UNIVERSIDAD DE GRANADA

TIME SERIES ANALYSIS IN BIG DATA ENVIRONMENTS

DOCTORAL DISSERTATION

presented to obtain the

DOCTOR OF PHILOSOPHY DEGREE

in the

INFORMATION AND COMMUNICATION TECHNOLOGY PROGRAM

by

Francisco Javier Baldán Lozano

PhD Advisor

José Manuel Benítez Sánchez

DEPARTMENT OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE

Granada, October 2021

Editor: Universidad de Granada. Tesis Doctorales
Autor: Francisco Javier Baldán Lozano
ISBN: 978-84-1117-189-2
URI: http://hdl.handle.net/10481/72082

This doctoral thesis has been supported by the Spanish National Research Project TIN2016-81113-R and by the Andalusian Regional Government Project P12-TIC-2958. Francisco Javier Baldán Lozano holds the FPI grant BES-2017-080137 from the Spanish Ministry of Economy and Competitiveness.

A mi familia.

Agradecimientos

El desarrollo de una tesis doctoral no es una tarea solitaria, todas las personas del entorno del doctorando aportan, incluso sin saberlo, una luz y apoyo incalculables para afrontar una tarea que puede resultar en ocasiones dura y frustrante. Por este motivo me gustaría dedicarles unas palabras a algunas personas especialmente importantes para mí.

En primer lugar, me gustaría dedicar esta tesis a mi familia, sin ellos nada de esto habría sido posible. A mi padre, Francisco, por enseñarme cómo tener la mejor actitud ante la vida y a hacer lo correcto en todo momento. A mi madre, María Henar, por su cariño, comprensión y apoyo incondicionales. A mi hermana, Henar, por esas risas, conversaciones y momentos que te llenan de vida. A mi pareja, Raquel, con quien he compartido más de la mitad de mi vida, que me ha acompañado durante todo el proceso. Has aguantado preocupaciones, agobios, ensayos, imprevistos, *deadlines*... y siempre me has brindado tu amor y apoyo cuando más los necesitaba, haciéndome feliz en los momentos más adversos. Gracias por todo, deseo poder ser capaz de devolverte todo lo que me has dado.

A Germán, Loly y Javier por ser mi segunda familia.

De mi etapa académica me gustaría agradecer a mi tutor y director de tesis José Manuel Benítez la ayuda prestada y el permitirme realizar esta tesis. A Gregorio, María Jesús y Álvar por los interesantes intercambios de ideas y colaboraciones realizadas. También me gustaría agradecer a todos los compañeros doctorandos y sufridores *SysOps* que me han acompañado durante estos años: Manu, Fran, Óscar, Eva, Sergio R., Jesús, Sergio G., José A., Alexis, Paco Luque, Roberto... A Dani, por el apoyo y guía ofrecidos tanto al inicio como al final de esta tesis, gracias por una estancia en Gante increíble. A Carlos, por esas conversaciones densas pero relajadas que hacían más ameno el día a día, amigo que se preocupa, sabe escuchar y aconseja con esmero en cada momento. Y finalmente a Diego, quien se ha convertido en un amigo muy especial con el que he compartido momentos inolvidables, risas, preocupaciones, frustraciones, consejos... y todo lo que aún está por venir. Muchas gracias por ser como eres.

Special thanks to Dr. Yvan Saeys and his DAMBI research group for the incredible treatment and support received. I would like to thank Daniel for the invaluable hospitality and attention received during my stay in Ghent.

Por último, me gustaría agradecer a mis amigos Ana, Alejandro y Marina por esas increíbles tardes lúdicas. A Cristóbal por esos paseos e interesantes conversaciones nocturnas. A David, Magda, Caroline y Luana por hacer de mi estancia en Gante una experiencia inolvidable.

Muchas gracias a todos/as. Formáis parte de esta tesis.

Table of Contents

List of Abbreviations

1NN+DTW	One-Nearest-Neighbor + Dynamic Time Warping
ALSTM-FCN	Attention LSTM-FCN
BOP	Bag of Patterns
c-RISE	Contract Random Interval Spectral Ensemble
catch22	CAnonical Time-series CHaracteristics
CIF	Canonical interval forest
DFTS	Distributed FastShapelet Transform
DTW	Dynamic Time Warping
DTW_A	Dynamic Time Adaptative Warping
DTW_D	Dynamic Time Dependent Warping
DTW_I	Dynamic Time Independent Warping
FEARS	Feature and Representation Selection
FS	FastShapelet
HIVE-COTE	Hierarchical Vote Collective of Transformation-based Ensembles
gRSF	Generalized Random Shapelets Forests
KDD	Knowledge Discovery in Databases
LPS	Learned Pattern Similarity
LSTM-FCN	Long Short Term Memory Fully Convolutional Network
MDDTW	Mahalanobis Distance-based Dynamic Time Warping measure
MLP	Multi-Layer Perceptron
MTS	Multivariate time series
MTSC	Multivariate time series classification
mv-ARF	Autoregressive forests for multivariate time series modeling
RDD	Resilient Distributed Datasets

ResNet	Residual Network
RNN	Recurrent Neural Network
SAX	Symbolic Aggregate approXimation
SDAEs	Stacked Denoising AutoEncoders
SMTS	Symbolic representation for Multivariate Time Series classification
ST	Shapelet Transform
SVM	Support Vector Machine
TapNet	Time series attentional prototype network
TSF	Time Series Forest
UEA	University of East Anglia
WEASEL	Word ExtrAction for time SEries cLassification
WEASEL+MUSE	Word Extraction for Time Series Classification extended with the Multivariate Unsupervised Symbols and Derivatives

Chapter I

PhD Dissertation

1 Introduction

Time series analysis aims to extract and understand all kinds of information of interest from any phenomenon that can be observed over a period of time. The main feature of this kind of process lies in the relationships between the time series current values and its past values [Fu11]. These relationships add an additional dimension on which the entire process depends, which is not present in traditional problems. The time dimension conditions the analysis performed and has to be considered throughout the complete process.

Nowadays, the surrounding environment is interconnected, providing a large number of information sources that generate an enormous volume of data over time. The monitoring of various parameters of a process over time generates multiple dimensions of the same time series, giving rise to multivariate time series and problems with a higher degree of complexity. Problems that until recently were addressed with closed models and very limited sets of variables now incorporate data from various sources together with machine learning models capable of adapting to a changing environment and offering very competitive results. Tasks such as predicting electricity consumption [DZY$^+$17], predicting stock performance and risks [BM15], or classifying abnormalities in electrocardiograms [PR15] have benefited from including both new data sources and different machine learning models.

