first Fourier coefficients correlate to lower frequency ranges, and are commonly used to describe the signal, thereby low pass filtering the signal and smoothing it. Once the series is truncated, it is discretized by Multiple Coefficient Binning (MCB) instead of using fixed intervals as in the previous cases. The latter, chooses the discretizing points estimating the distribution of the Fourier coefficients. For estimating those coefficients, the series is divided in several segments, a DFT is performed, and breakpoints for coefficients are chosen so that there is the same number of elements on each bin. Thus from the train data the labelled MCB intervals are obtained, and based on it, the SFA words are created. As in the previous algorithm, *numerosity reduction* [23] is applied. From the resulting SFA words a histogram is constructed. To perform the classification, they propose to use the 1-NN algorithm calculating the distances between the transformed time series by the BOSS distance. The latter is a non-isometric function, and only includes the distances between frequencies of words that occur on the first histogram passed as an argument.

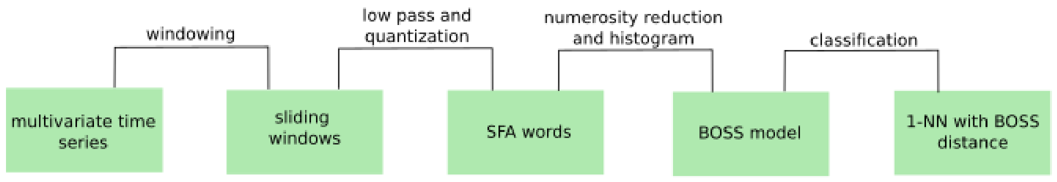


Figure 13: A summary of the procedure of the BOSS algorithm.

The BOSS algorithm uses a sliding window of fixed length. Nevertheless, the same authors proposed the **BOSS Ensemble classifier** [26], which uses multiple window lengths to allow for different structural sizes. The different length sliding windows are applied to the train data and are given a score. Then the classifier, classifies a query time series using the best window sizes, and those that present an accuracy above a given threshold are used for prediction. The latter are used to perform 1-NN classifications with the query, assigning for each of the window lengths a class label to it. Finally, the most frequent class label is chosen to be the output of the classification algorithm.

Later, the **contractable Bag of SFA Symbols** (cBOSS) [28] was proposed to overcome the considerable amount of build time and space the BOSS Ensemble algorithm needed on larger datasets. It fixes a number of different BOSS classifiers to construct (k), and selects the parameters of each of them by a filtered random selection. Moreover, they assess an alternative voting scheme based on weighted probabilities instead of using the majority vote method. Leave-one-out cross-validation is used to generate probabilities for the weightings and to estimate accuracy. Furthermore, for the classification of the base classifiers, it uses decision tree classifiers instead of using the 1-NN classifier, which requires higher amounts of memory. In some applications, the maximum ensemble size is replaced by an amount of time, continually building new classifiers until it runs out of time.

Middlehust *et al.* [29] proposed the **Temporal Dictionary Ensemble** (TDE), which is also an enseble of BOSS classifiers. It follows the general structure and weighting scheme of the cBOSS algorithm. Nevertheless, TDE uses a guided parameter selection for ensemble members based on a Gaussian process. Moreover, instead of using the BOSS distance, it uses the histogram intersection distance measure [30]. The BOSS distance ignores the location of words in a series, and the locations of certain discriminatory subsequences may happen to be important. The histogram intersection distance measure based on spatial pyramids [31] overcomes such issue. In this case, the height defines the importance of localisation for the transform. Histograms are weighted to give more importance to similarities in the same locations than global similarity, and are concatenated to form an elongated feature vector per instance (figure 14).

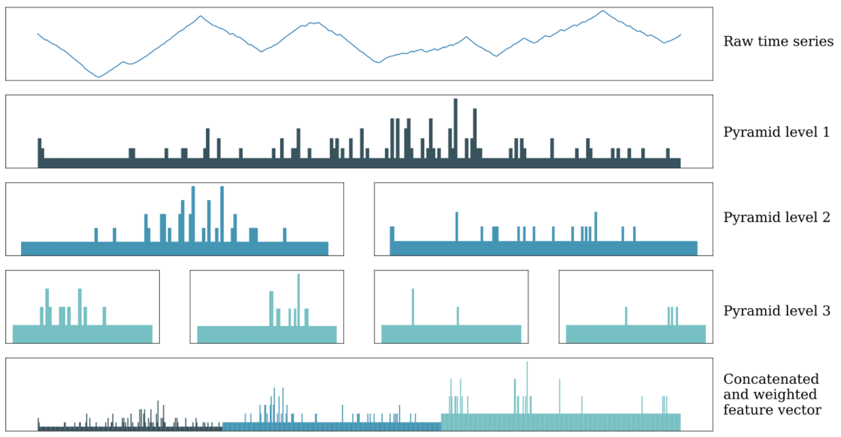


Figure 14: An example of the transformation of a time series using spatial pyramids. Image taken from [29].

Le Nguyen *et al.* [32] designed the **Multiple Representation Sequence Learner** (MrSEQL). This algorithm uses multiple domain representations, as it combines symbolic representations in time domain (SAX [23]) with frequency domain representations (SFA [27]). Moreover, it performs multiple resolutions of those symbolic representations by using different parameter settings. Each of the symbolic representations is trained by the SEQL classifier [33], which uses a greedy feature selection. The latter was originally designed as a binary classifier for sequence data such as biological sequences (e.g. DNA) or text. It explores the space of all subsequences employing a tree based approach with a branch-and-bound strategy, selecting a set of discriminative subsequences. As in the previous algorithms, the dataset is transformed according the new feature space. The instances represented in the new symbolic space are used to train a linear model composed by the set of the discriminative subsequences and their corresponding coefficients. The latter can be interpreted as the discriminative power of the subsequence. The general workflow of the MrSEQL algorithm can be seen in Figure 15.

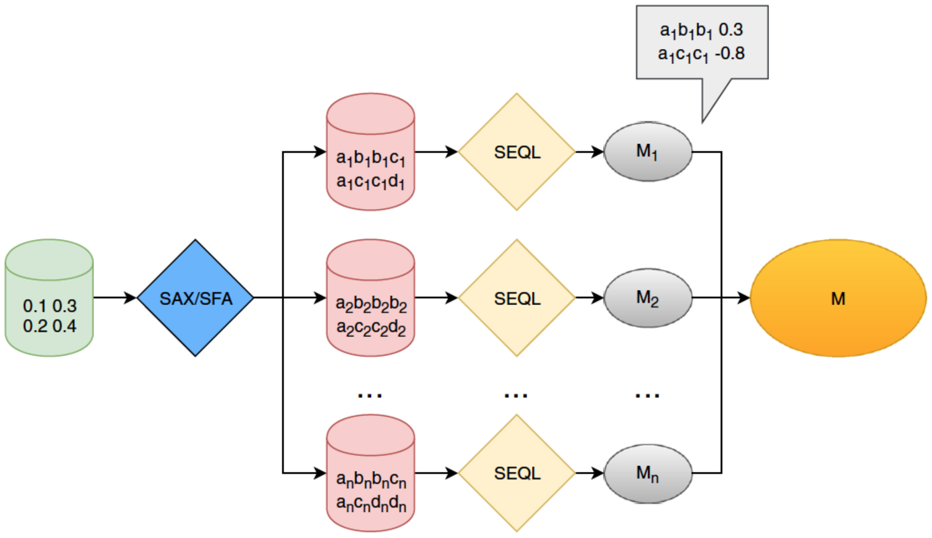
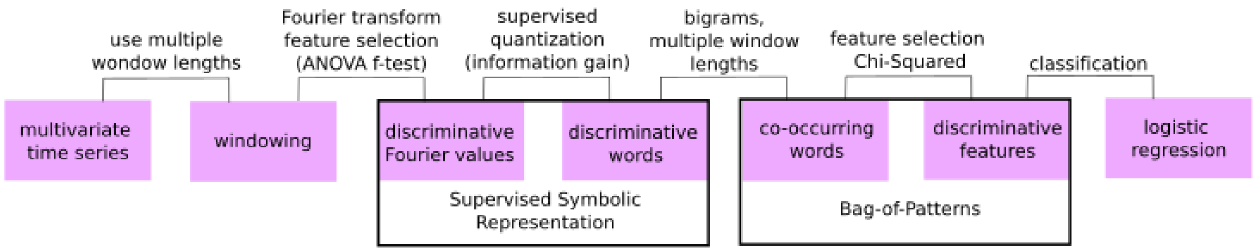


Figure 15: The workflow of the MrSEQL classifier. Image from [32].

Shcäfer *et al.* [34] proposed an algorithm known as **Word Extraction for Time Series Classification** (WEASEL). First, the SFA [27] symbolic representation is used to transform the real valued series into a set of words by using the sliding window technique. Nevertheless, instead of using the classical procedure, the choice of the most discriminative Fourier coefficients is performed by using ANOVA f-test [35] and apliying an information gain binning to find the most appropriate discretization boundaries (*discriminative quantization*). Moreover, it extracts multiple-length windows, and builds a single model from the concatenation of feature vectors instead of considering each fixed-length window as independent feature. With the counts of words derived from the discretization of TS, histograms are built. Then, WEASEL applies the Chi-squared (χ^2) test [36] feature selection aiming to remove irrelevant features from the classes, obtaining a highly discriminative feature set. Then, the time series are mapped to the new feature space using sparse vectors, and with that, a logistic regression method is trained. To classify unlabeled query time series, they are transformed according the model's feature space, then the trained logistic regression model predicts a class label for the query. Figure 16 is a graphical representation of the procedure of this algorithm.



use multiple wondow lengths
Fourier transform feature selection (ANOVA f-test)
supervised quantization (information gain)
bigrams, multiple window lengths
feature selection Chi-Squared
classification
multivariate time series
windowing
discriminative Fourier values
discriminative words
Supervised Symbolic Representation
co-occurring words
discriminative features
Bag-of-Patterns
logistic regression

Figure 16: The workflow of the WEASEL algorithm.

Later, they extended the version developing the **Word Extraction for Time Series Classification plus Multivariate Unsupervised Symbols and Derivatives** (WEASEL+MUSE) [37], for multivariate time series classification (MTSC), charachterized by the interplay of features in different dimensions. First, each of the dimensions is considered a univaried TSC problem and undergoes the transformation proposed in the WEASEL [34] algorithm. The same procedure is applied to the derivatives of those univariate time series. The resulting words are concatenated with an identifier (name of the dimension and window size) to form multivariate words. To avoid irrelevant features and dimensions a Chi-squared (χ^2) test [36] is applied to the histogram of discrete features extracted from all dimensions, to obtain a highly discriminative feature vector. The instances are mapped into the new feature space, then a logistic regression classifier is trained based on the transformed dataset. As the weight vector is trained over all the dimensions, this algorithm takes into account the interplay of the multiple dimensions. The procedure of the WEASEL+MUSE algorithm is summarized in Figure 17.

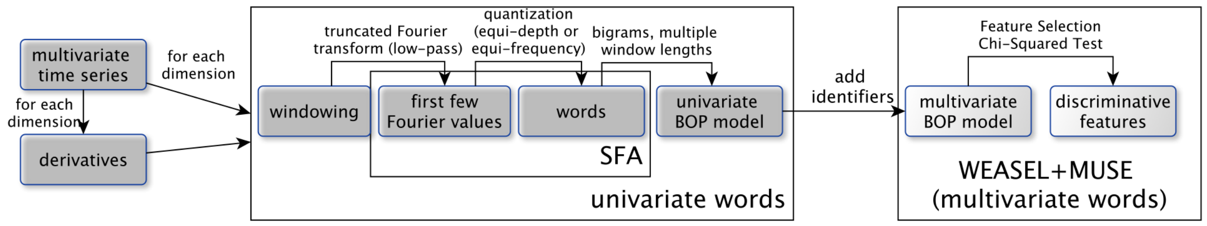


Figure 17: The workflow of the WEASEL+MUSE algorithm. Image from [37].

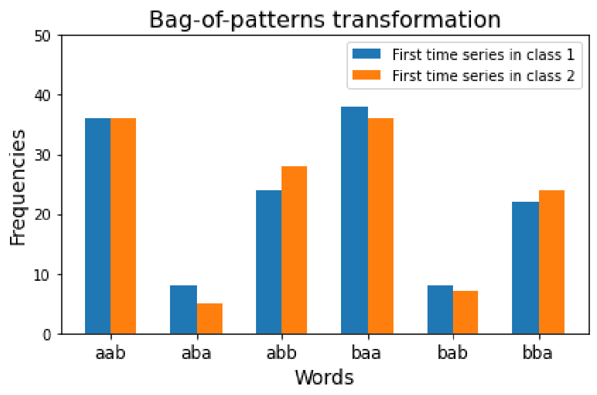
Table 2 sums up the main characteristics of the presented dictionary-based classifiers, such as, which symbolic representation(s) they use, whether they use fixed window length or multiple sliding windows are considered, whether they use a feature selection method and which is the classifier they use.

algorithm	symbolic representation	window length	phase independent	feature selection	classifier
BOP	SAX	fixed	yes	no	1-NN (euclidean distance between histograms)
SAXVSM	SAX	fixed	yes	no	1-NN with cosine similarity
BOSS	SFA	fixed	yes	no	1-NN with BOSS distance between histograms
BOSS Ensemble	SFA	multiple	yes	no	multiple 1-NN with BOSS distance between histograms and majority vote
Contractable BOSS	SFA	multiple	yes	no	multiple decision trees and weighted vote
TDE	SFA	multiple	no	no	multiple 1-NN with histogram intersection distance and weighted vote
MrSEQL	SAX and SFA	multiple	yes	yes, branch-and-bound	linear model
WEASEL	SFA	multiple	no	yes, Chi-squared test	logistic regression
WEASEL + MUSE	SFA	multiple	yes	yes, Chi-squared test	logistic regression

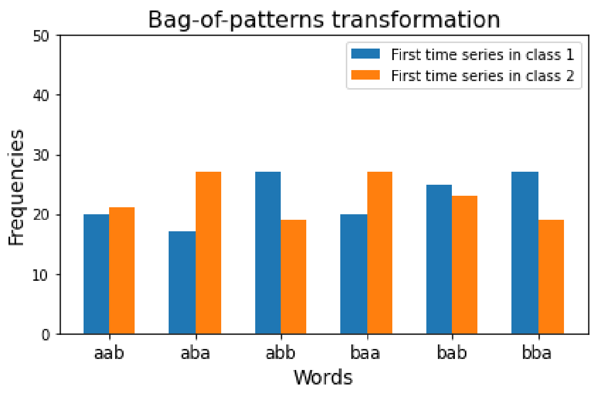
Table 2: Summary of the main characteristics of the presented dictionary-based algorithms.

In dictionary based classifiers, when the natural order of the instances is changed, the subsequences defined by the sliding window procedure will be different. Therefore, when they are discretized by a symbolic representation, the words that are formed are different. Consequently, the word count histograms related to the instances will be different, and the words and the distributions of the histograms will not longer be representative of the original temporal information of the dataset. Henceforth, in those cases where the classification is based on the distance between histograms (e.g. BOP, BOSS), that distance will not longer define in a correct way the real closeness between instances, thus the prediction may come out wrong. The classifier itself, is not sensitive to the alteration of the order; but as the given histograms represent incorrect temporal information, their predictions will lose accuracy.

For example, in the particular case of a BOP algorithm that uses an alphabet of size 2, a word length of 3 and a window size of 9, Figure 18(a) shows the resulting histograms of the first instance of class 1 and the first instance of class 2. If the temporal order of the observations is altered, the histograms related to those instances are the ones shown in Figure 18(b). Henceforth, Figure 18 shows that altering the temporal order of the instances changes the representation of the latter, and thus the histogram related to it. Therefore, when the 1-NN classifier is used to calculate the distance of the histograms, the results will vary, as the distance will no longer represent a true closeness between two instances. The effect of the variation on the temporal order is similar in the case of the BOSS algorithm (see Figure 19).

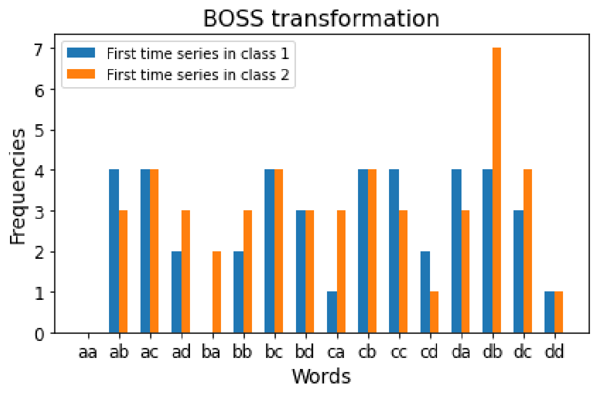


(a) Original

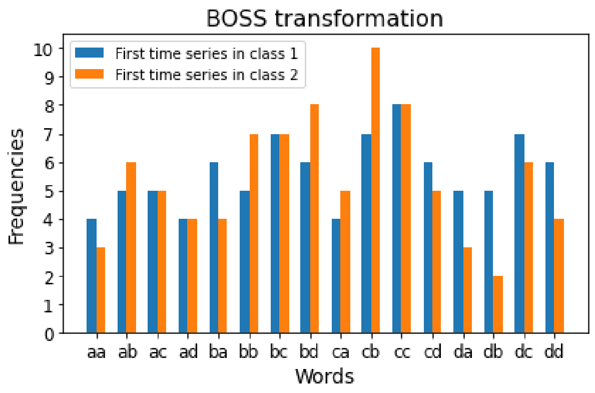


(b) Altered

Figure 18: The histograms of the first instance of class 1 and the first instance of class 2 in for the (a) original dataset and (b) the datasets with the temporal order of the instances randomly altered, using the BOP algorithm with an alphabet of size 2, a word length of 3 and a window length of 9 to classify the instances of the `Plane` dataset [38].



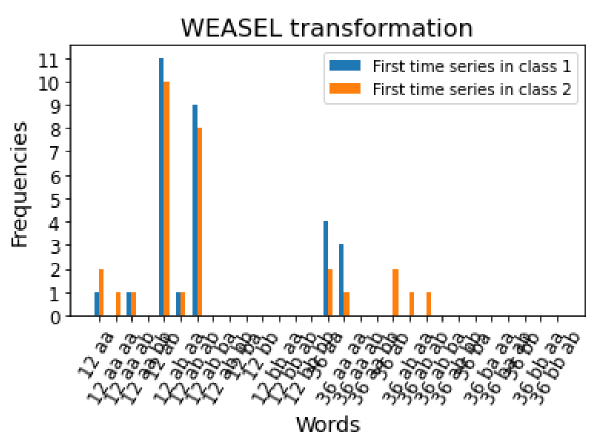
(a) Original



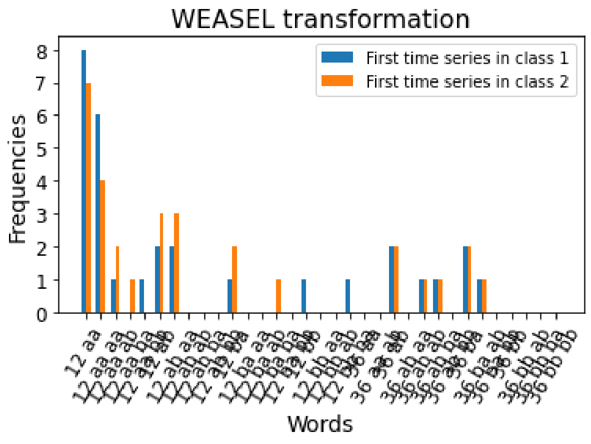
(b) Altered

Figure 19: The histograms of the first instance of class 1 and the first instance of class 2 in for the (a) original dataset and (b) the datasets with the temporal order of the instances altered, after the BOSS algorithm with an alphabet of size 4, a word length of 2 and a window size of 12 is applied to the `Plane` dataset [38].

Moreover, in the cases where the most discriminative words are selected (e.g. WEASEL), as the instances are represented by other representative words that do not belong to the original discriminative shapes (see Figure 20), the words of the altered dataset may not be as discriminant as the words derived from the original instances, as they have not been derived from original patterns. Therefore, the classification based on those words might not be as accurate as that based on the discriminative words of the original dataset.



(a) Original



(b) Altered

Figure 20: The histograms of the first instance of class 1 and the first instance of class 2 in for the (a) original dataset and (b) the datasets with the temporal order of the instances altered, for the `Plane` dataset [38], when applying the transformation corresponding to the WEASEL algorithm. In this example, the words are of size 2, the alphabet has a size of 2 and two different window lengths (12 and 36) are considered.

Besides, in the algorithms where the sliding window technique [24] is used, when a single fixed-size sliding window is used, only those temporal substructures that match with the window size can be extracted. Instead, when multiple window lengths are used, a wide variety of temporal substructures are considered for classification. Henceforth, the latter are expected to better catch the intrinsic temporal information of the observations.

To sum up the information of this section, Table 3 collects the main advantages and drawbacks of the algorithms considered in this section.

algorithm	advantages	drawbacks
BOP	good performance in long instances, anomaly detection, interpretable pattern distribution	single window size, minimum length required to capture the high-level structures
SAXVSM	interpretable class generalization, fast classification, robust to noise	computationally expensive learning phase
BOSS	noise reduction	cross validation to find the optimal set of parameters, single window size, memory and time inefficient
BOSS Ensemble	multiple resolutions of symbolic representations, noise reduction	large memory requirements
Contractable BOSS	efficiency, multiple resolutions of symbolic representations, noise reduction	large memory requirements
TDE	multiple resolutions of symbolic representations, noise reduction, temporal location of patterns	memory intensive, slow for classifying
MrSEQL	time and frequency domain representations, multiple resolutions of symbolic representations, interpretable patterns, feature selection, efficient time and space complexity	
WEASEL	fast classification, multiple resolutions of symbolic representations, feature selection, temporal order of windows	high build time, large memory requirements
WEASEL + MUSE	multiple resolutions of symbolic representations, robust against noise, capture general shapes, considers the interplay of features in multivariate time series	high build time, large memory requirements

Table 3: The main advantages and drawbacks of the dictionary-based algorithms.

2.2.2 Shapelets

Shapelets are time series subsequences which are in some sense highly representative of a class [39]. They provide the detection of phase-independent localized similarity between series that belong to the same class. Shapelet based algorithms perform the classification based on the distance to the shapelet, rather than the distance to the nearest neighbour. That is, they use the similarity between a shapelet and a series as a discriminatory feature. For that purpose, the principal aim of such algorithms is to find the set of best subseries that can help discriminate between classes. Each of the learned shapelets is placed at the best position in the time series (see Figure 21), under some distance measure (usually Euclidean), and the 'matching' of the shapelet to the time series refer to its distance at that position. Once the instances are represented by the new features, any feature-based classifier could be used for classification, although the original shapelet-based algorithms used them as a splitting criterion for decision trees. The main difference between the algorithms explained in this section is the method used in the shapelet search process, which will determine the computational cost of the algorithms.

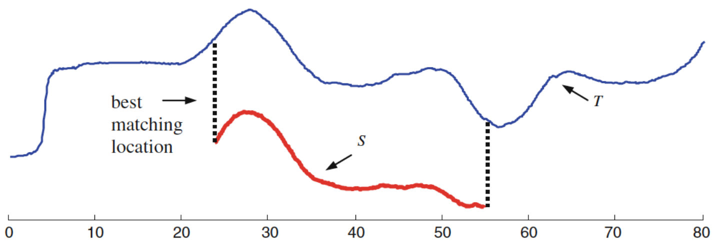


Figure 21: A graphical example of the best matching location. Image taken from [39].

In the original algorithm that made use of shapelets for classification [39], they are used as the splitting criterion for a decision tree. At each step of the decision tree induction, a shapelet and its corresponding split point is determined (see Figure 22). Therefore, each internal node of the decision tree contains information about a single shapelet classifier, and the left and right subtrees. To determine de quality of candidates, information gain is used. Moreover, the leaf nodes embody the information about the class labels. To predict the class labels of new instances, starting from the root of a shapelet classifier, the distance between the instance to classify and the shapelet in that node is calculated. If the distance is smaller than the splitting point, the left subtree is used; and the right subtree otherwise. The procedure is repeated until the leaf node is reached, and the corresponding class label returned. This algorithm has a high training complexity due to the large number of candidate shapelets and the repeated scanning of the data, as brute force algorithm is used for finding the candidate shapelets. Nevertheless, they implemented several techniques to speed up the algorithm, such as, early abandonding of distance computations or admissible entropy pruning of the information gain metric. In the last years, more algorithms based on shapelets have been developed, focused on optimising the original algorithm.

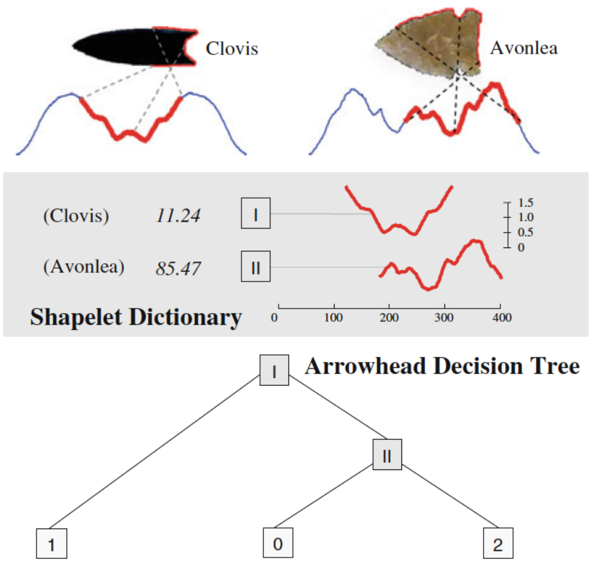


Figure 22: An example of the shapelet-based decision tree classifier for the ArrowHead TSC problem. The image shows the resulting shapelet dictionary with the corresponding threshold values. Image taken from [39].

Rakthanmanon and Keogh [40] described an extension of the original shapelet based approach [39], that provided a faster shapelet discovery, called **Fast Shapelets** (FS). Instead of performing a full enumerative search at each node, it solves the shapelet discovery problem with a change of representation, which is discrete and low-dimensional. For each object in the dataset, a transformation is performed in the time series using SAX. Multiple SAX words will be created for each time series provided the sliding window technique [24]. The dimensionality of the SAX dictionary is then reduced by masking randomly selected letters (see Figure 23); that is, performing a *random projection*. After multiple random projections are performed, a frequency count histogram is built for each class. Each SAX word will have a score that expresses how well the frequency table discriminates between classes. Then, the k-best words are selected and mapped back to the original shapelets. These shapelets are used as splitting criterion of a decision tree as in the original shapelet based algorithm.

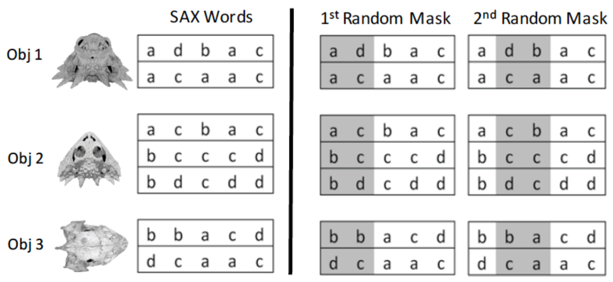


Figure 23: An illustration of the masking technique. Image taken from [40].

Shapelet Transform (ST) is a shapelet based algorithm proposed by Hills *et al.* [41], [42], that identifies the best k shapelets in a single scan of the data. Shapelet quality stands on how well they can discriminate between class labels. As quality measures they propose three alternatives: Kruskall Wallis, analysis of variance F-statistic and Mood's median. Moreover, the number of shapelets can be reduced either by ignoring the shapelets below a cut-off point, or by means of a clustering process. Once the top k shapelets are assessed, the transformed dataset is computed. The latter is composed of k attributes, and their value refers to the distance between each of the k shapelets and the best position in the time series. It is possible to reduce the number of shapelets afterwards in order to remove self-similar shapelets by a clustering process. Once the dataset is transformed according to the new set of attributes, any feature-based classifier can be used. Thus, in this algorithm, the shapelet discovery is separated from the classification. In the latest version of ST [42], the number of shapelets per class are balanced.

Grabocka *et al.* [43] proposed the so called **Learning Shapelets** (LS), where the search of shapelets is performed by a heuristic gradient descent procedure. The method learns the optimal shapelets instead of performing a search among all possible candidates. The algorithm starts with a rough initial guess for shapelets. Then the shapelets are optimized iteratively by minimizing a classification loss function, by means of a stochastic gradient descent optimization process. With that, the k best shapelets are found not restricted to being subseries of the original instances. (Those shapelets are initialised through a k-means clustering of candidates from the training data. A logistic loss function is used for the optimisation process, based on a logistic regression model for each class.) The LS algorithm jointly learns the weights for regression and the shapelets.

Table 4 summarizes the most general characteristics of the presented shapelet based algorithms, such as their shapelet search method, the quality measure of the shapelets and the classifier they use once the features are extracted.

algorithm	shapelet search	quality measure	classifier
original algorithm	brute force	information gain	originally decision tree, but any feature-based classifier could be used
Fast Shapelets	SAX + random projection	discriminative capacity of the frequency table	originally decision tree, but any feature-based classifier could be used
Shapelet Transform	exhaustive search	Kruskal Wallis, variance F-statistic and Mood's median + clustering to remove self-similar shapelets	any feature-based classifier
Learning Shapelets	heuristic gradient descent	loss function	feature-based regression algorithm

Table 4: Summary of the main characteristics of the presented shapelet-based algorithms: the shapelet search technique, the quality measure for shapelets and the final classifier.

When the order of the observations is altered, the shape of the time series changes, and so does the set of shapelets that will discriminate well between classes (see Figure 24). As they represent non original pattern, their ability to discriminate between classes should be worse than that of the discriminative shapelets of the original datasets. Consequently, the performance of the classifier is expected to be worse, as they are based on non real patterns. Moreover, the original instances are smoother that the altered, which present more abrupt changes and seem more noisy. That issue is reflected on the set of resulting discriminant shapelets: the shapelets of the original dataset are smoother and present clear patterns, while those of the altered problem reflect more noisy and chaotic patterns.

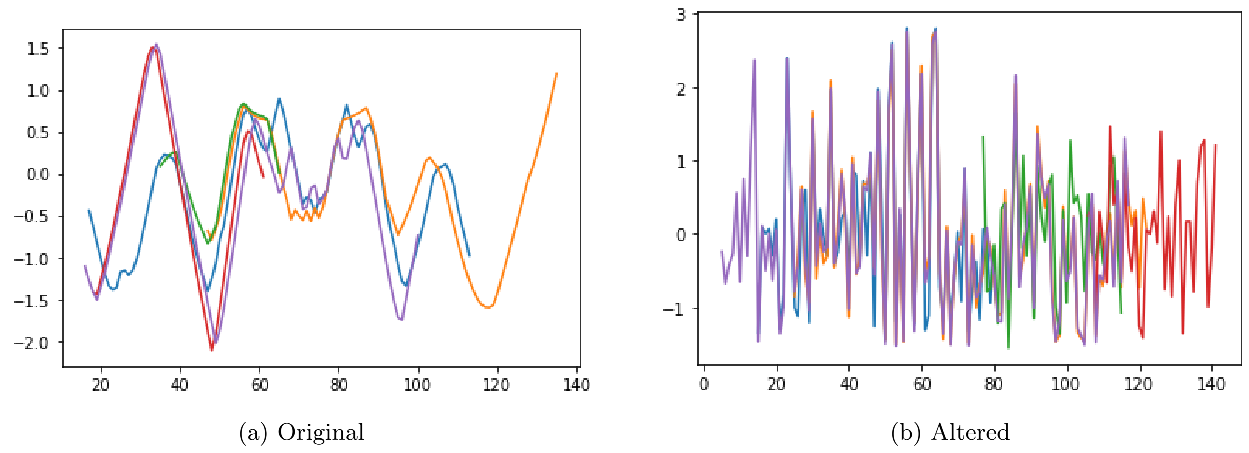

(a) Original (b) Altered

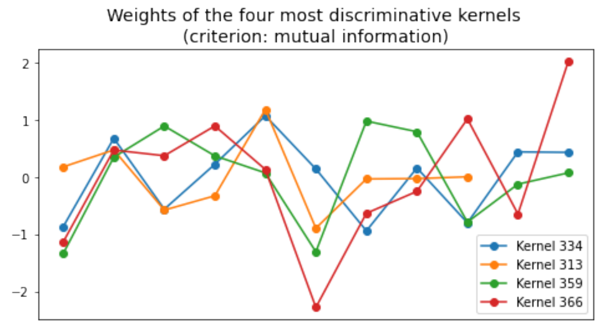
Figure 24: The resultant set of shapelet in which the classification will be based for the (a) original dataset and (b) the datasets with the temporal order of the instances altered, for the `Plane` dataset [38].

2.2.3 Convolutional Kernels

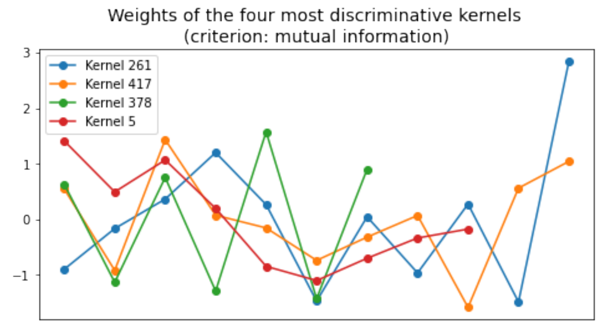
Closely related to shapelet based classifiers, several time series classifiers use convolutional kernels to detect patterns on the input. Those approaches perform the convolution between a kernel and a query time series producing a feature map in which the classification is based. The resulting feature maps represent the degree of presence of the patterns represented by the kernels on the time series.

Dempster *et al.* [44] developed the **Random Convolutional Kernel Transform** (ROCKET), which combines a large number of random convolutional kernels with a linear classifier (ridge regression or logistic regression). Each one of the 10.000 kernels is applied to every instance. The convolution of an instance and a kernel is performed through a sliding dot product, producing a feature map. For each convolution two features are created: the maximum value and the proportion of positive values (*ppv*). The *ppv* feature expresses the proportion of the series correlated to the kernel. Thus, each of the time series is represented as an instance of 20.000 attributes, and the transformed dataset is used to train a linear classifier. In principle, the classification can be performed by any classifier, but they found it performs best when a linear classifier is used. They propose the logistic regression for large datasets and the ridge regression for the rest.

As in the case of the previous algorithms, the alteration of the temporal order will have an effect on the workflow of the algorithms based on convolutional kernels. Altering the temporal order changes the shape of the instances, thus although the considered kernels remain the same, the result of the convolution of such instance with a given kernel will change. Therefore, for each of the kernels, the ability to discriminate between classes will change (see Figure 25). Consequently, when performing the classification based on the results of such convolutions, as the original shapes are lost, they will no longer be highly representative and may lead to wrong predictions.



(a) Original



(b) Altered

Figure 25: The four most discriminative kernels for (a) original dataset and (b) the datasets with the temporal order of the instances altered, in the case of the `Plane` dataset [38].

2.2.4 Interval Based Classifiers

Interval based classifiers create features from intervals of the original instances. There exists a wide possibility for interval features, but in most cases the most simple and interpretable features are used, such as the mean and the standard deviation of the interval. There are loads of possibilities in which a time series can be split, thus there exist a huge amount of possible intervals. Therefore, there are two crucial aspects to determine for interval based classifiers: how to deal with the increase in the dimension of the feature space and what to do with each interval. Rodriguez *et al.* [45] were the first to develop a interval based algorithm and addressed the first issue by using intervals whose lengths were powers of two, and the second, by calculating binary features on each interval based on threshold rules on the interval mean and standard deviation. The resulting feature set is used to train a support vector machine. As the latter considers several divisions by means of intervals of different size, the temporal substructures of different length can be extracted from the time series. That is, the algorithm is not based on a single-length temporal information. Moreover, once the dataset has been mapped to the new feature space defined by the interval features, any conventional classifier can be used for classification.

To overcome the problem of the immense feature space, Deng *et al.* [46] proposed the **Time Series Forest** (TSF), a random forest approach that uses the summary statistics of each interval as features. In this approach three interval features are selected: mean, standard deviation and slope. Time series trees are created by a recursive top-down method, similar to standard decision tree algorithms. They use the random sampling strategy employed in the random forest, but it uses *Entrance* (entropy + distance) gain as the splitting criterion. In each of the nodes of a time series tree, $\mathcal{O}(\sqrt{m})$ (being m the length of the series) interval sizes are considered randomly, as well as $\mathcal{O}(\sqrt{m})$ starting points, and the interval feature with the highest *Entrance* gain as is used for splitting. Thus the size of the resulting feature space is $\mathcal{O}(m)$. Moreover, a node is considered a leaf when all the features have the same value or all instances belong to the same class, as it is impossible to get any entropy gain. A time series forest is composed by such time series trees. To classify unlabeled instances, it considers the votes of all the trees and selects the majority class. Instead of evaluating all the possible split points aiming to find the best information gain, a fixed number of evaluation points is pre-defined to speed up the classifier. Secondly, a refined splitting criteria is used to choose between features with equal information gain. According to this criteria, if two splittings have the same entropy gain, the one that is furthest from the nearest case is preferred.

If the order of the observations of the time series is changed, the values of the summary statistics of the intervals will change. Therefore, although the variables will remain unchanged, their respective values will. When those values are used as an splitting criteria for the trees of a time series forest, as they do not represent the shape of the original instances, the predictions of the forest will not be as accurate as in the case of the original dataset, as the split based on the values of the summary statistics might not be as representative. Nevertheless, as in the forest architectures the majority vote technique is used, it may happen that considering so many trees the considerable alterations that may happen in one tree may be neutralized by the others. The variation in the performance may therefore be softened.

2.3 Decision Tree Architectures

Several decision tree architecture based classifiers have been developed for time series classification; for example, the newly presented TSF [46] or the structure of the original algorithm that used shapelets as the splitting criteria [39].

Lucas *et al.* [47] proposed the **Proximity Forest** (PF), composed of highly randomized Proximity Trees (PT). Proximity Trees branch on the *proximity* of a query time series to a set of reference series (see Figure 26), instead of branching according to attribute values, as in the case of the TSF. *Proximity* is defined by a given time series similarity measure. As traditional decision trees, the construction process is recursive. At each node, as much branches as classes are represented on the data it receives are created. For that, r (user-defined parameter) different splittings are considered, by choosing an exemplar for each class and parametrized similarity measure in which the proximity is based both uniformly at random. The data is passed through those branches, by finding the corresponding closest exemplar, according to the given similarity measure. Once all the split candidates are created, the chosen split is the one that maximizes the difference between the Gini impurity of the parent node and the weighted sum of Gini impurities of children nodes. As in the TSF, if in a node all the series belong to the same class, it is considered a leaf node. To perform the classification of unlabelled time series, at each node, the distance from the query to each of the exemplars of the node is calculated using the distance measure of that node, passing down the branch to which exemplar is nearest to. The process is repeated until it reaches a leaf node, assigning the class of such leaf. In the Proximity Forest, the classification is performed by majority votes of the Proximity Trees is composed of.