At present, the amount of data to be processed is constantly increasing, so both the resources needed to process it and the need to adapt the machine learning models to these new dimensions are growing. The traditional computing model has limitations in computational and memory capacities that make it impossible to apply it to large-scale problems. The concept and research in Big Data emerge as a solution to these limitations to face these recent problems through a new distributed processing paradigm known as MapReduce [DG04]. This paradigm allows us to pool the individual resources of different computers and use them transparently. This feature allows us to obtain computing capabilities far superior to those of a supercomputer but at a much lower cost. The objectives of Big Data are mainly marked by the 3 Vs [RGFG$^+$18]: velocity, volume, and veracity, among other Vs. These purposes show the needs that this paradigm seeks to meet. Apache Spark and Hadoop are the main development frameworks.

All the concepts mentioned until now are contained in Knowledge Discovery in Databases (KDD) [FPSM92]. KDD refers to the whole procedure necessary to discover useful knowledge in a database. Such procedure has the following steps:

- **Target identification**: it requires understanding the application domain and the use of relevant prior knowledge.
- **Obtaining the desired dataset**: by selecting a specific dataset or a subset of variables or samples from it.
- **Data cleaning and preprocessing**:criteria for the treatment of noise and missing values, data preparation to obtain information from the data, identification, and adaptation of the time variable in the case of time series, among others.
- **Data reduction and projections**: search for particularly useful features when representing the data, which can be some of the original ones or other new ones obtained by performing different transformations.
- **Search for the type of data mining technique that fits the objectives of the problem**: classification, regression, clustering, prediction, etc.
- **Exploratory data analysis and hypothesis selection**: based on the structure of the data, types of variables, the problem to be solved, among others, we select which combination of models and parameters could be the most appropriate to address the problem.
- **Data mining**: run of the different hypotheses made, whose results can be significantly improved based on what was done in the previous steps.

- **Evaluation of the obtained results**: evaluation and interpretation of these results, being able to return to previous steps, if necessary, to obtain better results.
- **Use of the discovered knowledge**: application of this information to improve the performance of other systems or preparation of reports that allow access to the extracted knowledge to other people.

Data mining has a crucial role in this knowledge extraction process since it is able to express the knowledge embedded in the data by extracting patterns, rules, groupings, among others. The results obtained in this step are strongly related to the previously seen steps of the Knowledge Discovery process itself in Databases. The use of one technique or another depends directly on the type of problem, output variable, to be solved. Based on this, we have two main groups:

- **Supervised Learning**: the target variable is defined. The relationship between the input variables and the output variable is sought. Depending on the type of output variable, continuous or discrete, two subgroups are defined:
  - **Classification**: in this case, the output variable is discrete. This variable can only take one value from a previously defined finite set of possible values. An example of this type of problem would be a classification of a set of handwritten numbers, the output set being the integer values of the interval [0, 9].
  - **Regression**: in this case, the output variable is continuous. This variable can obtain any value of the infinite possible ones. The electricity consumption of a household based on the type of housing, surface area, number of appliances and their type, etc., is a typical example of this type of problem. The prediction of electricity consumption using past values of the consumption itself or other variables recorded over time gives rise to a specific case of regression, where the temporal order of the data has special relevance. These data are known as time series, and these regression problems are translated into forecasting problems, which have a number of additional constraints and conditions that differentiate them significantly from traditional regression problems.
- **Unsupervised Learning**: the target variable is not defined. This type of problem aims to find the implicit relationship between the data of the problem itself. Depending on the relationships sought, we can differentiate two groups:
  - **Clustering**: we look for groupings of instances throughout the dataset. These groupings must be as compact as possible and must be as far apart as possible.
  - **Association**: we look for relationships between different variables, assessing the support and confidence, among others, of these relationships.

This thesis is focused on time series analysis, specifically in supervised classification tasks. Although the time series classification field has a large number of approaches to deal with this problem, the proposals made in this field can be classified into three main groups: distance-based, feature-based, and deep learning.

The distance-based approach has a large number of proposals, as it is the most direct approach. It based this approach on a direct point-to-point comparison between different time series with static or dynamic distance measures [BC94]. Depending on the type of comparison measure applied, the complexity of the model can be very high. In this approach, we also have proposals that look for a certain sub-sequence of great interest within a time series that allows its classification within one group or another [YK09]. Other proposals take into account the number of repetitions of the same pattern in a time series [LL09]. There are even proposals that join different classifiers through complex ensembles [LTB18] in order to obtain the best possible results.

The feature-based approach focuses on transforming the original time series into a set of features that represent different behaviors present in the original time series. Some of these features can be mean, variance, length, etc. In this approach, we find two main groups. A first group focused on obtaining classifiers with high performance, using huge amounts of features, hundreds or even thousand [FLJ13][FJ14], although the models obtained have high complexity. The second group focuses on obtaining reduced sets of features [BDHL12][NAM01], maintaining a reduced complexity in the models obtained and giving greater importance to the interpretability of the results.

The approach based on Deep learning [FFW$^+$19][WYO17] can be applied both on the original time series and on the transformations based on feature extraction discussed above. For the first case, the structures used have to consider the temporal component of the data. In the second case, this component does not exist. For this reason, we can apply models with typical structures of any problem based on traditional feature vectors. In both cases, the models used are focused on obtaining the best possible results.

In time series classification problems, undesired phenomena, like trend or seasonality, that are usually treated before the final modeling of the time series in forecasting problems are not usually analyzed and are allowed to be modeled by the applied algorithm as an additional part of the time series to be processed [MA14][LJZ15]. These behaviors can even be determinant when classifying a particular group of time series, so the models used must be able to extract useful information from them.

The increase of dimensionality in these problems leads to a considerable increase in their complexity. The search for relationships of interest between the different variables of a multivariate time series is a non-trivial task for which most univariate approaches are not prepared. Considering the nature of Big Data environments, the generation of these kinds of problems with a considerable volume of data is reasonable. For this reason, there is a growing need for proposals that address this problem.