Moreovoer, the PT with level 1 of depth is known as a Proximity Stump (PS) [48].

2.4 Ensemble Classifiers

There is a group of algorithms that provide high performances that have become very popular on TSC problems, which are based on ensembles of algorithms. Ensemble approaches are combinations of multiple base classifiers, whose individual outcomes are combined through a certain method in order to classify unlabeled instances. The base classifiers are given a weight aiming to maximize classification accuracy of the ensemble. Their corresponding complexity is that of the slowest base classifier. For example, TSF, TSBF and BOSS are homogeneous ensembles based on the same core classifier. Nevertheless, heterogeneous ensembles also exist.

The **Elastic Ensemble** (EE) [21] is an example of an heterogeneous ensemble, which is a combination of 1-NN classifiers that use elastic distance measures; such as, ED, DTW, DDTW, WDTW, LCSS, MSM etc.

Moreover, Bagnall *et al.* [49] proposed the meta ensemble **Collective of Transformation-based Ensembles** (COTE), which is a combination of classifiers in time, autocorrelation, power spectrum and shapelet domain. It is composed by 35 classifiers and, as in the previous method, the final response is created by weighted votes. Later, they proposed an improved version of COTE, called the **Hierarchical Vote Collective of Transformation-based Ensembles** (HIVE-COTE) [50]. The latter encapsulated the classifiers according to their domain, and a single probabilistic prediction is performed for each of those domains.

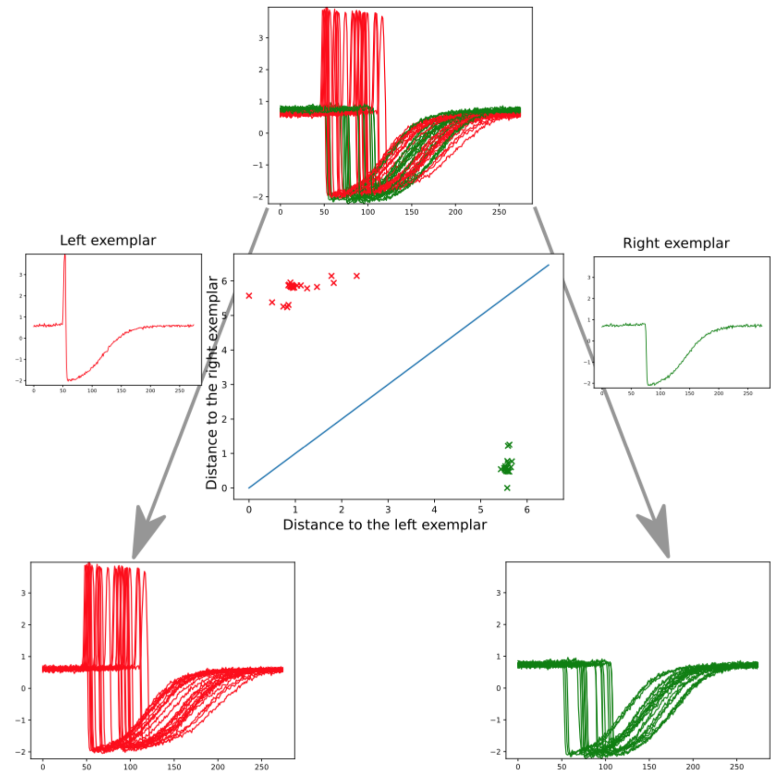


Figure 26: A graphical representation of a node for a proximity tree constructed for the `Trace` dataset. Image from [47].

Ensemble based approaches are computationally intensive and their use becomes unfeasible for big TSC problems. Therefore, although they have been found the be very accurate, they are impractical for real world problems.

In this algorithms, when the timestamps of the observations are randomly re-ordered, the performance of each of the constituent base classifiers will change as described in the previous sections, and therefore, the overall performance of the ensemble is expected to change also. It may happen, that as the constituents change in different way, and the final output depends on the vote of the majority, the change on the performance is softened by the compensation of the different biasing of the algorithms.

2.5 Deep Learning Approaches

Several deep learning architectures have been proposed for the classification problem of time series. Those networks are designed to learn hierarchical representations of the input data. A deep network is composed of layers (parametric functions), where each of them is considered a representation of the input domain. Each of the layers is formed by neurons, elementary units that compute one element of the layer's output. These architectures are trained efficiently to learn hidden discriminative features from raw time series data.

The **Multi Layer Perceptron** (MLP) was proposed by Wang *et al.* [51] as a baseline architecture for TSC. The network is composed by 4 layers, where each of them is fully connected (FC) to the output of the previous layer (see Figure 27). The final layer is a softmax layer, fully connected to the previous layer and having a number of neurons equal to the number of classes of the dataset. The other three layers are

FC layers formed by 500 neurons with the ReLU function as the activation function. Moreover each layer is preceded by a dropout operation to prevent overfitting.

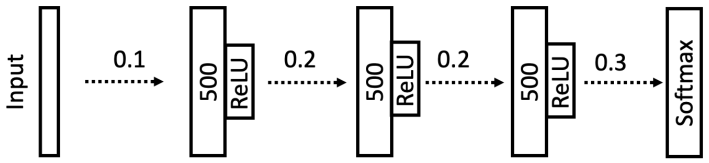


Figure 27: The architecture of the MLP. The dashed lines indicate the operation of dropout. Image from [51].

One of the drawbacks of the MLP is that the elements located in different time stamps are treated independently, assigning a weight to each of the time stamps. Therefore, it does not consider the temporal information, and altering the order of the instances will not affect its performance.

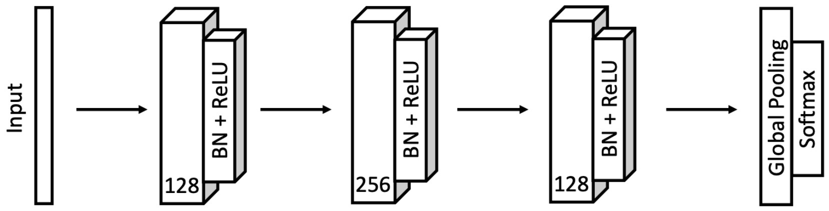


Figure 28: The architecture of the FCN. Image from [51].

The use of **Fully Convolutional Neural Networks** (FCNs) for time series classification was first proposed by Wang *et al.* [51]. FCNs are convolutional networks that do not contain pooling layers, and the traditional FC layer is replaced by a Global Average Pooling (GAP) layer (see Figure 28), which reduces drastically the number of parameters. The network is composed of three convolutional blocks. Each of them has a convolution, a batch normalization and a ReLU activation function. The result of the third convolutional block is averaged over the whole time dimension which corresponds to the GAP layer. Finally, a traditional softmax layer is fully connected to the output of the GAP layer. This approach performs several convolutions in the input series, which enables to learn discriminative features for classification. Those features depend on the temporal information, thus randomly altering the order of the instances will change the performance of this deep learning structure. The convolutions are dependant of the number located on a given timestamp and its adjacent values. Therefore, when the temporal order of the instances is shuffled, the results of those convolutions change. Therefore, the learning information will no longer be representative of the original temporal information. Moreover, the GAP layer creates an activation map, which provides the interpretation of class-specific regions and highlights the most discriminative subsequences.

The third model proposed by Wang *et al.* [51] is a deep **Residual Network** (ResNet). The network has 3 residual blocks followed by a GAP layer that averages the time series across the time dimension, and a final softmax classifier, whose number of neurons is equal to the number of classes in the problem (see Figure 29). Each residual block is composed by 3 convolutions, a batch normalization operation and the ReLU activation function. Moreover, it has short residual connections between consecutive convolutional layers. As in the previous architecture, convolutional layers are used, endowing the network with sensitivity to the temporal information and the GAP layer 's activation map provides information discriminative regions and subsequences. The performance of the ResNet architecture is better for longer and more complex data, while in easy databases is outperformed by the FCN.

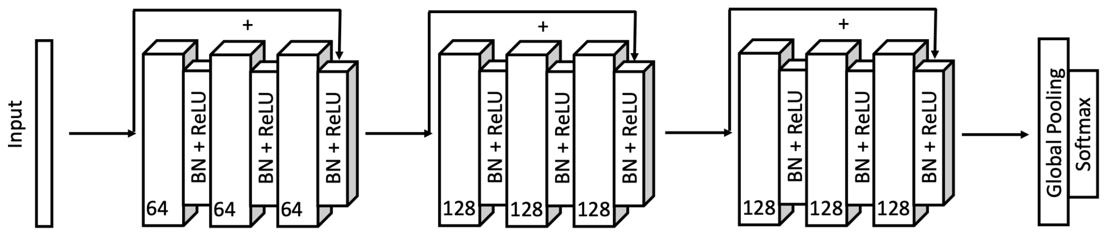


Figure 29: The architecture of the Residual Network. Image from [51].

Both FCN and ResNet architectures use batch normalization, providing great generalization capability and time efficiency to the networks.

Serrà *et al.* [52] developed a hybrid deep CNN called the **Encoder**. This network converts a variable-length time series to fixed length low-dimensional representations, which are learned to generalize the classification of unseen instances. It was inspired in the FCN, but replacing the GAP layer with an attention layer. The first three layers of the network are convolutional. Each convolution is followed by an instance normalization operation, whose output is fed to the PReLU activation function. The output of PReLU is followed by a dropout operation and a final max pooling. The third convolutional layer is fed to an attention mechanism that identifies which parts of the time series (in the time domain) are important for classification. Finally, a traditional softmax classifier is fully connected to the latter layer, having as many neurons as classes are in the dataset. This architecture requires from few training iterations. As in the previous two architectures, the convolutional layers capture the temporal correlation of the observations and the attention layer highlights the most discriminative regions of time series.

Cui *et al.* [53] proposed the **Multi-scale Convolutional Neural Network** (MCNN), which extracts features in different time scale and frequencies, resulting on a superior feature representation. This architecture is formed by two convolutions and max poolings, a FC layer and a final softmax layer. Nevertheless, the preprocessing is very complex (see Figure 30). Firstly, in the transformation stage, subsequences are extracted (data augmentation) by a Window Slicing method, and they undergo three transformations: identity mapping, down-sampling in the time domain to generate sketches of time series in different time scales and spectral transformations in frequency domain using low frequency filters with multiple degrees of smoothness to lower the effect of the noise (see Figure 30). Thus the univariate time series is transformed to a multivariate time series. Those transformations are followed by local convolutional stages to extracts features from each branch, using the sigmoid function as activation function. The convolutions for each branch are independent. Finally, there is a full convolutional stage, in which all the extracted features and convolutional layers are concatenated for the final output. This architecture requires from data augmentation to avoid overfitting whenever it is used in small datasets.

As stated before, the convolutional layers learn complex feature representations in which the intrinsic temporal information of the instances is reflected. Moreover, this particular architecture extracts features in frequency and time domains at different scales, expanding the temporal information considered for classification. A convolutional layer of a fixed scale can only detect local patterns of a given substructure. Nonetheless, using multi-scale convolutions the layers can learn more complex structures, equivalent to those cases where multiple window lengths were used to extract the representative features.

Time Le-Net (t-LeNet) was proposed by Le Guennec *et al.* [54], which contains two convolutions followed by a FC layer and a softmax classifier. For both convolutions the ReLU activation function is used.

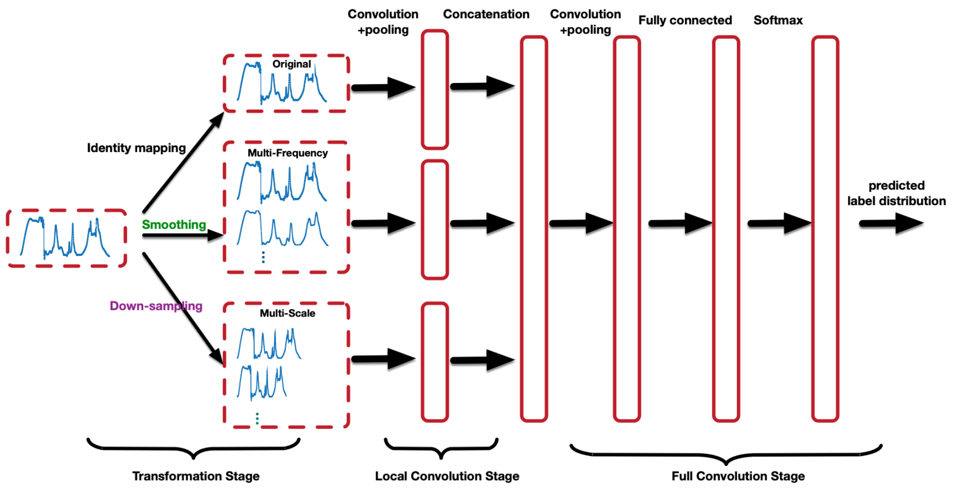


Figure 30: Architecture of the MCNN. Image from [53].

The convolutional blocks are followed by a non-linear FC layer of 500 neurons, each one using the ReLU activation function. Finally, it has a softmax classifier with a number of neurons equal to the number of classes of the problem. To avoid overfitting, this approach combines two data augmentation techniques: Window Slicing and Window Warping, which squeezes or dilates the time series. That is, it warps a randomly selected slice of a time series by speeding it up or down (see Figure 31). After that, all the slices are made equal length by a slicing window. This data augmentation technique is more time specific. Moreover, they proposed dataset mixing to learn the convolutional part of the network in an unsupervised manner. Therefore, the complete network is learned on a semi-supervised way. As in the previous cases, the presence of convolutional layers implies the network is sensitive to the temporal order of the instances.

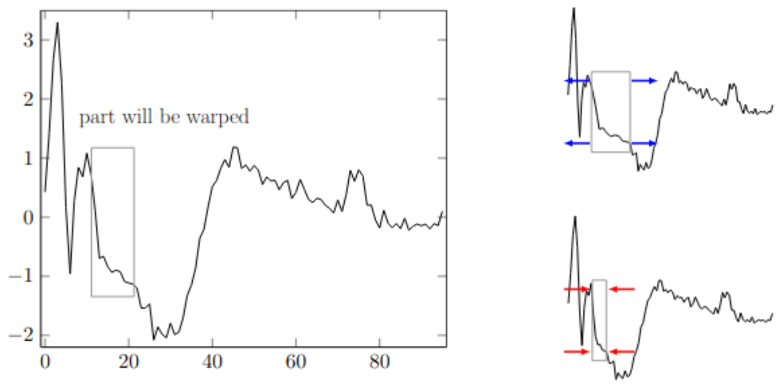


Figure 31: An example of the data augmentation technique used in the t-leNet network. Image taken from [54].

Time Convolutional Neural Network (Time-CNN) was proposed by Zhao *et al.* [55]. This approach has some special characteristics: it uses the mean square error (MSE) instead of the categorical loss-entropy function, as a final layer it has a traditional FC layer with sigmoid as activation function instead of the softmax classifier, and it uses a local average pooling operation instead of local max pooling. The network has two convolutional layers, with sigmoid as activation function, followed by a local average pooling operation (see Figure 32). The output layer is a FC layer with a number of neurons equal to the number of classes of the dataset. As the final classifier is fully connected to the output of the second convolution, it removes the GAP layer without replacing it with a FC non-linear layer. In this architectures, the input instances must be of the same length. Moreover, the convolutional layers endow the network with sensitivity to the temporal order of the instances.

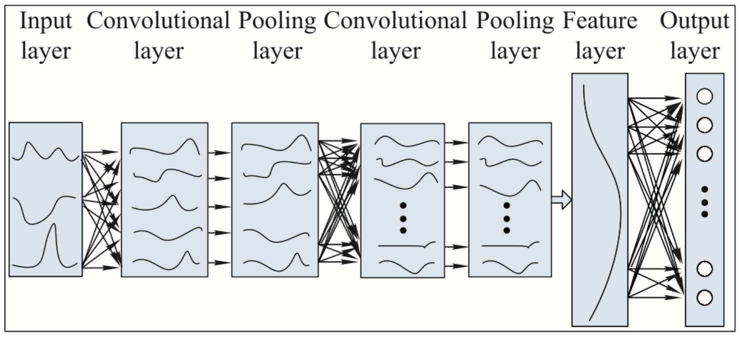


Figure 32: Architecture of the time-CNN network, in the case of a three-variate time series classification, via [55].

Tanisaro and Heidemann [56] proposed the **Time Warping Invariant Echo State Network**, which is a non-convolutional recurrent architecture. For each time stamp of a time series, the reservoir space is used to project the element to a higher dimensional space. Then for each element, a Ridge Classifier [57] is trained to predict the class of each time series element. With that, the trained Ridge Classifier, will output a probability distribution over the classes in the dataset for each of the elements of the time series. After that, a *posteriori* probability of each class is averaged over all the elements of a time series, to which the label of the class having the maximum average probability is assigned. This network has the capability to capture linear and non-linear patterns of the instances without any data transformation and dimension reduction processes. As this architecture is based on a recurrent procedure, the order in which the values of the time stamps are inserted affects the learning process of the algorithm. This network is highly dependant of the initial values. Thus as changing the temporal order will affect on which information is inserted first, it will have an impact in the performance of the network.

In convolutional neural network (CNN) architectures, the convolutional filters represent the local attributes of an input time series or feature map. The size of the filters determines the size of the temporal substructures captured by the extracted features. If the filter size is too small, it cannot well represent the representative features or waveforms. Contrarily, if the filter is too large, the local patterns will not be clearly represented on the extracted features.

Moreover, the number of the convolutional filters determines the number of extracted features for classification. Therefore, the higher the number of filters, the more features are extracted. Nevertheless, increasing the number of filters makes the network computationally more expensive to train. Therefore, once the most discriminative features are considered in the network architecture, adding more filters is helpless.

Regarding the pooling method, the larger the pooling size, the better performance it is obtained in the dimension reduction. Nonetheless, more information is lost in the transformation.

In general, CNN architectures tend to overfit less that FCN, because they tend to have much fewer weights to optimize than FC models. Nevertheless, they require from a huge amount of training data to be efficient. Henceforth, they tend to use data augmentation techniques, specially with small datasets. Those techniques build synthetic data by transforming existing labelled samples. Table 5 sums up all the advantages and drawbacks of the presented architectures. Moreover, Table 6 groups the deep learning architectures of this section depending on whether variable-length instances can be considered as input.

algorithm	advantages	drawbacks
MLP	generalization	non sensitive to temporal information
FCN	generalization, efficiency, interpretability	more overfitting than CNN
ResNet	generalization, efficiency, interpretability, better for long and complex datasets	worse than FCN in small datasets
Encoder	adaptation facility, efficient, few training iterations, interpretability	data augmentation to avoid overfitting in small datasets
MCNN	computational efficiency, noise reduction, different time scales and frequencies	data augmentation to avoid overfitting in small datasets
t-leNet	great efficiency in long and complex datasets, more time-specific data augmentation technique	data augmentation and datasets mixing to avoid overfitting in small datasets
time CNN	noise tolerance	time consuming training process
Echo	computationally cheap, simple learning mechanism, fast testing	

Table 5: The main advantages and drawbacks of the specific deep learning architectures for TS.

same length	variable length
MLP	Encoder
FCN	MCNN
ResNet	Echo
t-leNet	
time-CNN	

Table 6: The grouping of the deep learning architectures presented in this section depending on whether they classify variable length instances.

CNN architectures can also be used to extract representative features, and perform the classification with a conventional ML classifier, such as, SVM.

Regarding the sensitivity of the networks to the intrinsic temporal information of the TSC problems, all the architectures presented in this section are sensitive to the temporal information of the observations except for the MLP. The presence of convolutional layers endow the networks with temporal sensitivity, as the result of those convolutional layers depend on temporal substructures of the observations. Furthermore, multi-scale convolutional layers enable the network to capture and learn more complex temporal structures. Additionaly, the GAP layer or the attention layer that some of those CNN architectures have, provide information about the most discriminant regions and subsequences of TS. Moreover, the learning processes of networks with recurrent structures strongly depend on the initial values of the observations, thus their performance will be affected by the alteration of the temporal order of the instances.

3 Temporal Sensitivity of Specific Time Series Classifiers

Time series are a particular type of data where the observations are temporally correlated. In general, conventional ML classifiers happened to be insufficient to classify time series, thus specific classifiers were developed, which consider the particularity of TS data for the classification. The empirical study of this section analyses the sensitivity of several specific TS classifiers to the temporal order of the instances. That is, how much their performances vary when the time stamps of the instances are re-ordered randomly.

3.1 Methodological Setup

To examine how sensitive a classifier is to the temporal order of the instances, the classifier will be applied to several TSC problems and variants of those databases. Those variants will be created by randomly altering the temporal order of the instances of the original database. If a database is sensitive to the order of the observations, re-ordering the time stamps randomly will have a considerable effect on the performance of the algorithm. Contrarily, if the performance of a specific TS classifier appears to be robust against the random alteration of the order of the observations, its sensitivity to the temporal correlation of the instances is considered to be low.

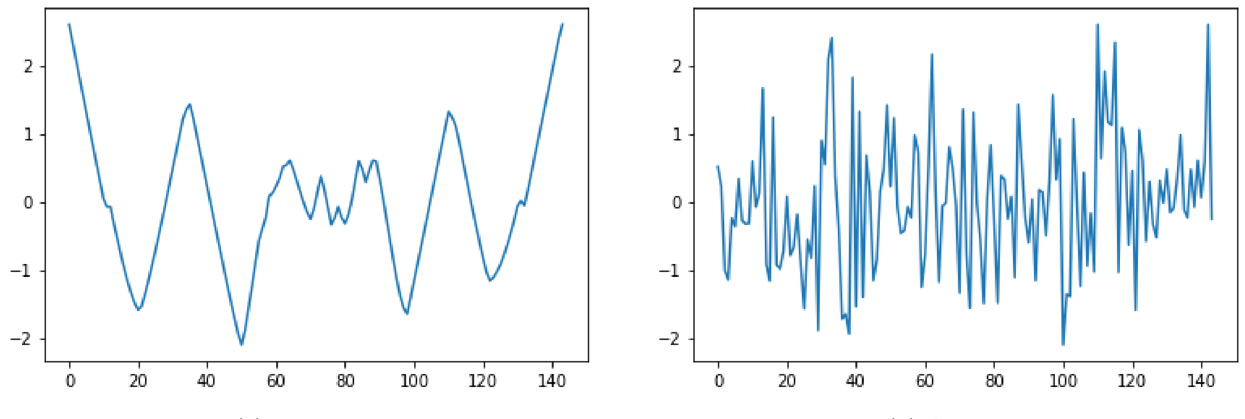
Each of the databases considered in this study will be randomized 1000 times according to a certain set of randomizing operators, and each of the randomized variants will undergo a classification process with all of the distinct classifiers. The randomizing operators are permutations of size N, created uniformly at random, being N the length of the instances. Therefore the subset of randomizing operators ($\mathcal{R}$) is defined as follows:

$$\mathcal{R} = \{r_1, ..., r_{1000}\}$$

where:

$$r_i = (\pi_1 \pi_2 \; ... \; \pi_n) \qquad \pi_i \in \{0, 1, ..., N-1\} \text{ and } \pi_i \neq \pi_j \; \forall i \neq j$$

where N is the length of the instances of the database under study. The permutation r_i describes the new order of the observations. When a random permutation is applied to the time stamps, the resulting instances will contain the same pointwise information as the originals, but with a different temporal order (see Figure 33). Therefore, the original shape of the instances, as well as the temporal correlation they represented, will be lost.



(a) Original

(b) Altered

Figure 33: One of the instances of the `Plane` dataset [38] (a) without any temporal alteration and (b) with the order of the observations randomly shuffled.

The databases considered in this experiment need to have instances of the same length, so that the random re-ordering of the timestamps is the same on every instance of the dataset.

In the case where a classification algorithm considers the temporal information as a discriminant factor for classification, randomly altering the temporal order of the observations will affect its performance, which in this case is measured by the accuracy obtained on the classification of the instances of the test set. Thus when a certain classifier is applied to a given dataset and all of its randomized variants, the resulting accuracy values can be considered as an empirical probability distribution (see Figure 34). If the resulting distribution is wide, the performance of the classifier has varied considerably when the order of the instances is randomly altered, thus the classifier has a high sensitivity to the order. On the other hand, sharp distributions indicate robust performances against the alteration of temporal order of the observations; thus, in those cases, the sensitivity to the temporal order of the instances is considered to be low. The sharpness of the empirical accuracy distribution will be quantified by the standard deviation.

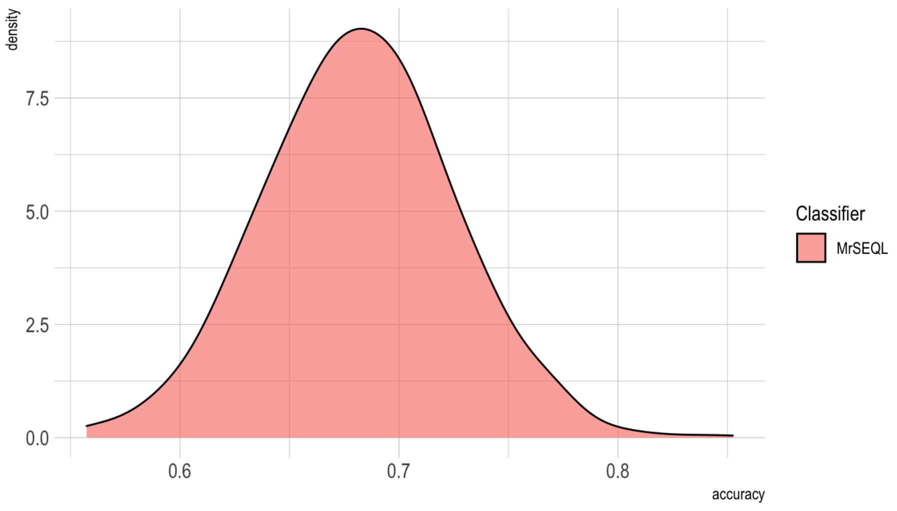


Figure 34: An example of the distribution of the accuracy values obtained when the MrSEQL classifier is applied to the `Lightning2` dataset and its 1000 randomized variants.

Let C_i be the different TS classifiers under analysis $(i = 1, ..., K)$ and σ_i the standard deviations of the distributions of the obtained accuracy values. If $\sigma_i > \sigma_j$, then the algorithm C_i will be considered to be more sensitive to the temporal order of the instances than the C_j algorithm.

For each of the considered TSC problems, a ranking will be performed subject to the standard deviations of the empirical distributions (see Figure 34) obtained with the classifiers under study. That is, the classifier with the highest value of the standard deviation will be in the first position of the ranking related to that dataset, and that with the lowest in the last. The rankings obtained for each of the datasets will be combined to perform a joint ranking. This ranking will offer a general overview about how much does a classifier's performance vary when the temporal information of a TSC problem is re-ordered randomly.

Moreover, pairs of classifiers will be considered, and the standard deviations of their corresponding distributions will be compared, aiming to study whether statistically significant differences exist between them, using the Wilcoxon signed-rank test [58]. The latter is a non-parametric statistical hypothesis test that compares a pair of dependent samples to examine whether their corresponding populations belong to the same distribution. In this particular case, it will indicate whether two sets of standard deviation values related to two different classifiers present statistically significant differences. Each of the values of those sets represents the standard deviation value of the empirical distribution obtained when that classifier is applied to a TSC problem and its randomized variants (see Figure 34). As multiple classifiers are considered in the analysis, the likelihood of incorrectly rejecting a null hypothesis increases. To hinder such issue, the Bonferroni correction [59] has been applied in combination with the Wilcoxon signed-rank test.

Regarding the results of the statistical test, in the cases where the p-value is above 0.05, it is considered there is not enough evidence to reject the null hypothesis. In that case, the difference between the samples is believed to follow a symmetric distribution around zero. That is, the standard deviations recorded on the empirical distributions of the corresponding classifiers do not present statistically significant differences. On the other hand, for the combinations where p-values below 0.05 are obtained, the null hypothesis is discarded.

3.2 Databases

In the analysis of the temporal sensitivity regarding the specific time series classifiers, 31 datasets have been used. Those datasets, which represent labeled univariate time series, have been obtained from the UCR time series database archive [38]. The latter contains the vast majority of publicly available synthetic and real time series databases, and it is the most common benchmark used for performing evaluations in studies about time series analysis.

The TS databases of the UCR archive belong to a wide variety of information domains, such as, image, sensorial information, motion, ECG, etc. The ideal procedure would be to take a considerable variety of datasets and to keep balance among the different domains of time series classification problems. Nevertheless, the computational cost required for the experiment is considerable, as for each dataset 1000 classification tasks have to be performed in the case of every classifier considered in the experiment. Henceforth, the range of possible datasets for which the analysis is feasible happens to be narrow. Despite the mentioned drawback, among the resulting 31 datasets, a variety of domains are represented (see Table 7).

As mentioned in the previous section, among the chosen TSC problems, the instances that belong to the same dataset have the same length. Moreover, it must be outlined that in this case only univariate time series have been considered, as not all the specific classifiers are designed to deal with multiple dimensions.

domain	num. of datasets	datasets
Image	5	ArrowHead, DistalPhalanxOutlineAgeGroup, DistalPhalanxTW, FaceFour, Herring,
Simulated	4	BME, CBF, ShapeletSim, UMD
Sensor	7	Car, ItalyPowerDemand, Lightning2, Plane, SonyAIBORobotSurface1, SonyAIBORobotSurface2, Trace
Traffic	1	Chinatown
Spectro	4	Coffee, Ham, Meat, Wine
Motion	6	GunPoint, GunPointAgeSpan, GunPointMaleVersusFemale, GunPointOldVersusYoung, ToeSegmentation1, ToeSegmentation2
ECG	3	ECG200, ECGFiveDays, TwoLeadECG
Device	1	PowerCons

Table 7: Information concerning the chosen set of TSC problems for the experimentation: domain, number of datasets and name.

3.3 Specific Time Series Classifiers

The purpose of this work was to analyse as many TS specific classifiers as possible, but the computational cost required for the designed experiment made the study of some of them unfeasible. Each classifier needs to perform 1001 classification tasks for each of the considered datasets. Henceforth, only those 16 classifiers that required less than three weeks to perform the 1001 classifications related to a dataset on the cluster were considered.

The specific TS classifiers for which the temporal sensitivity have been studied are the ones grouped in the Table 8.

In those classifiers where parameters needed to be defined, specially in dictionary-based algorithms where the window length, the alphabet size and the words length needed to be specified, those parameters have taken the default values given by the software. Table 9 shows the parameter settings for each of the classifiers.

raw time series distance based		feature extraction	
(dis)similarity matrix + 1-NN	decision tree architectures	dictionary based	interval based
1-NN-DTW (C_1) 1-NN-DDTW (C_2) 1-NN-WDTW (C_3) 1-NN-MSM (C_4) 1-NN-LCSS (C_5)	Proximity Tree (C_6) Proximity Stump (C_7)	Individual BOSS (C_8) BOSSEnsemble (C_9) cBOSS (C_{10}) WEASEL (C_{11}) WEASEL + MUSE (C_{12}) Individual TDE (C_{13}) TDE (C_{14}) MrSEQL (C_{15})	TSF (C_{16})

Table 8: The studied specific TS classifiers grouped according their procedure to perform the classification.

algorithm	parameters

Individual BOSS	window_size = 10
	words_length = 8
	alphabet_size = 4

BOSS Ensemble	max_ensemble_size = 500
	max_window_length = length of the observations
	min_window_length = 10

Contractable BOSS	n_parameters_sample = 250
	max_ensemble_size = 500
	max_window_length = length of the observations
	min_window_length = 10

Individual TDE	window_size = 10
	words_length = 8
	alphabet_size = 4

TDE	n_parameters_sample = 250
	max_ensemble_size = 500
	max_window_length = length of the observations
	min_window_length = 10

TSF	n_estimators = 500

Table 9: The default settings of those classifiers under study for which parameters need to be defined.

3.4 Performance Measures

As mentioned before, the performance of a classifier in a certain TSC problem (or any of the corresponding randomized variants) will be evaluated by the accuracy metric; in fact, the accuracy value obtained when performing the classification of the test set will be considered. The datasets of the UCR web archive regarding TSC problems are already split in two sets: one for training, and one for testing. The former will be used to fit the given TS classifier, and the latter for the performance evaluation. It is important to outline that the testing set must undergo the same randomizing operation as the training set of instances, so that the instances belonging to the test set have the same order of timestamps as those instances in the training set.

3.5 Software

All the empirical analysis has been carried out using python. The specific TS classifiers and the TSC problems datasets were provided by the `sktime` package [60]. To perform the basic mathematical operations, such as calculating the standard deviation of the distributions, the `numpy` package [61] has been used.

3.6 Results

For each TSC problem considered, a ranking has been performed regarding the standard deviation values obtained with each of the classifiers (Table 10).

According to the rankings of classifiers obtained for each of the datasets (Table 10), a joint ranking has been performed where the classifiers are listed subject to their sensitivity to the temporal order of the instances, being the first the most sensitive:

1. Proximity stump (C_7)
2. Proximity tree (C_6)
3. MrSEQL (C_{15})
4. BOSSEnsemble (C_9)
5. IndividualBOSS (C_8)
6. TDE (C_{14})
7. Individual TDE (C_{13})
8. 1-NN-LCSS (C_5)
9. cBOSS and WEASEL (C_{10} and C_{11})
10. WEASEL + MUSE (C_{12})
11. TSF (C_{16})
12. 1-NN-DDTW (C_2)
13. 1-NN-DTW (C_1)
14. 1-NN-WDTW (C_3)
15. 1-NN-MSM (C_4)

The most sensitive classifier has been Proximity stump (C_7), which has produced the empirical distributions with the highest standard deviation values for 26 datasets of the 31 considered. The second has been the Proximity tree (C_6) which has been in the top6 for all the datasets.

The classifier with the lowest sensitivity to the temporal ordering has been 1-NN-MSM (C_4), having the distribution with the lowest standard deviation for 25 datasets among the 31 considered.

Dataset	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}
AH	13	12	13	16	15	**1**	2	6	8	5	10	10	6	4	3	9
BME	3	13	4	16	15	**1**	2	11	5	7	12	7	10	7	5	14
Car	15	13	15	14	10	2	**1**	8	8	3	6	12	10	5	3	6
CBF	10	13	13	16	4	2	**1**	12	3	6	9	8	11	5	7	15
CT	14	12	15	13	2	3	**1**	4	6	10	8	7	5	11	9	16
Coffee	10	9	10	15	15	2	**1**	3	5	6	13	14	4	7	8	10
DPAG	14	10	14	16	5	2	**1**	4	6	9	10	12	3	8	6	13
DPTW	14	10	14	16	5	2	**1**	3	5	8	12	13	3	8	5	10
ECG200	13	13	15	16	7	2	**1**	7	3	11	4	12	9	9	4	6
ECG5D	14	13	15	16	3	2	**1**	8	8	11	10	5	6	6	4	12
Face4	4	3	6	16	5	2	**1**	10	10	9	12	14	12	7	14	7
GP	11	4	11	16	2	3	**1**	7	7	7	6	14	7	4	11	15
GPAS	10	6	10	10	2	3	**1**	6	6	10	10	15	9	15	4	4
GPMVF	13	15	13	10	2	6	**1**	6	5	10	6	3	6	10	4	16
GPOVY	14	13	14	14	12	2	**1**	4	9	7	4	9	6	7	3	11
Ham	13	12	14	15	16	2	**1**	7	3	5	6	11	7	4	7	10
Herring	7	3	7	16	2	4	**1**	5	10	10	13	13	6	12	7	15
ItalyPD	13	15	13	16	2	3	**1**	4	11	12	6	6	6	9	4	10
Light2	12	10	10	16	15	**1**	**1**	12	3	4	7	7	14	6	4	7
Meat	12	12	12	12	12	2	**1**	3	6	8	10	11	4	7	5	9
Plane	13	13	13	13	3	2	**1**	11	6	6	9	9	11	6	5	4
PCons	13	7	14	16	2	4	**1**	4	3	9	11	11	6	8	9	15
SSim	4	6	11	4	16	6	11	6	15	11	**1**	6	6	11	**1**	**1**
SonyAI1	11	9	12	16	2	4	**1**	13	8	15	6	5	13	10	3	7
SonyAI2	14	9	15	16	2	3	**1**	9	7	5	6	11	11	4	8	13
ToeSeg1	12	11	15	16	14	2	**1**	9	5	10	3	4	7	7	5	13
ToeSeg2	14	13	15	16	9	5	**1**	10	4	6	8	2	12	3	7	11
Trace	14	12	13	16	3	2	**1**	6	8	8	5	11	10	7	4	15
TLECG	14	13	14	16	8	2	**1**	11	5	7	8	11	10	5	4	3
UMD	10	13	9	16	15	**1**	2	8	3	3	12	11	6	5	7	14
Wine	13	12	13	15	15	**1**	7	4	8	11	6	2	3	8	4	8

Table 10: Ranking of the standard deviation values regarding each of the classifiers for the considered databases, obtained by the randomizing operators applied to it.

The most sensitive classifiers have decision tree structures. The intermediate positions of the ranking are occupied by dictionary based classifiers and the 1-NN-LCSS. The next classifier having the highest sensitivity is the TSF. Finally, the least sensitive algorithms have happened to be those that perform a (dis)similarity matrix based on distance measures between the whole raw time series, and perform the classification by the 1-NN classifier (except for the 1-NN-LCSS algorithm).

The obtained results have been studied more in detail by considering a statistical test based on the populations of standard deviation values related to each of the classifiers. In fact, there is a set of 31 standard deviation values related to each of the classifiers, where the values are obtained when the classifier is applied to each of the 31 TSC problems (and its corresponding randomized variants).

Applying the Wilcoxon signed-rank test with the Bonferroni correction to pairs of populations of standard deviation values of distributions related to the classifiers, the resulting p-values are shown in Figure 35. The p-values above 0.05 are colored in pink, while those below 0.05 are colored in green.