The present thesis addresses time series analysis in Big Data environments, focusing on classification tasks. First, a fully scalable time series classification algorithm for Big Data will be designed, which improves the results obtained in traditional environments and enables the use of traditional vector-based classification algorithms already implemented in Big Data. Second, a set of complexity measures and well-known time series features is proposed for extracting information of time series' structure to face time series classification problems. It allows the application of vector-based classifiers to time series problems, and improving the interpretability of models. In third place, a transformation based on well-known time series features applied to multivariate time series classification problems is presented. This transformation enables the application of traditional classification algorithms on multivariate time series problems. In addition, it allows the classifiers used to find the possible relationships of interest between the different variables of a multivariate time series. This last proposal focuses on improving interpretability in the multivariate time series domain, being able to identify variables of interest. Finally, a scalable and distributed transformation for univariate and multivariate time series based on well-known time series features for Big Data environments is proposed. This transformation, implemented according to the MapReduce paradigm, allows addressing time series problems in Big Data environments in a fully scalable way. This proposal significantly increases the limited amount of tools available for time series processing in Big Data environments.

The thesis is organized in two clearly differentiated parts: the doctoral thesis and the publications. In this first part, Section 1 describes the environment in which this thesis is developed. Section 2 explains the main concepts and the state-of-the-art on which this thesis is based. The justification and motivation of this thesis are shown in Section 3. Section 4 shows the objectives pursued by this thesis. The methodology followed during the development of this thesis is detailed in Section 5. Section 6 provides an introduction to each of the publications that compose this thesis. Section 7 explains the results obtained in each publication. The conclusions obtained in each work are shown in Section 8. Finally, in Section 9, the future lines of work that this thesis has left open are specified.

The second part shows the four articles that compose this thesis, following the order given by the proposed objectives:

- Distributed FastShapelet Transform: a Big Data time series classification algorithm.
- Complexity Measures and Features for Times Series classification.
- Multivariate times series classification through an interpretable representation.

- SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments.

Introducción

El análisis de series temporales tiene como objetivo extraer y comprender todo tipo de información de interés de cualquier fenómeno que se pueda observar a lo largo de un periodo de tiempo. La principal característica de este tipo de procesos reside en las relaciones existentes entre los valores actuales de una serie temporal y sus valores pasados [Fu11]. Estas relaciones añaden una dimensión adicional de la que depende todo el proceso, que no está presente en los problemas tradicionales. La dimensión temporal condiciona en gran medida el análisis realizado y ha de tenerse en cuenta a lo largo de todo el proceso.

En la actualidad una gran parte del entorno que nos rodea está interconectado, ofreciendo una gran cantidad de fuentes de información que generan un enorme volumen de datos a lo largo del tiempo. La monitorización de varios parámetros de un proceso a lo largo del tiempo genera múltiples dimensiones de una misma serie temporal, dando lugar a series temporales multivariantes y a problemas con un mayor grado de complejidad. Problemas que hasta hace poco se afrontaban con modelos cerrados y conjuntos de variables muy acotados, ahora incorporan datos de diversas fuentes junto a modelos de aprendizaje automático capaces de adaptarse a un entorno cambiante y ofrecer resultados muy competitivos. Tareas como la predicción del consumo eléctrico [DZY$^+$17], predicción del rendimiento y los riesgos de las acciones [BM15] o la clasificación de anomalías en electrocardiogramas [PR15] se han visto beneficiadas de la inclusión, tanto de nuevas fuentes datos, como de diferentes modelos de aprendizaje automático.

Hoy en día, la cantidad de datos a procesar no deja de aumentar, por lo que crecen tanto los recursos necesarios para procesarlos como la necesidad de adaptar los modelos de aprendizaje automático a estas nuevas escalas. El modelo de computación tradicional cuenta con limitaciones en la capacidad de cómputo y memoria que imposibilitan su aplicación en problemas de grandes dimensiones. El concepto y la investigación en Big Data surgen como una solución a dichas limitaciones para afrontar estos nuevos problemas, a través de un nuevo paradigma de procesamiento distribuido conocido como MapReduce [DG04]. Este paradigma permite unir los recursos individuales de diferentes ordenadores pudiendo utilizarlos de forma transparente. Esta característica nos permite obtener capacidades de cómputo muy superiores a las de un supercomputador, pero con un coste muy inferior. Los objetivos del Big Data quedan marcados principalmente por las 3 V [RGFG$^+$18]: velocidad, volumen y veracidad, entre otras V. Estos objetivos nos indican las necesidades que busca suplir dicho paradigma. Siendo Apache Spark y Hadoop los principales entornos de desarrollo.

Todos los conceptos mencionados hasta ahora se engloban dentro del Descubrimiento del Conocimiento en las Bases de datos (KDD en inglés) [FPSM92]. KDD hace referencia a todo el procedimiento necesario para descubrir conocimiento útil en una base de datos. Dicho procedimiento cuenta con los siguientes pasos:

- **Identificación del objetivo**: requiere de la comprensión del dominio de aplicación y la utilización de los conocimientos previos pertinentes.
- **Obtención del conjunto de datos deseados**: por medio de la selección de un dataset concreto o de un subconjunto de variables o muestras del mismo.
- **Limpieza y preprocesado de los datos**:criterios de tratamiento del ruido y de valores perdidos, preparación de los datos que permita la obtención de información de estos, identificación y adecuación de la variable temporal en el caso de las series temporales, entre otros.
- **Reducción de datos y proyecciones**: búsqueda de características especialmente útiles a la hora de representar los datos, pudiendo ser algunas de las originales u otras nuevas obtenidas por medio de diferentes transformaciones.
- **Búsqueda del tipo de técnica de minería de datos que se adecue a los objetivos del problema**: clasificación, regresión, clustering, predicción, etc.

- **Análisis exploratorio de los datos y selección de hipótesis**: en base a la estructura de los datos, tipos de variables, la problemática a solventar, entre otros, seleccionamos que combinación de modelos y parámetros podrían ser los adecuados para afrontar dicho problema.
- **Minería de datos**: ejecución de las diferentes hipótesis realizadas, cuyos resultados pueden verse significativamente mejorados en base a lo realizado en los pasos anteriores.
- **Evaluación de los resultados obtenidos**: valoración e interpretación de dichos resultados, pudiendo volver a pasos anteriores, en caso de ser necesario, para obtener unos mejores resultados.
- **Utilización del conocimiento descubierto**: aplicación de esta información para mejorar el desempeño de otros sistemas o preparación de informes que permitan acceder al conocimiento extraído a otras personas.