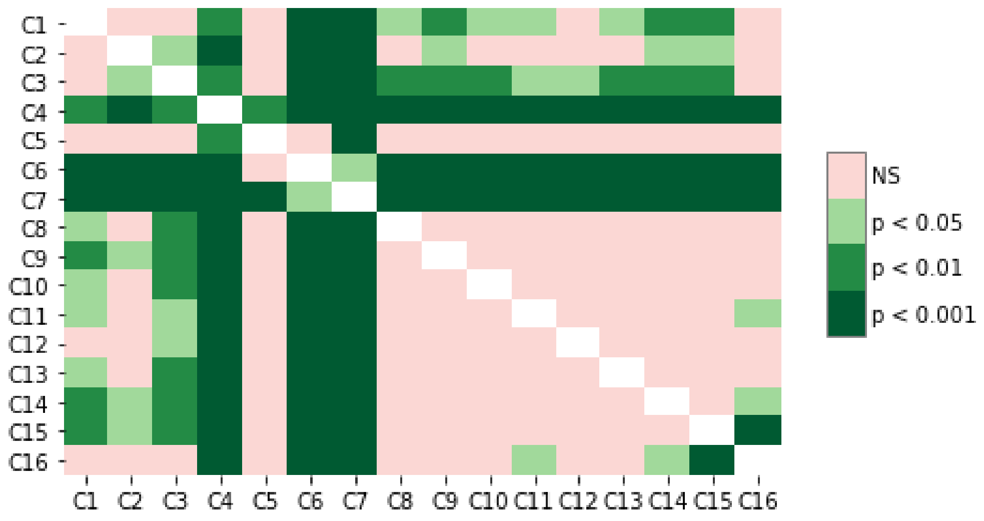


Figure 35: The p-values obtained when the Wilcoxon signed-rank test [58] with the Bonferroni correction [59] is applied to pairs of samples of standard deviation values related to each of the considered classifier. The light pink color refer to comparisons with p-values above 0.05; that is, the null hypothesis is not rejected. On the other hand, the green colors denote p-values below 0.05, getting darker for lower values. In these cases, the null hypothesis is rejected.

When comparing the standard deviation sets of the empirical accuracy distributions of the classifiers that belong to the same group, that is, if comparisons between those algorithms belonging to the same columns of Table 8 are considered, among the classifiers that calculate the similarity of the whole raw time series then classify them using the Nearest Neighbour classifier (C_1-C_5), the empirical distributions obtained with the 1-NN-MSM classifier presents statistically significant differences with the rest of such algorithms. Moreover, 1-NN-DDTW and 1-NN-WDTW have also significant differences regarding the empirical distributions. The decision tree structures based on the proximity (C_6 and C_7), obtain significantly distinct empirical distributions. Furthermore, regarding the dictionary based classifiers (C_8-C_{15}), the null hypothesis is not rejected in any case, when they are compared to each other.

If the empirical distributions obtained with classifiers belonging to different groups are compared, in general terms, almost all the classifiers that calculate the similarity of the whole raw time series then classify them using the 1-NN classifier present statistically significant differences with almost all the algorithms of the rest of the groups except for the 1-NN-DDTW (C_2) and 1-NN-LCSS (C_5) algorithms. The decision tree structures based on the proximity, obtained significantly different standard deviations on the empirical accuracy distributions when compared to the rest of the algorithms, except for the case where the proximity tree (C_6) algorithm is compared to the 1-NN-LCSS (C_5). In general, dictionary based algorithms have empirical distributions with standard deviations that are significantly different from the proximity based algorithms, 1-NN-MSM, 1-NN-DTW and 1-NN-DTW. Moreover, some dictionary based algorithms present also significant differences when compared to the TSF (C_{16}). This last classifier also provides empirical accuracy distributions with standard deviations different from 1-NN-MSM and both of the proximity based algorithms.

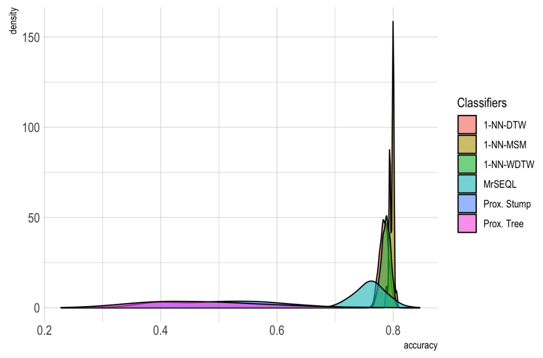
The classifiers that calculate a (dis)similarity matrix and then classify the instances by the 1-NN, are the least sensitive to the temporal order of the instances (except for the 1-NN-LCSS which is in the intermediate positions of the ranking). The latter is a consequence of the differences they have in their internal procedures compared to the rest of algorithms considered in this experiment: the feature based algorithms consider higher-level structures, and do not restrict themselves to local similarities. Therefore, the sensitivity is lower in the case where only local similarities are considered. The 1-NN-MSM is the least sensitive among them, differing significantly in sensitivity with the rest. Moreover, when it comes to the DTW distance, adding derivatives makes the classifier more sensitive, while adding weights decreases that sensitivity; being the standard deviations of the empirical distributions obtained with the 1-NN-DDTW and 1-NN-WDTW significantly different. Furthermore, the 1-NN-LCSS is the most sensitive classifier of this group of algorithms.

Among the dictionary based algorithms, MrSEQL have resulted the most sensitive. This classifier uses two different symbolic representations, one in time domain (SAX) and the other in frequency domain (SFA). Moreover, for each of the symbolic representations, it uses multiple resolutions. Therefore, it captures a wide amount of different temporal structures, and the classification is based on a more complete temporal information of the instances. Furthermore, regarding the classifiers based on the BOSS algorithm, BOSS Ensemble is the most sensitive among them, followed by the individual BOSS. The least sensitive among them is the contractable BOSS, and in between, the TDE and individual TDE algorithms are found. Even if the contractable BOSS, BOSS Ensemble and TDE are ensembles of BOSS algorithm, they use different procedures to create the ensemble and distinct classifiers, resulting in different behaviours of their performances when the temporal order of the instances is shuffled. The conventional BOSS Ensemble is more sensitive than the individual BOSS algorithm, but the contractable BOSS and TDE algorithms, which have a similar structure and weighting scheme, are less sensitive. Furthermore, the individual TDE algorithm, which consider a single BOSS classifier but choosing the parameters by a Gaussian process, is less sensitive to the individual BOSS. Besides, the algorithms WEASEL and WEASEL + MUSE are the least sensitive dictionary based algorithms. These algorithms, perform a feature selection before the classification, to choose the most discriminative temporal substructures. Thus, when selecting only the most discriminative features, the accuracy of the classifier varies less when the temporal order of the instances is altered randomly.

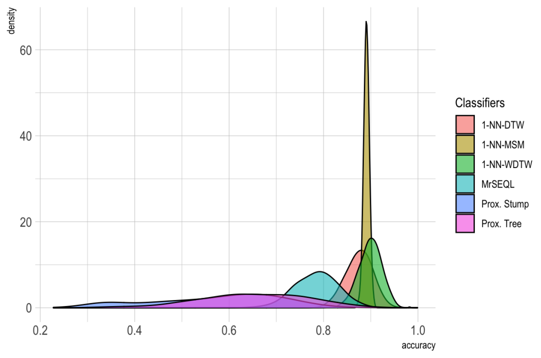
Single decision tree structures have been shown to be the most sensitive. That is, their performance have varied the most when the temporal order of the instances has changed. Moreover, according to these results, the deeper the decision tree, the less sensitive is the classifier to the random alteration of the observations. Furthermore, when multiple decision trees are combined in a forest, the performance of the classifier does not vary as much when the the time stamps of the instances are re-ordered randomly. As the TSF is a group of decision trees whose individual outputs are combined for the final prediction, the effect of the random alteration of the time stamps is softened as the biasing of the individual trees can be compensated.

In conclusion, considering higher-level structures instead of only restricting to local similarities, increases the sensitivity of a specific TS classifier. In dictionary based algorithms, considering symbolic representations in both time domain and frequency domain, and multiple resolutions for each, makes the classifier highly sensitive to the temporal order of the observations. On the other hand, creating ensembles of BOSS algorithms in the conventional way, increases the sensitivity of the classifier. Nonetheless, when the ensemble is created based on a random process, the sensitivity is decreased. Furthermore, when the parameters of the BOSS algorithm are chosen by a Gaussian process, the sensitivity of the algorithm is also decreased. In the case where a single multiple resolution symbolic representation is used, performing a feature selection, considering only the most discriminative substructures, decreases the sensitivity of the algorithm. Besides, for decision tree structures, considering deeper structures decreases the sensitivity of the algorithm, as well as combining multiple decision trees in a forest.

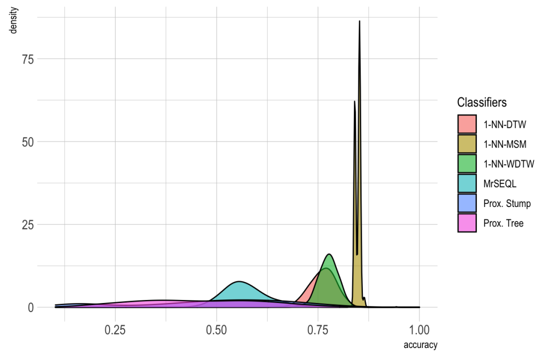
In order to see the different empirical accuracy distributions obtained with the most sensitive and the least sensitive classifiers, Figure 37 shows the obtained empirical accuracy distributions for some of the considered datasets, for the case of the three most and three least sensitive classifiers.



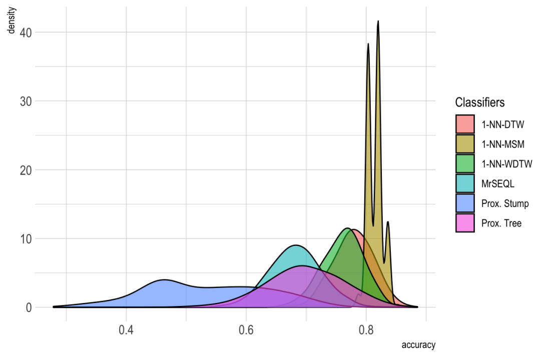
(a) ArrowHead



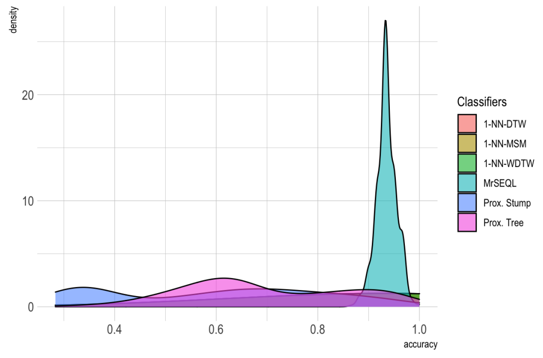
(b) CBF



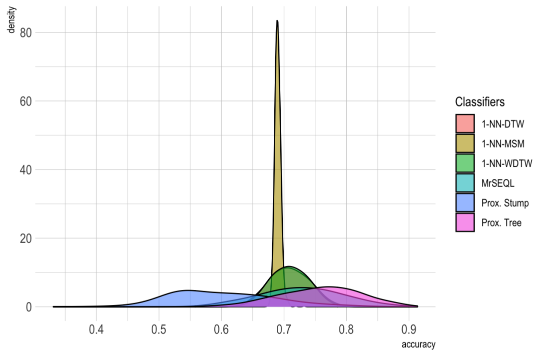
(c) FaceFour



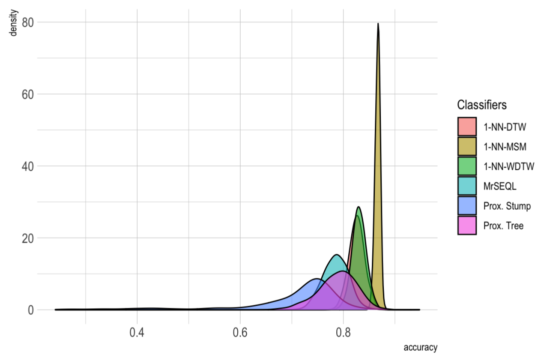
(d) Lightning2



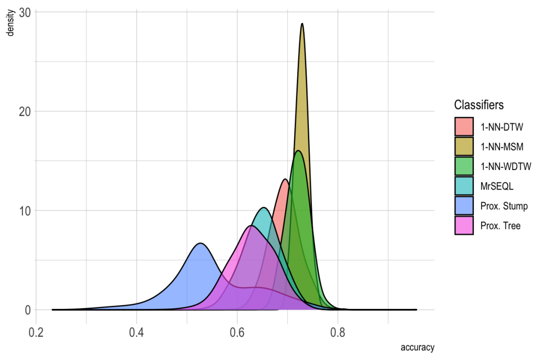
(e) Meat



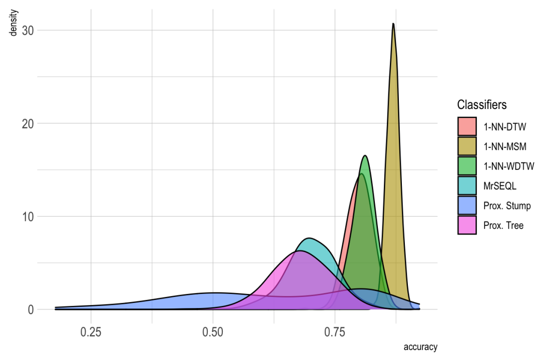
(f) SonyAIBORobotSurface1



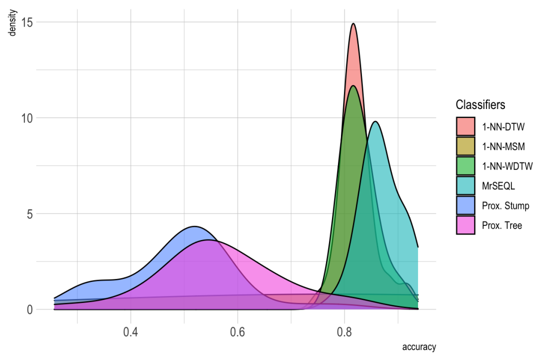
(g) SonyAIBORobotSurface2



(h) ToeSegmentation1



(i) ToeSegmentation2



(j) UMD

Figure 36: The accuracy distributions obtained with the three most sensitive and three least sensitive classifiers when they are applied to different TSC problems.

4 The Relevance of the Temporal Information in the Classification

TS is a special type of data where the observations are temporally correlated (or correlated in another ordered dimension), in contrast to the conventional classification problems, where the attributes do not follow any specific order. The nature of TS instances has lead to the assumption that TSC problems need to be studied by specific classifiers, which take heed of the intrinsic temporal information of the observations for the classification. Therefore, in principle, non-specific methods should not outperform or at least be competitive with specific methods, as they do not consider any temporal information for classification.

Nonetheless, for different TSC problems, the temporal order does not have the same relevance for classification. In those cases where the temporal order of the instances is discriminatory for classification, non-specific methods should be outperformed by specific methods. On the other hand, in the cases where the temporal order of the instances is not relevant for classification, non-specific classifiers could also be considered, as the problem would read as a conventional supervised classification problem. This section holds an empirical study to identify the relevance of the temporal order in different TSC problems.

Abanda *et al.* [62] performed a preliminary examination to study to which extent different TSC problems required from specific classifiers. They assumed that in the cases where the 1-NN-DTW is outperformed by 1-NN-EUC, the order of the observations is not discriminatory for the classification, as the latter is a non-specific classifier without any sensitivity to the temporal correlation of the observations. They also considered three additional non-specific conventional classifiers: Support Vector Machine (SVM), Naive Bayes (NB) and Random Forest (RF). They demonstrated that in most of the cases where the 1-NN-EUC classifier outperformed 1-NN-DTW, the standard classifiers were able to perform better than the 1-NN-EUC and 1-NN-DTW classifiers. However, in the opposite case, there were little cases where 1-NN-DTW was outperformed by any of the standard ML classifiers. That is, whenever they supposed the temporal order of the observations was not discriminatory for classifications, non-specific classifiers were found to outperform 1-NN-DTW, while in the opposite case, they were not found to be competitive in most of such problems.

The experiment of this section studies the relevance of the temporal order in a given TSC problem from a different perspective. In fact, it is based on the empirical accuracy distributions obtained in the previous section. Actually, this empirical study is based on the comparison between the accuracy obtained in the original dataset and the accuracy distribution obtained when the randomly altered variants are classified. If the accuracy of the original dataset is good enough compared to its randomized variants, the temporal order of the instances is considered to be relevant for the classification. Nonetheless, if there is a considerable number of randomized variants that offer better performances, the temporal information has low relevance for classification.

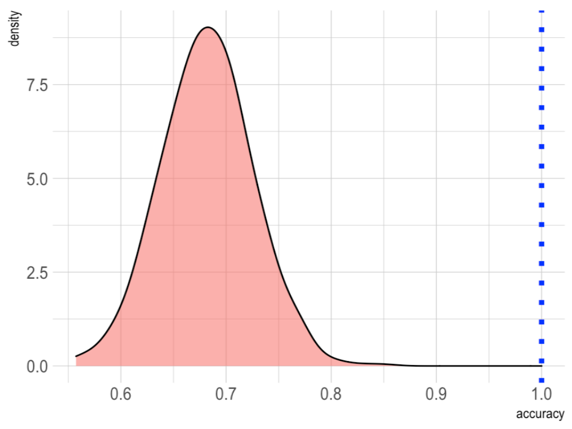
4.1 Metodological Setup

The relevance of the temporal order of the instances in a given TSC problem will be measured based on the distributions obtained in the previous section. In fact, the relevance of the temporal order of the instances will be estimated by calculating the percentile of the accuracy obtained with the original dataset in relation to the empirical accuracy distribution obtained with all its randomized variants (see Figure 37). When the temporal order of the instances is randomly shuffled and the temporal correlation of the original problem is completely lost, if the classifiers perform better or similar to when they are applied to the original dataset, the temporal order of the instances will be considered irrelevant, as loosing the temporal information has not have a negative impact on the performance of the classifier. Therefore, the higher the percentile of the accuracy value of the original dataset related to the empirical accuracy distribution obtained with all the variants, the more relevant the temporal order of the instances is in that particular TSC problem.

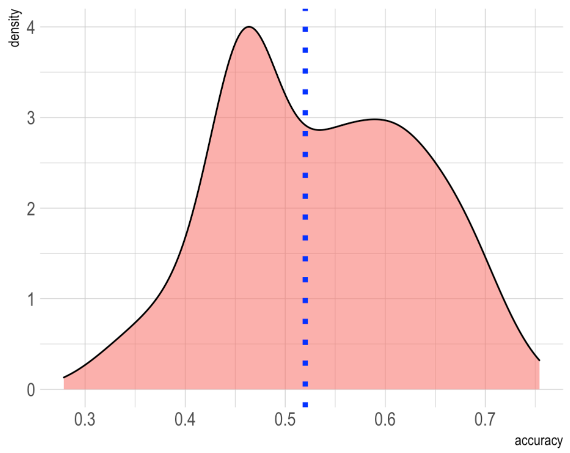
Each dataset has associated 16 accuracy distributions, each of them obtained with a different specific

TS classifier. For those empirical distributions, the position of the accuracy of the original datasets will be examined, by calculating the percentile of such value in relation to the distribution. The percentiles obtained for each of the datasets, will be averaged over the 16 classifiers. Therefore, each of the datasets will have associated a single value of the percentile, which will be the mean value of all the recorded percentile values for that dataset. In those dataset where the recorded mean value of the percentile is high, the temporal information is considered to be discriminant for the classification. Contrarily, if the mean value of the percentile is low, the temporal correlation of the instances will be considered to have low relevance for the classification.

Additionally, for the set of 16 percentile values related to a dataset (each of them obtained with a different specific TS classifier), the standard deviation value will be calculated. If in the case of a particular dataset the standard deviation value of the set is low, similar percentile values are obtained with the different classifiers. On the contrary, if the value of the standard deviation is high, there are considerable discrepancies between the percentiles obtained with the distinct specific classifiers.



(a)



(b)

Figure 37: An example of an accuracy distribution obtained when the temporal order of the instances is altered in a dataset. In (a) changing the temporal order of the observations has negatively affected the performance of the classifier, while in the case shown in (b) the alteration of the temporal order has improved the performance of the classifier in some cases. That is, in (b) the performance obtained with the original dataset is not good enough compared to the randomized variants, which do not contain the original temporal information. Both distributions have been obtained using the MrSEQL classifier. The disribution in (a) corresponds to the `Plane` dataset, and the one in (b) to the `Herring` dataset.

As the experiment hold in this section is based on the study carried out in the previous section, the datasets, classifiers and software used are those of the previous section (see Sections 3.2-3.5).

4.2 Results

In this experiment, for each dataset, we have calculated the percentile corresponding to the accuracy value of the original dataset in comparison with the empirical accuracy distribution obtained considering that dataset and its randomized variants. This procedure has been repeated for all the considered specific classifiers. Therefore, each of the datasets has a set of 16 percentile values, each of them related to a different specific TS classifier. The mean value and standard deviation of the sets recorded for each of the datasets are shown in Table 11.

Dataset	Mean	Standard deviation
Plane	99.24	2.17
Trace	98.28	3.81
ShapeletSim	96.22	12.13
GunPoint	95.63	12.63
ToeSeg2	89.54	27.78
Lightning2	89.22	22.51
ToeSeg1	89.14	29.68
SonyAI1	87.39	15.36
TwoLeadECG	86.73	31.45
DistPhalAG	85.84	30.87
SonyAI2	82.50	32.82
CBF	82.35	34.72
GunPointOVY	82.04	20.69
FaceFour	78.20	33.44
GunPointAS	78.10	35.88
Car	77.22	35.62
Wine	75.16	32.49
UMD	74.90	36.53
DistPhalTW	71.08	29.38
ECG200	70.40	36.48
GunPointMVF	67.89	38.40
PowerCons	64.09	35.83
ECGFiveDays	62.52	43.31
Ham	60.24	30.27
Coffee	57.87	34.40
ArrowHead	54.02	36.41
Meat	48.38	47.37
BME	43.66	38.32
ItalyPD	40.49	35.58
Chinatown	38.93	26.33
Herring	34.05	35.64

Table 11: The mean and standard deviation of the percentiles of the accuracy values of the original datasets regarding the empirical accuracy distribution obtained considering itself and its randomized variants. The top of the table shows the datasets with the highest average percentiles, and the bottom, those with the lowest mean percentile values.

As it can be seen from Table 11, there are 4 datasets that have significantly higher mean percentile values: `Plane`, `Trace`, `ShapeletSim` and `GunPoint`, all of them having mean percentile values over 90. Moreover, they have the lowest standard deviation values, meaning that the agreement between the different classifier is high. On the other hand, the datasets `Herring`, `Chinatown`, `ItalyPD`, `BME` and `Meat` have the lowest mean values for the percentiles, which are below 50. This datasets have higher standard deviation values, thus there are bigger discrepancies between the percentile values obtained with the different classifiers.

Table 11 shows a trend between the mean percentile value obtained for a dataset and the discrepancies between the different classifiers on the percentile value: in the case of the datasets with high mean percentile values, the classifiers are in high agreement, while for datasets with lower mean percentile values, the discrepancies between the classifiers tend to increase. This issue has been studied more in detail by means of a boxplot (see Figure 38). Each of the positions of the X axis refers to a dataset considered in the experiment, and they are ordered according to their mean percentile values: the datasets with the highest mean percentile values are located on the left. For each of the positions of the X axis, the values recorded along the Y axis correspond to the percentile values recorded for that dataset with the 16 specific classifiers considered. That is, the percentile corresponding to the accuracy value of the original dataset in comparison with the empirical accuracy distribution obtained with itself and its 1000 randomized variants, when a particular classifier is used for classification.

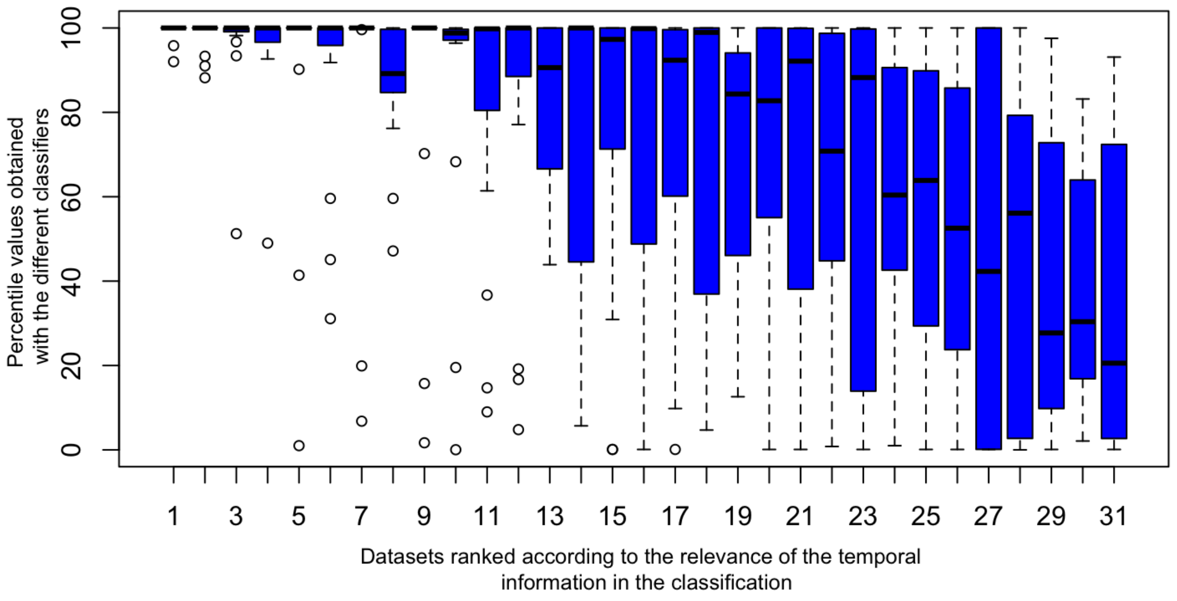


Figure 38: The distribution of the percentile values obtained with the different classifiers in the case of each of the TSC problems considered. The datasets are ordered in the X axis according to the relevance of the temporal order in the classification. That is, those classifiers with the highest associated mean percentile values are on the left, and those with the lowest mean percentile values on the right.

According to Figure 38, those datasets that recorded the highest mean percentile values (when averaged over the results of the different specific TS classifiers) have narrower distributions than the ones that had the lowest mean percentile values. That is, in the datasets where the recorded mean percentile value is high, the agreement between the classifiers on the value of the percentile of the accuracy of the original dataset in relation to the empirical accuracy distribution is higher. Besides, the lower the mean percentile value related to the TSC problem, the wider its distribution about the recorded percentile value gets. That is, there are greater discrepancies among the results obtained with the distinct classifiers.

In short, in the case where the temporal order seems to be discriminatory, all the classifiers seem to agree; while discrepancies appear in those cases where the relevance of the temporal order seems to be low for classification.

4.3 Comparison With the Experiment of Abanda *et al.* [62]

In this section, a comparison is performed between the empirical study about the relevance of the temporal order on the classification of the previous section and the results obtained in [62], considering the datasets that are in common in both experiments. Those common datasets have been divided in two tables: on the one hand, those datasets where 1-NN-DTW outperformed 1-NN-EUC, and on the other, the datasets where 1-NN-EUC performed better than the 1-NN-DTW. For those datasets, it has been specified the mean percentile value obtained in the previous section, and the particular percentile value that it is obtained with the 1-NN-DTW classifier.

For those cases where 1-NN-DTW outperformed 1-NN-EUC, their corresponding mean and particular percentile values are shown in Table 12. Most of the datasets present high percentile values; nonetheless, there are datasets such as the `Coffee` dataset, where the value of the percentile is low. Moreover, the percentile values obtained with the 1-NN-DTW classifier are very high for all the datasets of this table except for the `Coffee` dataset.

Dataset	Mean Percentile	Percentile 1-NN-DTW
Trace	98.28	100.0
Lightning2	89.22	99.90
TwoLeadECG	86.73	100.0
CBF	82.35	100.0
FaceFour	78.20	100.0
Coffee	57.87	2.90

Table 12: The percentiles of the datasets where 1-NN-DTW outperformed 1-NN-EUC [62].

Moreover, the respective percentile values of those datasets where 1-NN-EUC outperformed 1-NN-DTW can be found in table 13. For these datasets, in most of the cases, neither the mean values of the percentiles, nor the particular percentile values obtained with the 1-NN-DTW classifier are low.

Dataset	Mean Percentile	Percentile 1-NN-DTW
GunPoint	95.63	100.0
SonyAI1	87.39	76.20
SonyAI2	82.50	99.50
ECG200	70.40	61.80
ECGFiveDays	62.52	100.0

Table 13: The percentiles of the datasets where 1-NN-EUC outperformed 1-NN-DTW [62].

Therefore, it does not hold that in those datasets where 1-NN-DTW outperformed 1-NN-EUC, the highest percentile values are obtained, neither with the 1-NN-DTW nor when averaging the percentile values over the 16 specific classifiers considered. Additionally, in the cases where 1-NN-EUC outperformed 1-NN-DTW, the recorded mean percentile values and the percentiles regarding the 1-NN-DTW classifier are not the lowest.

Besides, the results from the previous section (see Section 3.6), demonstrated that the 1-NN-DTW algorithm was one of the least sensitive specific classifiers. Thus it might not be the most adequate algorithm to study the relevance of the temporal order of the instances by making comparisons with non-specific classifiers. Instead, a classifier with high sensitivity to the alteration of the temporal order of the observations should be considered.

A similar study to that in [62] has been performed, but in this case, considering the three classifiers with the highest sensitivity to the temporal information (Proximity Stump, Proximity Tree and MrSEQL), as well as 1-NN-EUC and three conventional non-specific classifiers (SVM, NB and RF). Furthermore, the 5 datasets with the highest percentile value and the 5 with lowest have been considered. Each of them has been classified by every one of those 7 classifiers. The accuracy values obtained in each of the cases are recorded in Table 14.

Note that when computing the euclidean distance between two time series, it is invariant to the temporal order of the observations. Thus the 1-NN-EUC classifier is considered non-specific.

	Specific TS classifiers			Non-specific TS classifiers			
Dataset	Prox. Stump	Prox. Tree	MrSEQL	1-NN-EUC	SVM	NB	RF
Plane	**1.00**	0.99	**1.00**	0.96	0.96	0.95	0.99
Trace	0.78	0.53	**1.00**	0.76	0.52	0.79	0.85
ShapeletSim	0.57	0.58	**1.00**	0.54	0.53	0.49	0.5
GunPoint	0.48	0.86	**0.99**	0.91	0.77	0.79	0.93
ToeSeg2	0.19	0.75	**0.91**	0.81	0.74	0.65	0.75
Herring	**0.59**	0.50	**0.59**	0.52	**0.59**	0.58	0.58
Chinatown	0.86	0.91	0.95	0.94	0.89	0.96	**0.98**
ItalyPD	0.51	0.93	0.91	0.96	0.96	0.90	**0.97**
BME	0.58	0.49	0.90	0.83	0.64	0.97	**0.98**
Meat	0.67	0.88	0.92	0.93	0.88	**0.95**	0.93

Table 14: The accuracy values obtained with the specific and non-specific classifiers, for the databases with the highest (first 5 databases) and lowest (last 5 databases) percentiles. The values in bold refer to the highest accuracy obtained for the dataset with the specific and non-specific classifiers considered in this table.

According to Table 14, in the TSC problems where the recorded mean percentile is high, the specific TS classifiers outperform those that are not sensitive to the temporal correlation of the observations. Conversely, in those datasets where the resulting mean percentile value is low, the non-specific classifiers obtain better or at least competitive performaces in comaprison to the specific TS classifiers. Therefore, as expected in the initial hypothesis, in TSC problems where the relevance of the temporal information is high for the classification, the specific time series classifiers outperform the rest, as they are able to learn the information given by the temporal ordering. On the other hand, in the cases where the temporal order of the instances is not discriminatory for the classification, conventional non-specific classifiers record better or competitive performances than the specific time series classifiers. Thus, in the case of TSC problems where the temporal order of the observations is not relevant, conventional non-specific classifiers could also be considered, as there is no need to read the information given by the temporal correlation of the observations.

Henceforth, when considering high sensitivity classifiers, the results seem to have higher coherence. That is, in the case of the TSC problem where the percentile is high, it is supposed that the temporal order of the observations is of high relevance for the classification, thus specific classifiers outperform the ones that are not sensitive to the temporal information. Moreover, in the opposite cases, where the temporal order seems to be of little relevance for the classification, as the recorded percentiles are low, specific TS classifiers are outperformed by conventional ML classifiers.

Actually, the MrSEQL classifier has presented the most informative results regarding the specific TS classifiers. When considering the cases with the highest percentile values, it outperforms the conventional ML classifiers. Furthermore, when it comes to the datasets that recorded low percentile values, it is outperformed by at least 2 of the conventional ML classifiers (SVM, NB or RF), except for the case of one dataset.

Figure 39 shows the best accuracy obtained among the conventional ML classifiers (SVM, NB and RF) in comparison with the accuracy obtained with the MrSEQL specific TS classifier, with the datasets positioned in the X axis according to their corresponding mean percentile value (averaged over the results of the 16 specific TS classifiers). In classifiers with high mean values of the percentile, MrSEQL outperforms the conventional ML classifiers; but its performance is upper-bounded by the best performance of the non-specific classifiers when applied to the datasets with the lowest average percentile values.

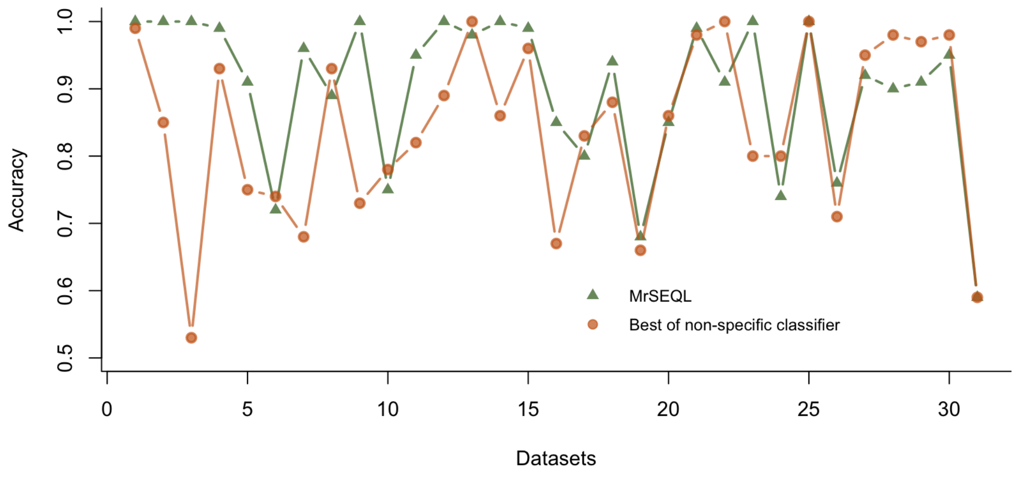


Figure 39: The acuraccy of the datasets using the MrSEQL classifier and the best accuracy obtained with the non-specific classifiers.

5 Correlations Between Different Characteristics

As the main motivation to develop specific TS classifiers was to design algorithms that could outperform the conventional ML algorithms by considering the temporal correlation of the instances on the classification, one would expect that the higher the sensitivity of the classifier to the temporal information, the better its performance would be. Moreover, it could be also expected that the most sensitive classifiers are the ones obtaining the highest mean percentiles, whenever they are applied to the original datasets and their randomized variants, particularly in those problems where the temporal order is of high relevance for the classification.

This section studies the correlations between the different characteristics of the classifiers and the datasets, based on the results attained in the empirical studies of the previous sections.

5.1 Sensitivity vs. Performance of the Classifiers

As mentioned in the introductory section, in some TSC problems conventional ML classifier were found to be insufficient, and specific classifiers that take into account the temporal correlation of the instances were proposed. Moreover, in the previous section, the empirical results showed that in the cases where the temporal order is relevant for classification (that is, when the TSC problems recorded high mean percentile values), the specific TS classifiers outperformed the conventional non-specific classifiers. Nonetheless, it is still unknown how does the performance vary among the considered specific TS classifiers; in other words, whether the classifiers with the highest sensitivity to the temporal information are the ones performing the best, that is, obtaining the highest accuracy values.

If the performance (accuracy values) of the classifiers are averaged over the 31 datasets, the mean accuracy values are those shown in Figure 40. There is no apparent correlation between the sensitivity of a classifier and its performance; that is, being more sensitive does not directly imply a better average performance. In fact, the classifiers that have performed best over the considered datasets are positioned in intermediate positions of the temporal sensitivity ranking: WEASEL + MUSE (10th position), WEASEL (9th position), MrSEQL (3rd position) and TDE (6th position). Therefore, although the main purpose of developing specific TS classifiers was to define classifiers that were able to use the information given by the temporal correlation of the instances to improve the accuracy, it is not a direct consequence that those classifiers that are more sensitive to the temporal order perform the best.

The best average performance is given by the WEASEL + MUSE algorithm, very closely followed by the WEASEL algorithm. The other two algorithms that recorded the best performances are MrSEQL and TDE. These two last approaches use multiple window lengths to capture different temporal substructures combined, but unless other algorithms using multiple window lengths, they consider the order of the windows, instead of considering each window as an independent feature. That is, the information about the location of the substructures is also relevant for classification.

Furthermore, among the dictionary-based classifiers, all the classifiers that are ensembles of individual BOSS classifiers (BOSS ensemble, contractable BOSS and TDE), perform better than the former. That is, considering multiple window lengths, in order to extract the information of the different substructures of the series, improves the results of the algorithm that considers a single size of the sliding window. Moreover, the individual TDE outperforms the individual BOSS, showing that even if both consider a single combination of parameters, selecting those parameters by means of a Gaussian process improves the accuracy. Moreover, inside the TDE classifiers, the individual TDE classifier is outperformed by the TDE, showing again that considering different parameter combinations the overall performance of the classifier is improved. To sum up, in these dictionary-based algorithms, considering multiple window lengths to extract features that are representative of different temporal substructures, endows the classifier with a higher capability to capture

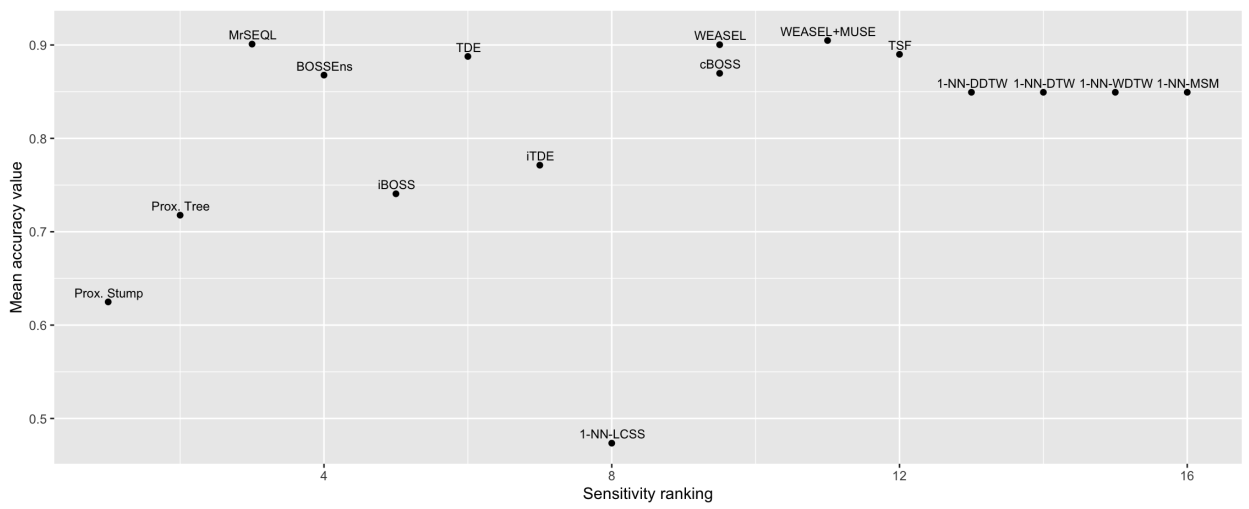


Figure 40: The mean accuracy values for the different classifiers, ordered on the X axis according to their position in the joint ranking.

the particular temporal information regarding the different TSC problems.