La minería de datos tiene un papel crucial en este proceso de extracción de conocimiento ya que es capaz de expresar el conocimiento incrustado en los datos por medio de la extracción de patrones, reglas, agrupamientos, entre otros. Los resultados obtenidos en este paso están fuertemente relacionados con los pasos vistos anteriormente del propio proceso del Descubrimiento del Conocimiento en las Bases de datos. La utilización de una técnica u otra depende directamente del tipo de problema, variable de salida, a resolver. En base a esto, tenemos dos grandes grupos:

- **Aprendizaje Supervisado**: la variable objetivo está definida. Se busca la relación entre las variables de entrada y la variable de salida. Dependiendo del tipo de variable de salida, continua o discreta, se definen dos subgrupos:
  - **Clasificación**: en este caso la variable de salida es discreta. Esta variable solo puede tomar un valor de un conjunto finito, previamente definido, de posibles valores. Un ejemplo de este tipo de problemas sería clasificación de un conjunto de números escritos a manos, siendo el conjunto de salida los valores enteros del intervalo [0, 9].
  - **Regresión**: en este caso, la variable de salida es continua. Esta variable puede tomar cualquier valor de los infinitos posibles. El consumo eléctrico de un hogar en base al tipo de vivienda, superficie, cantidad de electrodomésticos y el tipo de estos, etc., es un ejemplo típico de este tipo de problemas. La predicción del consumo eléctrico utilizando valores pasados del propio consumo u otras variables registradas en el tiempo da lugar a un caso específico de regresión, donde el orden temporal de los datos tiene especial relevancia. Estos datos se conocen como series temporales y estos problemas de regresión se traducen a problemas de predicción (forecasting en inglés), los cuales cuentan con una serie de restricciones y condiciones adicionales que los diferencian de forma muy significativa de los problemas de regresión tradicionales.
- **Aprendizaje No Supervisado**: la variable objetivo no está definida. Lo que se trata de encontrar en este tipo de problemas es la relación implícita existente entre los propios datos del problema. En función de las relaciones buscadas, podemos diferenciar dos grupos:
  - **Clustering**: se buscan agrupaciones de instancias a lo largo del dataset. Estas agrupaciones han de ser lo más compactas posibles y tienen que estar lo más separadas posible entre sí.
  - **Asociación**: se buscan relaciones entre diferentes variables, valorando el soporte y la confianza, entre otros, de dichas relaciones.

Esta tesis se centra en el análisis de series temporales, concretamente en tareas de clasificación supervisada. Aunque el campo de la clasificación de series temporales cuenta con un gran número de enfoques para abordar este problema, las propuestas realizadas en este campo se pueden clasificar en tres grandes grupos: basadas en distancia, basadas en características y deep learning.

El enfoque basado en distancia cuenta con una gran cantidad de propuestas, ya que es el enfoque más directo. Este enfoque se basa en la comparación directa, punto a punto, entre diferentes series temporales, con medidas de distancia estáticas o dinámicas [BC94]. Dependiendo del tipo de medida de comparación aplicada la complejidad del modelo puede llegar a ser muy elevada. Dentro de este enfoque tenemos también propuestas que buscan una determinada subsecuencia de gran interés dentro de una serie temporal que permitan su clasificación dentro de un grupo u otro [YK09]. Otras propuestas tienen en cuenta la cantidad de repeticiones de un mismo patrón en una serie temporal [LL09]. Incluso existen propuestas que unen diferentes clasificadores por medio de complejo ensembles [LTB18] con el objetivo de obtener los mejores resultados posibles.

El enfoque basado en características se centra en transformar las series temporales originales en un conjunto de caracteríticas que representen diferentes comportamientos presentes en las series temporales originales. Algunas de estas características pueden ser la media, la varianza, la longitud, etc. En este enfoque encontramos dos grandes grupos. Un primer grupo centrado en obtener clasificadores con un alto rendimiento, utilizando enormes cantidades de características, cientos e incluso miles [FLJ13][FJ14], aunque los modelos obtenidos cuenten con una elevada complejidad. El segundo grupo se centra en obtener conjuntos reducidos de características [BDHL12][NAM01], manteniendo una complejidad reducida en los modelos obtenidos y dando una mayor importancia a la interpretabilidad de los resultados.

El enfoque basado en Deep learning [FFW$^+$19][WYO17] puede aplicarse tanto sobre las series temporales originales como sobre las transformaciones basadas en la extracción de características comentadas anteriormente. Para el primer caso, las estructuras utilizadas han de tener en cuenta la componente temporal de los datos. En el segundo caso dicha componente no existe. Por este motivo podemos aplicar las estructuras típicas de cualquier problema basado en vectores de características tradicionales. En ambos casos los modelos utilizados se centran en la obtención de los mejores resultados posibles.

En los problemas de clasificación de series temporales, los fenómenos no deseados tratados antes del modelado final de la serie temporal en problemas de predicción, no suelen ser analizados, y se permite su modelado por parte del algoritmo aplicado como una parte más de la serie temporal a procesar [MA14][LJZ15]. Estos comportamientos pueden llegar incluso a ser determinantes a la hora de clasificar un determinado grupo de series temporales, por lo que los modelos utilizados han de ser capaces de extraer información útil de ellos.

El aumento de dimensionalidad en estos problemas conlleva un aumento de su complejidad considerable. La búsqueda de relaciones de interés entre las diferentes variables de una serie temporal multivariante es una tarea no trivial para la que la mayor de las propuestas univariable no está preparadas. Dada la naturaleza de los entornos Big Data, la generación de este tipo de problemas con un volumen de datos considerable es normal, por lo que hay una creciente necesidad de propuestas que afronten esta problemática.

La presente tesis aborda el análisis de series temporales en entornos Big Data, centrándose en tareas de clasificación. En primer lugar, se diseña un algoritmo de clasificación de series temporales totalmente escalable para Big Data, que mejora los resultados obtenidos en entornos tradicionales y permite el uso de algoritmos de clasificación tradicionales basados en vectores ya implementados en Big Data. En segundo lugar, se propone un conjunto de medidas de complejidad y características de series temporales bien conocidas para extraer información de la estructura de las series temporales para afrontar problemas de clasificación de series temporales. Esto permite la aplicación de clasificadores basados en vectores a problemas de series temporales, y mejorando la interpretabilidad de los modelos interpretables. En tercer lugar, se presenta una transformación basada en características conocidas de las series temporales aplicada a problemas de clasificación de series temporales multivariantes. Esta transformación permite aplicar los algoritmos de clasificación tradicionales a problemas de series temporales multivariantes. Además, permite que los clasificadores utilizados encuentren las posibles relaciones de interés entre las diferentes variables de una serie temporal multivariante. Esta última propuesta se centra en mejorar la interpretabilidad en el dominio de las series temporales multivariantes, pudiendo identificar las variables de interés. Finalmente, se propone una transformación escalable y distribuida para series temporales univariantes y multivariantes basada en características conocidas de las series temporales para entornos Big Data. Esta transformación, implementada según el paradigma MapReduce, permite abordar problemas de series temporales en entornos Big Data de forma totalmente escalable. Esta propuesta incrementa significativamente la limitada cantidad de herramientas disponibles para el procesamiento de series temporales en entornos Big Data.

Esta tesis está organizada en dos partes claramente diferenciadas: la tesis doctoral y las publicaciones. En esta primera parte, en la Sección 1 se describe el entorno sobre el que se desarrolla esta tesis. En la Sección 2 se explican los principales conceptos y el estado del arte en los que basamos esta tesis. La justificación y motivación de esta tesis se muestra en la Sección 3. En la Sección 4 se muestran los objetivos que persigue esta tesis. La metodología seguida durante el desarrollo de esta tesis se detalla en la Sección 5. En la Sección 6 se ofrece una introducción a cada una de las publicaciones que componen esta tesis. En la Sección 7 se explican los resultados obtenidos en cada publicación. Las conclusiones obtenidas en cada trabajo se muestran en la Sección 8. Por último, en la Sección 9, se especifican las futuras líneas de trabajo que esta tesis deja abiertas.

En la segunda parte se muestran los cuatro artículos que componen esta tesis, siguiendo el orden dado por los objetivos propuestos:

- Distributed FastShapelet Transform: a Big Data time series classification algorithm.
- Complexity Measures and Features for Times Series classification.
- Multivariate times series classification through an interpretable representation.
- SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments.

2 Preliminaries

This section shows the main concepts on which this thesis is based. First, in Section 2.1, an introduction to the time series is presented. Section 2.2 introduces the concept of Big Data and its environments. Section 2.3 contains an analysis of the time series classification problem and its singularities in Big Data environments. Sections 2.5 studies the features-based time series classification approach in univariate time series environments. Finally, Section 2.4 analyses the state-of-the-art of multivariate time series classification.

2.1 Time Series

A time series is composed of a succession of values registered over time. Some common examples of time series data are the stock market values, the registers of energy consumption, the heart rate recorded by the smartwatches, etc. The main feature of this type of data is the relationships between the current and past values of the time series. Those relationships cannot be ignored, and they must be considered when a time series is processed [Cry86]. This feature limits the traditional statistical methods application, which assumes that nearby observation is independent and identically distributed. In a classical vector-based dataset, we can disorder the features o even eliminate some of them, but this cannot be done in time series data because we would destroy the temporal relationship between the values of the time series.

In typical time series, the current value is strongly linked to the nearest past values but is less related to the most distant values. It means that there is a relationship between nearby values. In Figure 1, we can see this behavior. In the time series field, it is typical to represent the values of the time series on the y-axis and the time indices on the x-axis. For the red dots, we know that the next subsequent values will be a little high than the current ones. In the case of the blue dots, we can see that they will be near to a central value, the mean, with minor variations. On the other hand, the green dots do not show a relationship with the red or blue dots, or even between the different green dots groups. Of course, there are some cases where the behavior observed is different even we can find a strong relationship between those points, but the most usual behavior is that those relationships decrease when the points are more distant. The aim of figure 1 is to show the importance of the time variable in time series data.

Two main approaches can be considered in time series analysis, which are not exclusive between them: time domain approach and frequency domain approach [SSS00]. In the time domain approach, we focus on lagged relationships between the different points of a time series (for example, if yesterday rains, how it affects the forecasting of rain probability for tomorrow?). On the other hand, the frequency domain analyses the cycles of a time series (for example, in medical areas, we study the cycles in patients' health through the periods of their electrocardiogram, if they are normal or abnormal).

Time series data exhibit behaviors which are not present in other types of data. The most relevant are trend and seasonality. The trend component indicates a direction of increase or decrease of the time series values. We have a positive trend when the time series values increase with time and a negative trend when the time series values decrease with time. We can observe a positive trend in the red points in Figure 1. A typical use case of this component is showed in the stock market. If you have a positive trend, you know that the action price is growing in time, because of this, if you want to gain benefits, you have to buy at earliest of this trend and sell at the end.

The seasonality component represents cycles of values repetitions in time. For example, the electric consumption of a house has multiple seasonality components:

- A seasonality by day, because every day in the morning, evening, and night you have typical, but different between them, consumption patterns.
- A seasonality by week, because there are significant differences between the Monday and the Sunday consumptions, but every Monday, you probably will consume a similar quantity of energy.

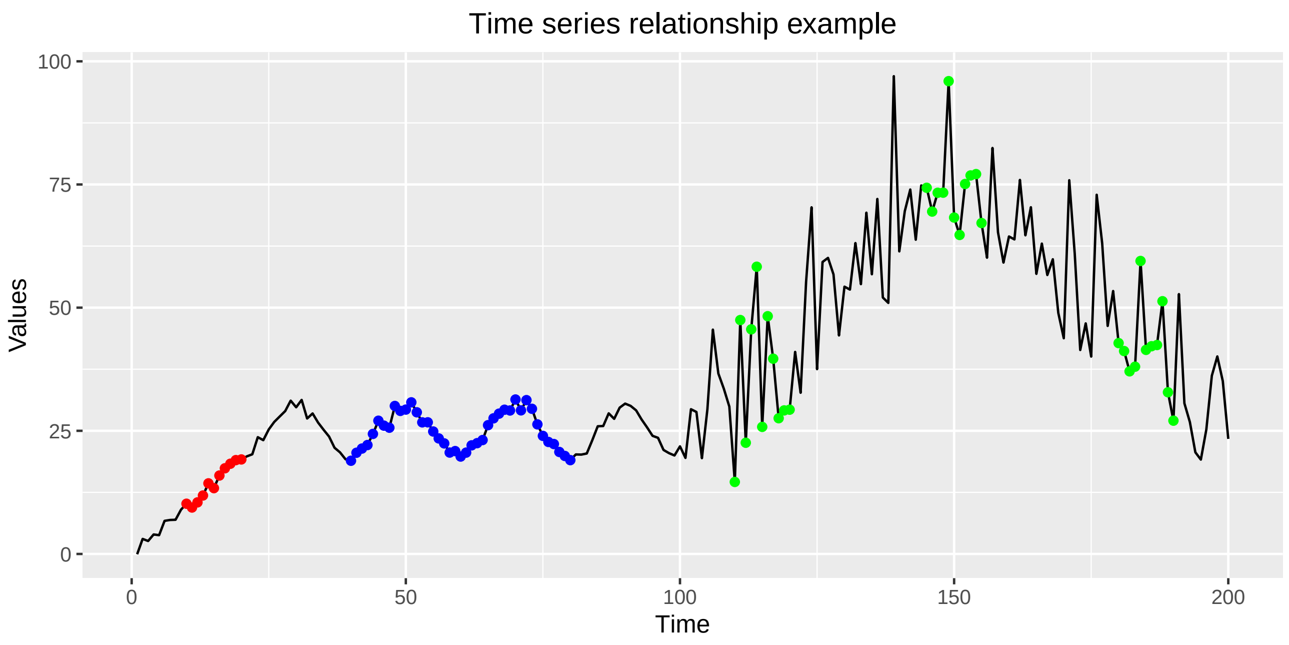


Figure 1: Time series relationships example.