That is, in those classifiers based on the BOSS algorithm, the sliding window technique is used to extract temporal information from subsequences. In the cases where a single fixed-size sliding window is used, only the temporal substructures that match with the window size can be studied. Nonetheless, when multiple window lengths are considered, a wide variety of different sized substructures are analysed, and thus more temporal information is extracted for classification. Consequently, the latter perform best than those using a single window length.

In the case of the algorithms with decision tree architectures, the TSF performs better in average that the proximity tree and the proximity stump. That is, considering multiple tree structures, and having an output that is a combination of their individual predictions, improves the average performance. Moreover, the proximity tree outperforms the proximity stump, which is a proximity tree of level 1.

The effect of considering a forest instead of a single decision tree, is similar to that when multiple window lengths are used instead of a single window. Each of the trees of the forest may detect different temporal structures, thus when combining their predictions, the final results is be based on all the temporal information caught by the individual trees. Thus the forest has a greater ability to read the intrinsic temporal information of the instances as it considers multiple temporal substructures at the same time.

Moreover, Figure 40 shows that 1-NN-DTW, 1-NN-WDTW, 1-NN-DDTW and 1-NN-MSM have identical performances, outperforming the 1-NN-LCSS classifier. The latter has recorded the worst average performance, while the rest of the algorithms that perform a similarity matrix in which the 1-NN classifier is based, are found in the intermediate positions regarding their performance.

5.2 Sensitivity vs. Mean Percentile of the Classifiers

In a previous section, the sensitivity of a classifier to the temporal order of the instances has been estimated by observing how the performance of the classifier vary when the original temporal information is lost, when the timestamps of the instances are re-ordered randomly. The ones that have shown the highest sensitivies to the alteration of the temporal order of the instances could be expected to better catch the temporal information of the TSC problems (particularly in those cases where such information is discriminatory for the classification), as when such order has been lost by creating alternative temporal orderings, their performance has varied considerably.

Nonetheless, by averaging the percentile values obtained with each of the classifiers over the 31 datasets (see Figure 41), actually, the two most sensitive classifiers (Proximity Stump and Proximity Tree) are two of the classifiers with the lowest mean percentiles. That means, that even if their performance varies when the temporal order of the observations changes randomly, and in conclusion are sensitive to such order, they fail to read the correct temporal information of particular TSC problems. That is, their high sensitivity makes them be too general, losing their ability to detect the particular temporal correlation of a given TSC problem.

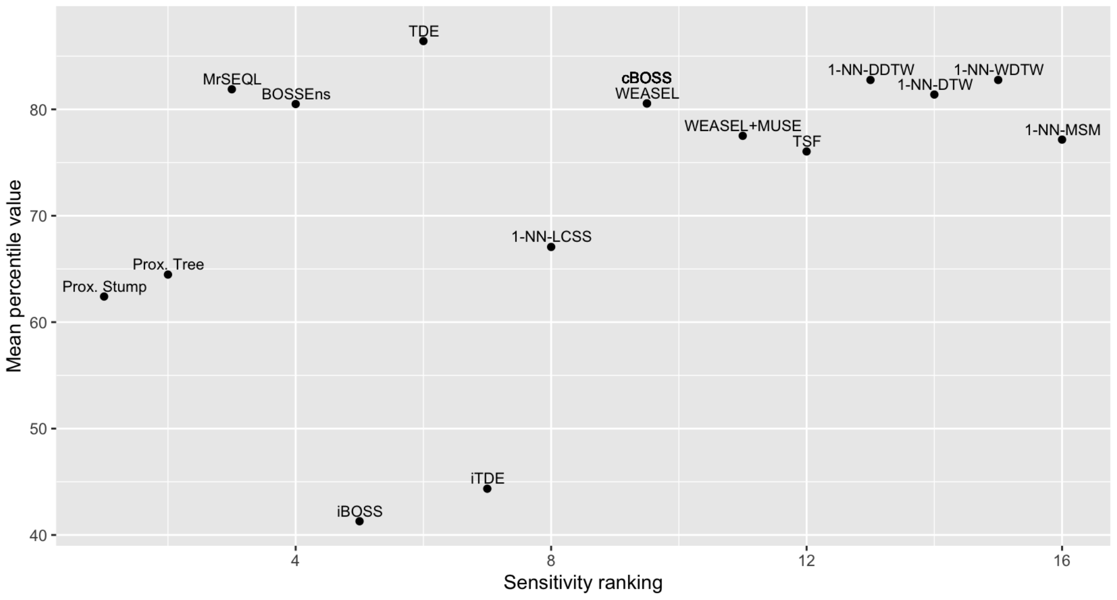


Figure 41: The mean percentile values of the classifiers, when averaged over the 31 datasets considered in this work. That is, the percentile of the accuracy obtained with the original dataset in comparison to the empirical accuracy distribution resulting from the classification of its randomized variants. Each classifier obtains a percentile value for each TSC problem considered, and this figure shows the average of those values for each of the specific TS classifier studied in this work.

Therefore, although it is important for the classifiers to be sensitive to the temporal order of the instances, particularly in those cases where the temporal information is discriminatory for the classification, they need to be able to correctly read the information given by the temporal correlation of particular TSC problems.

For example, even if the TSF is not a forest of proximity trees, but rather a forest of decision trees based on interval attributes, it gives an idea of what happens when different decision trees are combined creating more complex structures: although the overall sensitivity of the classifier decreases, it becomes better in reading the correct temporal information of the given dataset.

Moreover, when the parameters of a BOSS classifier are chosen by a Gaussian process (individual TDE), the ability of the classifier to better detect the temporal information for classification improves (see Figure 41), as well as when ensembles of individual BOSS algorithms are considered, defining different parameter settings. In the cases where multiple sliding windows are used to extract features, multiple different length temporal substructures are considered in the attributes, expanding the diversity of the temporal information considered for classification. According to this empirical study, they have a higher capacity to capture the particular temporal information of different TSC problems.

Furthermore, the MrSEQL algorithm, which consider both time domain and frequency domain symbolic representations, has also a great ability to detect the temporal information.

Thus according to the results obtained in this section, creating ensembles of base algorithms or forests of single decision trees, upgrades the ability of the classifier to detect the specific temporal information regarding a particular TSC problem, and use it on the classification of the instances. That aptitude is also improved when symbolic representations of different domains are combined. Nonetheless, in some of those cases, their capacity for detecting the correct temporal information is similar to the one that simpler algorithms such as 1-NN-DTW, 1-NN-WDTW, 1-NN-DDTW and 1-NN-MSM have (see Figure 41). This result explains why these benchmark classifiers have happened to be so hard to beat.

5.3 Mean Percentile vs. Performance of the Classifiers

The main conclusion of the previous section has been that those classifiers that are more sensitive against the temporal alteration of the instances are not necessarily the best in detecting the correct temporal information of a given TSC problem. Nonetheless, it is true that those classifiers that have the greatest ability to correctly detect the particular temporal information of TSC problems (that is, those that have the highest average percentile value) are the ones performing the best in average (see Figure 42).

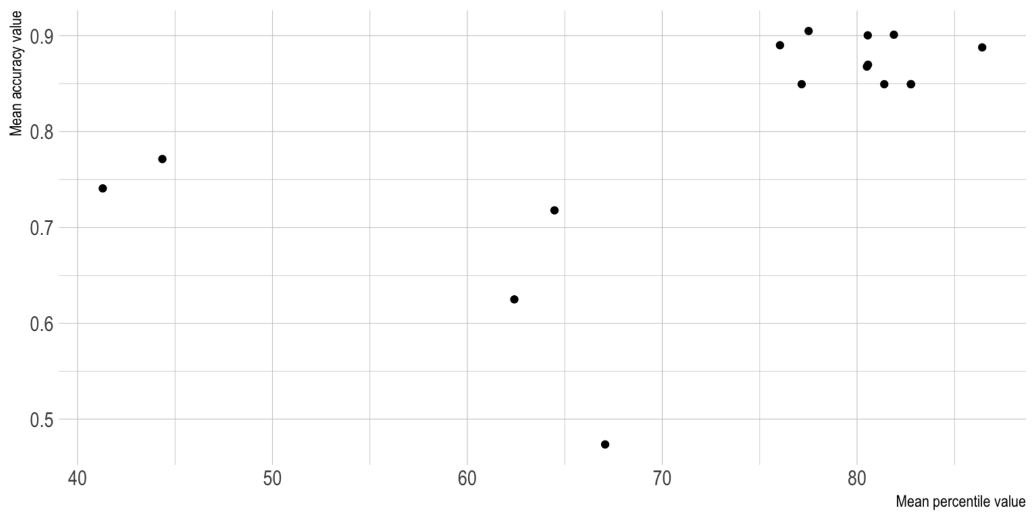


Figure 42: The relation between the mean accuracy and percentile values (averaged over the 31 datasets) recorded in the studied specific TS classifiers.

In conclusion, when more accurate classifiers are designed, their performance is improved not directly because they are more sensitive, but because they have a greater ability to detect and learn the particular temporal information given by a specific dataset for its classification.

Therefore, according to this study, the ability of a classifier to correctly read the particular temporal information of a given TSC problem is related to the temporal information it captures: the higher the diversity of the temporal structures it considers, the greater skills it develops to read the correct temporal information of specific TSC problems.

Among the dictionary based classifiers, considering multiple window lengths in the window sliding method, discretizing the TS by means of symbolic representations of different domains (e.g. time and frequency domain) or capturing the temporal location of the patterns, expands the temporal information captured by the classifier as well as the diversity of such information. Consequently, the classifiers can better read the correct temporal information of each particular TSC problem, improving their performance.

Similarly, when single decision trees are combined forming forests, the ability for the classifier to read the correct temporal information is increased. Each of the decision trees may capture a distinct temporal information. Thus when their individual outcomes are combined, the final prediction is based on a wider range of different temporal structures, making the classifier more accurate.

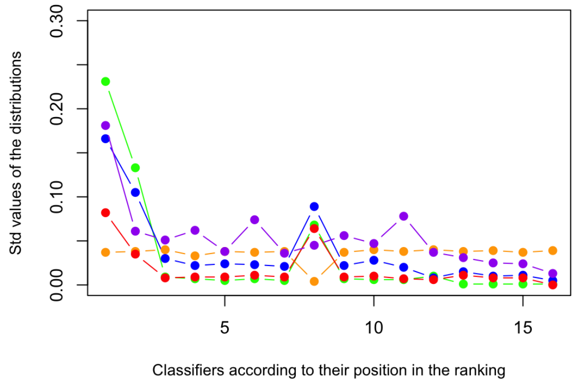
Coming back to the results of the experiment of Abanda *et al.* [62], 1-NN-DTW is not among those specific TS classifiers that best read the temporal information. Therefore, it would be interesting to replicate their analysis using a classifier that has a greater capacity to better read the particular temporal information of TSC problems. That is, using a specific TS classifier with a higher mean percentile value. For example, as it has been proposed in a previous section, using the MrSEQL algorithm would be a good choice as it is among those classifiers that catches best the temporal information of particular TSC problems, better than the 1-NN-DTW algorithm.

5.4 The Effect of the Relevance of the Temporal Information on the Accuracy Distribution

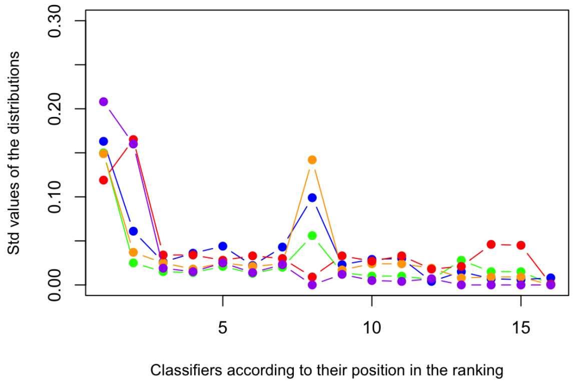
The sensitivity of a specific TS classifier to the temporal information is independent of the TSC problem, as it is a characteristic of the classifier itself. Therefore, ideally those classifiers that are significantly more sensitive, that is those that recorded statistically significant wider distributions, should maintain their lead position whenever their are applied to different datasets and their corresponding randomized variants.

To verify whether that statement is true, we have considered the 5 datasets with the highest average percentile value and the 5 datasets with the lowest. For those TSC problems, we have represented graphically the standard deviations of the their empirical accuracy distributions obtained with the different specific TS classifiers considered in this study (see Figure 43). Figure 43(a) shows the values of the standard deviations of the empirical accuracy distributions obtained from the different specific classifiers for the datasets in which the relevance of the temporal information has been high (highest mean percentile values), and Figure 43(b) for those datasets in which the relevance of the temporal information was low (low mean percentile value). In both graphics, each of the curves represents a different dataset and the classifiers are ordered in the X axis according to the sensitivity ranking. Figure 43 shows that the observed general trends are equivalent in both groups: the classifiers with the highest sensitivity record the widest empirical distributions, and they get narrower for classifiers with lower sensitivity. Therefore, independently of the relevance of the temporal information for a given TSC problem, the most sensitive classifiers obtain empirical accuracy distributions with significantly higher standard deviations.

Moreover, Figure 44 shows that the change on the performance of a classifier may vary from one dataset to another, as the same classifier has obtained distributions with different standard deviation values when applying to different TSC problems. Some classifiers have recorded greater variations than others. Nonetheless, those classifiers with higher average standard deviation values vary in ranges limited by higher values.



(a)



(b)

Figure 43: The standard deviation values recorded for the distributions related to each of the classifiers for those datasets (a) with the highest mean percentile values and (b) those with the lowest percentile values. The classifiers are ranked in the X axis according to their sensitivity to the temporal order of the instances.

That is, even if the variation of their performance (the standard deviation of the empirical accuracy distribution) changes when they are applied to different TSC problems, the specific TS classifiers that have been shown to be the most sensitive, always record the empirical distributions with the highest standard deviations.

Therefore, although the width of the empirical accuracy distribution obtained with a classifier varies from one dataset to the other, in general, the most sensitive specific classifiers have recorded the widest distributions and the least sensitive classifiers the narrowest.

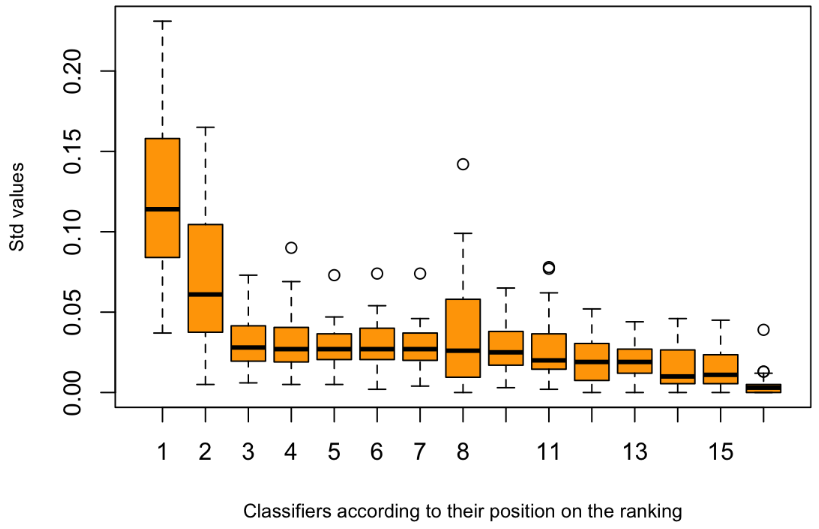


Figure 44: The values of the standard deviation values of the empirical accuracy distributions obtained with the different classifiers when they are applied to different TSC problems and their randomized variants, in which the temporal order is altered. Each point on the X axis refers to a specific classifier, ordered according to their position in the sensitivity ranking. On the other hand, the Y axis holds the standard deviation values of the empirical accuracy distributions obtained when the classifier is applied to each of the considered TSC problems and its randomized variants.

5.5 Accuracy vs. Relevance of Temporal Information

In Section 4 we have proven empirically that in those TSC problems where the relevance of the temporal information for the classification is low, the conventional ML classifiers outperform or are at least competitive with the specific TS classifiers. With that, one could expect that the higher the relevance of the temporal information on the classification, the better the performance of the specific TS classifiers would be. Nonetheless, Figure 45 shows there is no clear correlation between such two attributes. The X axis represents the percentile values related to the different datasets when the accuracy of the original dataset in compared to the accuracy distribution obtained with its 1000 randomly altered variants. Actually, it represents the mean percentile values averaged over the considered 16 classifiers. Moreover, the Y axis accounts for the mean accuracy values of each of the datasets, calculated by averaging the performance of the different classifiers on that specific TSC problem.

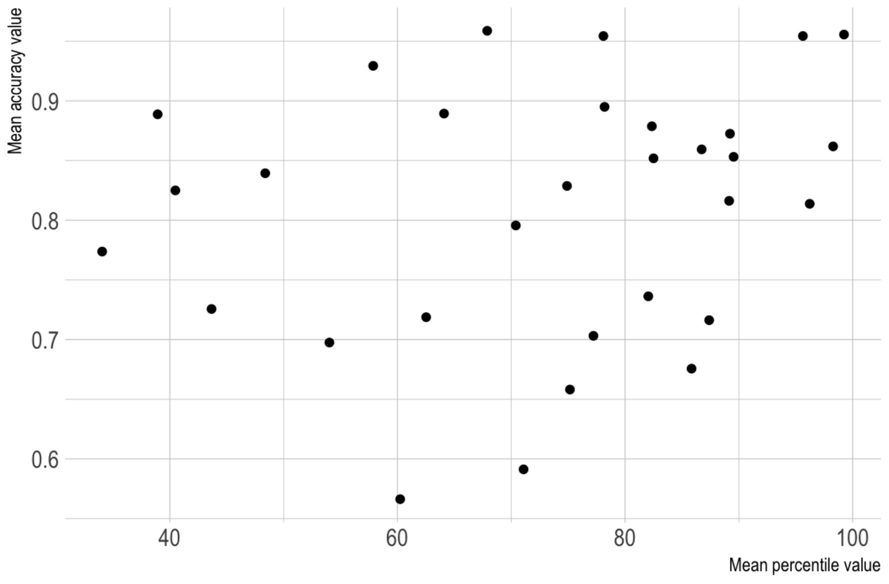


Figure 45: The mean percentile values and mean accuracy values of the different TSC problems averaged over the 16 specific classifiers.

This result is not surprising as the accuracy of a TSC problem depends on numerous factors, such as, the class balance among the instances of the dataset, the number of instances used for training the classifier, etc. Nonetheless, it is true, that if the temporal information of a given database is discriminatory for classification, those classifiers with the greatest ability to capture the intrinsic temporal information of a given TSC problem will perform the best (see Figure 46). This figure shows the performances of the different classifiers, ordered in the X axis according to their ability to capture temporal information. The ones that best capture the particular temporal information of different TSC problems (that is, the ones that recorded the highest mean percentile values in Figure 41) are located on the left. Moreover, the TSC problems for which the accuracy is accounted in the graphic are the top 5 TSC problems where the relevance of the temporal information has been found to be the highest. This figure shows that there is a difference between the performances of the classifiers that better capture the temporal information to those that have less ability to do so.

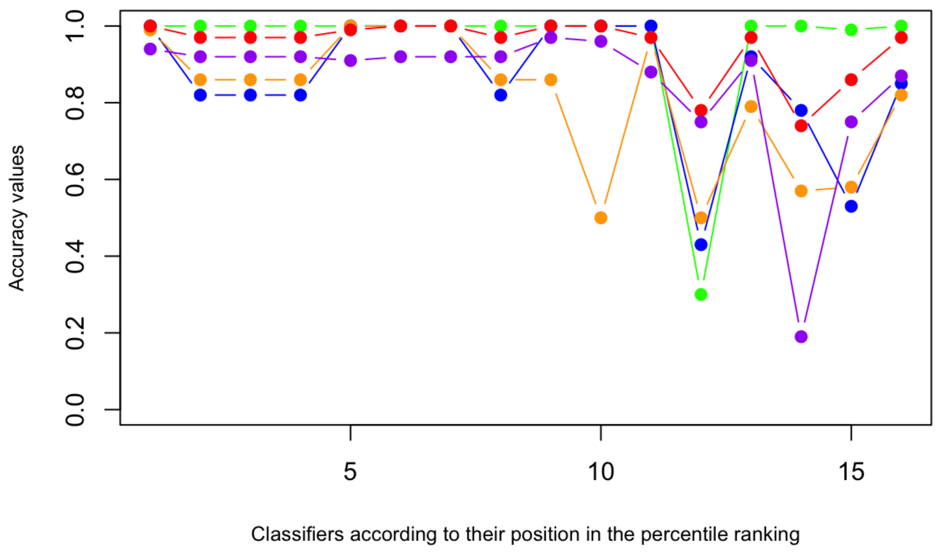


Figure 46: The mean accuracy values of the 5 TSC problems where the temporal information had the highest relevance for classification averaged over the 16 specific classifiers.

Moreover, if the same analysis is carried out in those datasets where the temporal information has been found to have low relevance for classification (see Figure 47), the difference between the performances of the algorithms, each of them having a different capacity to capture the temporal information, is smaller. Nonetheless, it is true that in general, those algorithms that have a greater ability to capture the temporal information of the problem perform the best: as little as it might be the intrinsic temporal information that is relevant for the classification, they are able to capture it better.

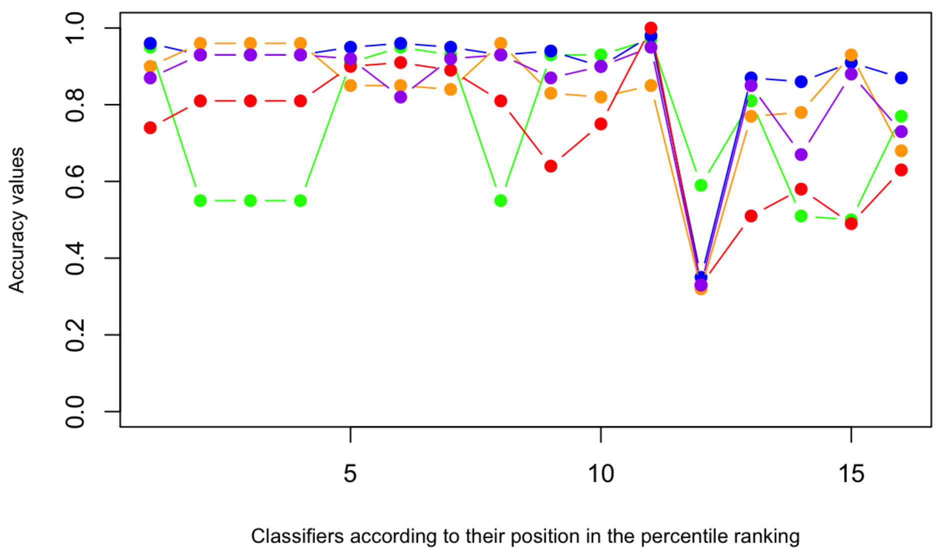


Figure 47: The mean accuracy values of the 5 TSC problems where the temporal information had the lowest relevance for classification averaged over the 16 specific classifiers.

6 Conclusions

The first part of this work has been an overview of some of the existing specific TS classifiers, which have been grouped according to their classification procedure. Some approaches use a time series distance measure to calculate the (dis)similarity matrix, and then use a distance-based classifier. Other approaches extract discriminative features, and then map the dataset to the new feature space in which the classification is based by means of conventional distance-based or feature-based classifiers. Depending on the algorithm, those features are created by extracting words from real-valued TS using a symbolic representation, by calculating the distance of a TS to discriminative patterns, performing convolutions with kernels or by means of summary statistics of intervals. In addition, ensembles of base classifiers have also been proposed in literature, which combine the individual outcomes of the base classifiers for a final prediction. Furthermore, TS distance measures and features that capture structural information can be used as a splitting criteria in decision tree structures. Finally, several recently proposed deep learning structures have been presented. Moreover, it has been outlined how each classifier is affected when the timestamps of the observations are re-ordered randomly.

In those algorithms where conventional classifiers are used for classification (after computing the similarity matrix or once the features have been extracted), the temporal information is intrinsically transmitted by the distance measures used and the features that are extracted from the real valued TS. The final classifier will change its performance depending on the given distance values and the set of attributes and their values; not because it is sensitive to the temporal information, but because depending on the ordering of the timestamps the temporal information defined by the distance values and the set of attributes and their values is different. Therefore when the temporal order of the instances is randomly altered the distance matrix and feature based descriptions will not longer be representative of real temporal information, thus the final classifier will have erroneous information to base the classification on, affecting its performance.

In deep learning architectures, it is not easy task to identify how will the alteration of the temporal order of the instances affect its performance as the internal behaviour if hidden layers is not easily interpretable, because of the non-linear transformations that are performed in the input data. Nonetheless, the effect of the random re-ordering of the timestamps has been estimated in the case of the presented approaches. All the presented architectures except MLP were found to be sensitive to the intrinsic temporal information of the observations. The architectures that contain convolutional layers happen to be sensitive to the temporal information, as the result of those convolutional layers depend on temporal substructures of the observations. Furthermore, considering multi-scale convolutional layers allows the network to capture and learn more complex temporal structures. Additionally, the GAP layers and attention layers that are present in some of the presented CNN architectures, highlight the most discriminant regions and subsequences of the TS. Moreover, in recurrent architectures, the learning processes of networks strongly depend on the initial values of the observations, thus their performance will be affected by the random alteration of the temporal order of the instances.

Besides, this work has analysed the sensitivity of different specific TS classifiers to the random alteration of the temporal order of the instances, based on empirical accuracy distributions. For each TSC problem considered, the performance variations of the different classifiers have been estimated subject to the standard deviation values of the empirical accuracy distributions obtained when the original TSC dataset and 1000 of its randomized variants were classified. The process has been repeated for 31 different TSC problems developing a joint ranking, from which a general ranking of sensitivity has been concluded.

This study has concluded that when considering higher-level structures instead of only restricting to local similarities, the sensitivity of a classifier increases. Regarding dictionary based algorithms, discretizing the real-valued time series by symbolic representations in both time domain and frequency domain, and considering multiple resolutions for each, increases the sensitivity of the classifier to the temporal order of the observations. On the other hand, creating ensembles of BOSS algorithms in the conventional way, increases the sensitivity of the classifier. Nonetheless, when the ensemble is created based on a random process, the sensitivity is decreased. Furthermore, when the parameters of the BOSS algorithm are chosen by a Gaussian process, the sensitivity of the algorithm is also decreased. In the case where a single multiple resolution symbolic representation is used, when a feature selection is perform to consider only the most discriminative substructures for classification, the sensitivity of the algorithm decreases. Besides, among decision tree structures, considering deeper structures decreases the sensitivity of the algorithm, as well as combining multiple decision trees forming a forest.

Furthermore, we have found out that the temporal information is not always discriminatory for classification. That is, there are TSC problems where the temporal order of the instances has low relevance for classification. To study such issue, we have compared the performance of the original dataset to those obtained when the timestamps of the observations were randomly shuffled. If the performance obtained with the original dataset is not good enough in comparison to its randomized variants, the temporal order of the instances has low relevance for classification. On the other hand, if losing the original temporal information makes the performances of the specific TS classifiers deteriorate, then the temporal information is considered to be discriminant for classification. In the cases where the temporal information was not found to be relevant, conventional non-specific classifier outperformed or at least were competitive with specific TS classifiers. Therefore, conventional ML classifiers could be used to classify TSC problems in which the temporal correlation of the attributes has low relevance for classification.

Moreover, the results of this empirical study have shown that, while specific TS classifiers were needed to classify at least those TSC problems where the temporal information was discriminant for classification, being more sensitive to the temporal order of the instances does not directly imply a better performance. That is, although the classifiers need to be sensitive to capture the temporal information in the classification process, their ability to correctly read the particular temporal information of a given TSC problem is what ensures a better performance.

The ability of a classifier to correctly read the particular temporal information of a given TSC problem is related to the temporal information it captures: the higher the diversity of the temporal structures it considers, the better its potential to read the temporal information of a specific TSC problem. Therefore, among the dictionary based classifiers, considering multiple window lengths in the window sliding method, discretizing the TS by means of symbolic representations of different domains (e.g. time and frequency domain) or capturing the temporal location of the patterns, expands the temporal information captured by the classifier as well as the diversity of such information, which endows the classifier with a higher ability to read the correct temporal information from each particular TSC problem, improving its performance.

Equivalently, when single decision trees are combined forming forests, the ability for the classifier to read the correct temporal information is increased. Each of the decision trees may capture a distinct temporal information. Therefore, when their individual outcomes are combined, the final prediction is based on a more diverse set of temporal structures, which makes the classifier become more accurate.

Besides, the average accuracy of TSC problems have been demonstrated to be independent of the relevance of the temporal information for classification. Nonetheless, in those TSC problems where the temporal information has higher relevance for classification, the differences in performances between the specific classifiers that better captured the temporal information and those that do it worse, is greater. Furthermore, this work showed, that even when the temporal information that is relevant for classification is little, and non-specific classifiers are competitive with the specific ones, among the latter, those that have the greatest ability to capture the temporal information still perform better. That is, as little as it might be the temporal information that is relevant for classification, the specific TS classifiers that better read such information, still capture it better and record better performances. Nonetheless, in this case, the differences in performances among the specific TS classifiers are smaller.

Moreover, the results of this study have shown that although the width of the empirical accuracy distribution obtained with a classifier varies from one TSC problem to other, in general, the most sensitive classifiers have recorded the widest distributions in all the TSC problems.

The number of datasets and classifiers considered in this empirical study, has been limited by the temporal cost of the experiment. For each combination of dataset and classifier 1001 classification processes had to be carried out. Therefore, only those needing a reasonable time to perform a single classification were feasible to use for this work. Moreover, only those classifiers which had their implementations in python were used. Therefore, it would be interesting to widen the set of considered specific classifiers, when new algorithms requiring reasonable computational times to perform a single classification are added to python. For example, adding deep learning architectures to this study would be one of the foremost objectives of future works.

In further work, it would be also riveting to mature this analysis by defining an statistical analysis to estimate the relevance of the temporal information of a given TSC problem by means of a mathematical formalization. Nonetheless, for doing so, the probability distribution of the temporal information would be needed to define.

References

[1] P. J. Brockwell, P. J. Brockwell, R. A. Davis, and R. A. Davis, *Introduction to time series and forecasting*. Springer, 2016.

[2] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining and knowledge discovery*, vol. 7, no. 4, pp. 349–371, 2003.

[3] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[4] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multidimensional time-series," *The VLDB Journal*, vol. 15, no. 1, pp. 1–20, 2006.

[5] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clusteringâa decade review," *Information Systems*, vol. 53, pp. 16–38, 2015.

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

[7] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, pp. 1–49, 2020.

[8] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[9] P. Esling and C. Agon, "Time-series data mining," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–34, 2012.

[10] U. M. Carrascal, "Contributions to time series data mining departing from the problem of road travel time modeling," Ph.D. dissertation, Universidad del País Vasco-Euskal Herriko Unibertsitatea, 2015.

[11] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.

[12] A. Jalalian and S. K. Chalup, "Gdtw-p-svms: Variable-length time series analysis using support vector machines," *Neurocomputing*, vol. 99, pp. 270–282, 2013.

[13] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[14] T. W. Liao, "Clustering of time series dataâa survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[15] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognition*, vol. 44, no. 9, pp. 2231–2240, 2011.

[16] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proceedings of the 2001 SIAM international conference on data mining*, SIAM, 2001, pp. 1–11.

[17] T. G$\tilde{A}^3$*recki* and M. Åuczak, "Using derivatives in time series classification," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 310–331, 2013.

[18] A. Stefan, V. Athitsos, and G. Das, "The move-split-merge metric for time series," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1425–1438, 2012.

[19] M. A. R. Khan and M. Zakarya, "Longest common subsequence based algorithm for measuring similarity between time series: A new approach," *World Applied Sciences Journal*, vol. 24, no. 9, pp. 1192–1198, 2013.

[20] G. Soleimani and M. Abessi, "Dlcss: A new similarity measure for time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 92, p. 103 664, 2020.

[21] J. Lines and A. Bagnall, “Time series classification with ensembles of elastic distance measures,” *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565–592, 2015.

[22] J. Lin, R. Khade, and Y. Li, “Rotation-invariant similarity in time series using bag-of-patterns representation,” *Journal of Intelligent Information Systems*, vol. 39, no. 2, pp. 287–315, 2012.

[23] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing sax: A novel symbolic representation of time series,” *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[24] P. Geurts, “Pattern extraction for time series classification,” in *European conference on principles of data mining and knowledge discovery*, Springer, 2001, pp. 115–127.

[25] P. Senin and S. Malinchik, “Sax-vsm: Interpretable time series classification using sax and vector space model,” in *2013 IEEE 13th international conference on data mining*, IEEE, 2013, pp. 1175–1180.

[26] P. SchÃ¤fer, “The boss is concerned with time series classification in the presence of noise,” *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.

[27] P. Schäfer and M. Högqvist, “Sfa: A symbolic fourier approximation and index for similarity search in high dimensional datasets,” in *Proceedings of the 15th international conference on extending database technology*, 2012, pp. 516–527.

[28] M. Middlehurst, W. Vickers, and A. Bagnall, “Scalable dictionary classifiers for time series classification,” in *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, 2019, pp. 11–19.

[29] M. Middlehurst, J. Large, G. Cawley, and A. Bagnall, “The temporal dictionary ensemble (tde) classifier for time series classification,” *arXiv preprint arXiv:2105.03841*, 2021.

[30] J. Large, A. Bagnall, S. Malinowski, and R. Tavenard, “On time series classification with dictionary-based classifiers,” *Intelligent Data Analysis*, vol. 23, no. 5, pp. 1073–1089, 2019.

[31] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, vol. 2, IEEE, 2006, pp. 2169–2178.

[32] T. L. Nguyen, S. Gsponer, I. Ilie, M. OâReilly, and G. Ifrim, “Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations,” *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 1183–1222, 2019.

[33] T. L. Nguyen, S. Gsponer, and G. Ifrim, “Time series classification by sequence learning in all-subsequence space,” in *2017 IEEE 33rd international conference on data engineering (ICDE)*, IEEE, 2017, pp. 947–958.

[34] P. Schäfer and U. Leser, “Fast and accurate time series classification with weasel,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 637–646.

[35] D. A. Freedman, *Statistical models: theory and practice*. cambridge university press, 2009.

[36] M. F. Zibran, “Chi-squared test of independence,” *Department of Computer Science, University of Calgary, Alberta, Canada*, pp. 1–7, 2007.

[37] P. SchÃ¤fer and U. Leser, “Multivariate time series classification with weasel muse,” *arXiv preprint arXiv:1711.11343*, 2017.

[38] [Online]. Available: `http://timeseriesclassification.com/dataset.php`.

[39] L. Ye and E. Keogh, “Time series shapelets: A novel technique that allows accurate, interpretable and fast classification,” *Data mining and knowledge discovery*, vol. 22, no. 1, pp. 149–182, 2011.

[40] T. Rakthanmanon and E. Keogh, “Fast shapelets: A scalable algorithm for discovering time series shapelets,” in *proceedings of the 2013 SIAM International Conference on Data Mining*, SIAM, 2013, pp. 668–676.

[41] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, “Classification of time series by shapelet transformation,” *Data mining and knowledge discovery*, vol. 28, no. 4, pp. 851–881, 2014.

[42] A. Bostrom and A. Bagnall, “Binary shapelet transform for multiclass time series classification,” in *International conference on big data analytics and knowledge discovery*, Springer, 2015, pp. 257–269.

[43] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, “Learning time-series shapelets,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 392–401.

[44] A. Dempster, F. Petitjean, and G. I. Webb, “Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels,” *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, 2020.

[45] J. J. Rodríguez and C. J. Alonso, “Support vector machines of interval-based features for time series classification,” in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Springer, 2004, pp. 244–257.

[46] H. Deng, G. Runger, E. Tuv, and M. Vladimir, “A time series forest for classification and feature extraction,” *Information Sciences*, vol. 239, pp. 142–153, 2013.

[47] B. Lucas, A. Shifaz, C. Pelletier, L. OâNeill, N. Zaidi, B. Goethals, F. Petitjean, and G. I. Webb, “Proximity forest: An effective and scalable distance-based classifier for time series,” *Data Mining and Knowledge Discovery*, vol. 33, no. 3, pp. 607–635, 2019.

[48] A. Bagnall, F. Király, M. Löning, M. Middlehurst, and G. Oastler, “A tale of two toolkits, report the first: Benchmarking time series classification algorithms for correctness and efficiency,” *arXiv preprint arXiv:1909.05738*, 2019.

[49] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, “Time-series classification with cote: The collective of transformation-based ensembles,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, 2015.

[50] J. Lines, S. Taylor, and A. Bagnall, “Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles,” *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 5, 2018.

[51] Z. Wang, W. Yan, and T. Oates, “Time series classification from scratch with deep neural networks: A strong baseline,” in *2017 International joint conference on neural networks (IJCNN)*, IEEE, 2017, pp. 1578–1585.

[52] J. Serrà, S. Pascual, and A. Karatzoglou, “Towards a universal neural network encoder for time series.,” in *CCIA*, 2018, pp. 120–129.

[53] Z. Cui, W. Chen, and Y. Chen, “Multi-scale convolutional neural networks for time series classification,” *arXiv preprint arXiv:1603.06995*, 2016.

[54] A. L. Guennec, S. Malinowski, and R. Tavenard, “Data augmentation for time series classification using convolutional neural networks,” in *ECML/PKDD workshop on advanced analytics and learning on temporal data*, 2016.

[55] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, “Convolutional neural networks for time series classification,” *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[56] P. Tanisaro and G. Heidemann, “Time series classification using time warping invariant echo state networks,” in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2016, pp. 831–836.

[57] A. E. Hoerl and R. W. Kennard, “Ridge regression: Applications to nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 69–82, 1970.

[58] S. Siegel, “Nonparametric statistics for the behavioral sciences.,” 1956.

[59] C. Bonferroni, “Teoria statistica delle classi e calcolo delle probabilita,” *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, vol. 8, pp. 3–62, 1936.

[60] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király, "Sktime: A unified interface for machine learning with time series," *arXiv preprint arXiv:1909.07872*, 2019.

[61] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[62] A. Abanda, U. Mori, and J. A. Lozano, "¿requiere la clasificación de series temporales métodos específicos?" In *XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2018) 23-26 de octubre de 2018 Granada, España*, Asociación Española para la Inteligencia Artificial (AEPIA), 2018, pp. 790–795.