- A seasonality by month, by each season, and by year, because in each of those time periods, you will have different behavior and climate conditions that strongly affect your consumption patterns.

Behaviors like trend or seasonality, among others, prevent us from obtaining a stationary time series. Stationarity [GR08] is the main desirable property of a time series. It means that other properties are stable over time. A significant share of the time series literature is aimed to obtain stationary time series [MWH08][KWH12]. Although in some cases or specific types of problems, the applied models are capable of turning non-stationary components into additional information that improves the obtained results. Due to this, the choice of the way to follow depends on each problem, being impossible to provide a general solution for the time series field.

2.2 Big Data

In an interconnected world, where everything is continuously producing data over time, time series data is one of the most generated types of data. Huge amounts of data are generated, stored, and need to be processed. The emergence of the MapReduce paradigm [DG04] made available to legions of programmers an easy way to tame clusters composed of thousands of computers. This new programming paradigm —inspired in a classic functional decomposition— has enabled the efficient processing of huge loads of data relieving the programmers from deep technical concerns on parallelization, data distribution, load balancing, and fault tolerance. Engineers at Google created the concept, and many implementations were developed by groups of programmers [LKRH15], and of course, evolving it to improved ideas.

In Figure 2, the application of MapReduce to a simple problem is illustrated: word count problem. The green box represents the operations and data on which users interact. The blue box represents internal operations, which are transparent for the users. The basic idea is to use pairs key-value with map and reduce primitives. A unique key identifies each data instance of the problem, and it can distribute the data and the required computation for the cluster transparently for the programmer. The map transformation applies an operation over the data in a distributed way, generating a set of intermediate key/value pairs. Then we can combine the transformed data without problems applying a reduce action to all values with the same key. This simple schema let us parallelize large computation easily. In addition, it provides us with a robust fault tolerance mechanism.

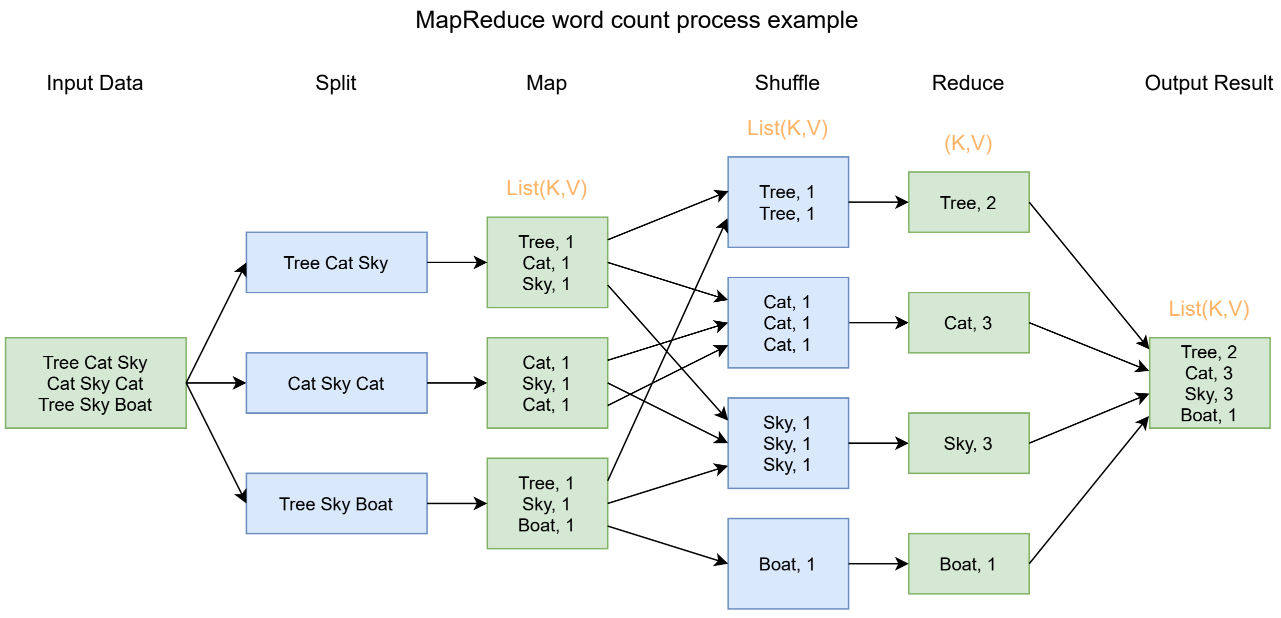


Figure 2: Workflow of MapReduce application on typical word count problem.

Apache Hadoop is the best known open-source framework based on MapReduce to work in Big Data [Whi12]. It was the first to offer a distributed and robust environment to process Big Data problems. Hadoop is programmed in Java, thus it can be deployed in any machine capable of running Java. It allows users to combine multiple and different machines in a heterogeneous and powerful cluster. The MapReduce paradigm is excellent when applicable, but it suffers from several limitations that restrict the type of problems to which it can be applied. Its limitations include:

- Intermediate results from each computing step have to be written to disk. This process is very slow in comparison with operations performed in the main memory.
- Hadoop is oriented to batch processing. It cannot process streaming data or online data due to its latency.
- Poor efficiency in iterative tasks. Hadoop does not support cyclic or iterative data workflows. Due to this, the intermediate steps between iterations explode the running time. Iterative behavior is fundamental in machine learning tasks.

The open-source framework Apache Spark [Spa16] was developed to solve the main limitations of Hadoop. Spark still provides the main advantages of Hadoop for Big Data environments: distributed processing, MapReduce, robustness, fault tolerance, among others. In addition, it provides in-memory processing, allowing to save in memory the data at the start of the process and the intermediate steps dramatically increasing the speed of processing data. Due to its in-memory processing, Spark can face iterative processes without problems and with high efficiency. The speed-up offered by Spark concerning Hadoop allows it to face online data and streaming data problems with a good performance. However, new proposals with improved performance for these particular tasks are arising: for example, Apache Flink [CKE$^+$15][Fli19] provides even more performance in those environments.

The new abilities provided by Spark are supported by a new distributed data structure known as Resilient Distributed Datasets (RDD) [ZCD$^+$12]. The RDDs just can be generated from data in stable storage or other

RDDs. The user can apply two types of operations on RDDs: transformations or actions. Operations like map, filter, and join are some examples of transformations. But Spark does not apply the transformations immediately over the RDD. Each RDDs has enough information about how it reaches its actual state (its lineage), and it just runs all transformations when an action is called over the RDDs. An example of action is the reduce operation, typical in MapReduce paradigm. This lineage provides excellent fault tolerance capabilities because it has recorded every operation performed over the RDDs, letting us recover the RDDs at any point of this lineage.

There are two additional capabilities of RDDs that let the users optimize the reuse of datasets and control the data distribution: persistence and partitioning. The first one, persistence, lets the user choose the storage strategy for RDDs. If the user will apply an iterative process over an RDD, it is advised to use an only-memory strategy. If the user has memory limitations, the hybrid memory and disk strategy is the best option. On the other hand, if the dataset will not receive intensive use, the only-disk strategy lets the user use the memory in other prioritized tasks. The partitioning lets the user distribute the data across the cluster based on a key in each record, obtaining the capability of use placements optimizations with high robustness.

Due to the capabilities provided by Spark to process Big Data problems, the community has developed the main machine learning algorithms oriented to scalability. Those proposals are included in the MLLIB [MBY+16] when they are considered robust enough or spark packages [Pac19] if they are in their early versions. Although the community has invested a high effort to bring as many as possible algorithms to the Big Data environment, this is not a simple task. Even now, there is a lack of proposals in Big Data environments. For this reason, it is a completely open field of research.

2.3 Univariate Time Series Classification

The time series classification aims to find patterns of interest within a time series dataset that allows distinguishing among the existing groups. Problems as finding a representative pattern of a particular cardio-vascular condition in a patient [CVA19], patterns of electricity consumption typical of a fraud [OBPR21], anomalous driving patterns [LJZ15], among others, are typical examples of this type of problem.

Time series classification can be divided into three main areas, based on the approach used to tackle the problem: **Distance-Based approach**, **Feature-Based approach**, and **Deep Learning approach**.

The **Distance-Based approach** was the first to be developed because of the initial simplicity of its application. This approach is based on the comparison of time series or sub-sequences of time series with each other. Depending on the tools used, there are six main groups of methods:

- **Methods that use all the values that compose a time series**: they use multiple measures of similarity and distance. The reference algorithm in this type of method is the One-Nearest-Neighbor + Dynamic Time Warping (1NN+DTW) [BC94], which classifies the input time series with the same label as the most similar time series. In addition, it uses an elastic distance measure such as DTW, which searches for the correspondence between points in both time series that minimizes the distance between them.

- **Phase-dependent methods**: these methods use time series sub-sequences to compare different time series, using the same sub-sequence intervals across all the time series that compose the problem. Time Series Forest (TSF) [DRTV13] and Contract Random Interval Spectral Ensemble (c-RISE) [FLB19] are some of the best-known methods of this approach. Currently, c-RISE offers competitive results concerning the state-of-the-art.

- **Phase-independent methods**: this approach includes Shapelets [YK09], which are sub-sequences extracted from the time series, regardless of the position in which they are found, and which serve to classify whether a time series belongs to one class or another based on the presence or not of this sub-sequence in the processed time series. The proposals within this approach are very varied, ranging from a simple decision tree based on Shapelets [YK09][RK13], logical combinations of Shapelets [MKY11] that increase their expressiveness or creation of distance matrices to Shapelets that allow vector-based classifiers to be applied to time series problems [LDHB12]. This last method is the Shapelet Transform (ST) and offers competitive results concerning the state-of-the-art.

- **Dictionary-based methods**: in these methods, the number of times a particular pattern is repeated in a time series is especially relevant [LL09][Sch15]. A typical example of this type of proposal is Bag of Patterns (BOP), which creates a dictionary of simplified subsequences of the original time series known as Symbolic Aggregate approXimation (SAX) [LKWL07]. One of the latest proposals in this field is Word ExtrAction for time SEries cLassification (WEASEL) [SL17a] which extracts discriminative features concerning the class, using statistical tests to confirm the final selection.

- **Model-based methods**: these models are based on fitting a model to each input time series and comparing the similarity between the different models obtained to classify the processed time series [BJ14][CTTY13]. This approach is interesting to apply with particularly long time series.

- **Ensemble-based methods**: these methods are focused on using the results offered by different classifiers to provide a final classification. Currently, the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) [LTB18] is the proposal that provides the best results in the state-of-the-art, being an ensemble composed of classifiers from different domains across five large modules. These modules offer a probability of correctly classifying a time series and obtain weight in the final decision according to their accuracy on the training set. It is an expensive method.

The **Feature-Based approach** can be split into two main subgroups, depending on whether the experimental or theoretical approach predominates in the proposals.

In the **experimental approach**, we find proposals that use extensive sets of extractable features from a series of data, not necessarily specific to the time series, with the aim of extracting as much information as possible. Once these features have been obtained, there are two options:

- First, a problem is addressed with the information available in these extensive sets of features. For example, it has been attempted to find possible underlying structures between different time series that allow grouping them through the extracted features [FLJ13]. Working with a large set of features is a complex task, and most of the works perform a visual study of the results as best as possible.

- Second, multiple authors try to reduce the initial set of features, making it more manageable and trying to obtain a new subset as general as possible that can be effectively applied to a wide variety of problems [FJ14]. In this case, there are proposals such as CAnonical Time-series CHaracteristics (catch22) [LSK$^+$19b], which offers a set of 22 features that have been shown to obtain the best classification results on an extensive set of datasets. The major drawback of this approach is that new behaviors may always appear that have not been considered during the experimentation, and therefore fall outside the range of behaviors that the proposed measures can capture. This issue affects the proposal performance and is difficult to solve because of the purely experimental approach used. Recently, the Feature and Representation Selection (FEARS) method has been proposed [BGL$^+$19]. FEARS obtains an alternative representation of the time series based on the calculation of derivatives, cumulative integrals, power spectrum, among others. This proposal starts from a well-known set of time series features, but the features selection criteria is focused on obtaining the most informative set possible.

The **theoretical approach** focuses on selecting features especially representative of the time series with the aim of capturing behaviors of interest of the time series expressed through well-known features. We can find proposals that start from four familiar features such as mean, typical deviation, skewness, and kurtosis, which are able to deal with the problem of classification of synthetic control chart patterns used in the statistical process control [NAM01]. Proposals focused on improving accuracy using ensembles of classifiers trained with data from different representations of the time series [BDHL12], such as power spectrum, autocorrelation function, and a principal components space. We can also find works that use these well-known features to create synthetic time series with specific desired behaviors expressed through these features [KHL+18].

The **Deep Learning** approach is dominated by proposals that seek to improve the accuracy results obtained. In this approach, we can differentiate two types of proposals: **Generative Models** and **Discriminative Models**.

In **Generative Models**, usually, there is an unsupervised training step previous to the classifier learning phase. Depending on the approach, two subgroups can be distinguished: **Auto Encoders** and **Echo State Networks**.

In the **Auto Encoders** subgroup, we can find proposals like Stacked Denoising AutoEncoders (SDAEs) [BYAV13], which models the time series before the classifier is included in an unsupervised pre-training step, or Recurrent Neural Network (RNN) Auto Encoder [RT18], which generates time series in a first step, then uses the learned representation to train a traditional classifier. In contrast, the **Echo State Networks** reconstruct time series and use the learned representation in the space reservoir for classification. These networks are useful to define kernels based on the learned representations. Then apply Multi-Layer Perceptron (MLP) or Support Vector Machine (SVM) as a classifier.

The **Discriminative Models** applied in supervised problems are able to learn the mapping between the time series' values. Then, they return the probability distribution over the class variable of the problem. **Feature Engineering** and **End-to-End** are the two subgroups that compose this approach. In the **Feature Engineering** subgroup, we can find time series transformations to images, using different techniques such as Markov transition fields [WO15] or recurrence plots [HGD18], and uses this information as input of a deep learning discriminating classifier [NTAGA18]. In contrast, the **End-to-End** approach incorporating feature learning while adjusting the discriminative classifier as a typical procedure.

2.4 Multivariate Time Series Classification

Multivariate time series classification (MTSC) has received considerable attention in recent years [RFL+20]. The initial proposals made in this field are multivariate versions of the main univariate time series classification methods. New, specific methods for the multivariate case are emerging. These new proposals are able to find relationships of interest between the different variables that compose a multivariate time series (MTS) and exploit them to improve their classification. In this field, we can identify three main approaches: the distance-Based approach, the Feature-Based approach, and the Deep Learning approach.

In the distance-based approach, we can find the multivariate version of One Nearest Neighbor + Dynamic Time Warping (1NN+DTW). In the MTS case, this proposal focuses on minimizing the distance between two MTS, which is calculated as the cumulative distance between each variable composing the two processed MTS. Since we are in a multivariate environment, two clearly differentiated approaches arise within this proposal, Independent Warping ($\mathrm{DTW_I}$) and Dependent Warping ($\mathrm{DTW_D}$), together with a third approach that seeks to correctly choose between the two previous ones based on a threshold found from the training data. This third approach is known as Adaptive Warping ($\mathrm{DTW_A}$) [SYHJ+17].

- In $\mathrm{DTW_I}$, the best possible alignment or temporal warping is sought for each variable independently. The final distance is obtained from the sum of the distances obtained for each variable.
- In $\mathrm{DTW_D}$, it is supposed that the best possible alignment is the same for all the variables that compose the MTS. Here, the distance matrix is not obtained by measuring the distance between two points. But, in this case, we calculate the Euclidean distance between the vectors composed by the points of each variable.
- $\mathrm{DTW_A}$ evaluates the two previous cases, $\mathrm{DTW_I}$, and $\mathrm{DTW_D}$, using cross-validation. During this process, this method separates the instances correctly classified by each approach. Then, $\mathrm{DTW_A}$ calculates information gain for each subset of data, and a threshold is calculated as a function of both information gains. Finally,

each instance is classified with one distance measure or the other in a way that maximizes the probability of obtaining a correct classification.

Another proposal related to DTW is the Mahalanobis Distance-based Dynamic Time Warping measure (MDDTW) [MLWG16], which uses the Mahalanobis distance to calculate the local distance between different vectors in the MTS case. MDDTW is able to find relationships of interest between each variable and the class to which it corresponds.

In the Feature-Based approach, we can find a wide variety of proposals, from multivariate generalizations of feature extraction techniques to ensembles with very different approximations:

- HIVE-COTE is applied to multivariate environments by simply processing each problem variable as a univariate time series problem [RFL$^+$20]. The individual predictions made by each internal classifier for each problem variable are combined, creating a probability distribution for each internal classifier.
- Canonical interval forest (CIF) [MLB20] is an ensemble composed of time series tree classifiers [DRTV13] based on the feature set proposed by Canonical Time-Series Characteristics (Catch22) [LSK$^+$19a] and extraction based on phase-dependent intervals of simple summary statistics.
- Word Extraction for Time Series Classification extended with the Multivariate Unsupervised Symbols and Derivatives (WEASEL+MUSE) [SL17b] is composed of a combination of univariate time series extraction and processing techniques, with a selection process of the most representative features, which can offer competitive results in multivariate. WEASEL+MUSE was considered the state-of-the-art for a long time, as it obtained the best results against its direct competitors: Autoregressive forests for multivariate time series modeling (mv-ARF) [TB18], Generalized Random Shapelets Forests (gRSF) [KPB16], Symbolic representation for Multivariate Time Series classification (SMTS) [BR15], and Learned Pattern Similarity (LPS) [BR16]. For this comparison, 20 datasets obtained from the database from [Bay17] were used.
- ROCKET [DPW20][RFL$^+$20] is currently considered the best state-of-the-art algorithm. This proposal performs simple linear classifiers using random convolutional kernels, being able to provide the best results with computational times that are at least an order of magnitude lower than those of its competitors. These conclusions have been obtained in an extensive study of 26 of the 30 datasets available in the new reference repository in the MTSC field, the University of East Anglia (UEA) repository [BDL$^+$18].

In the Deep Learning approach, we can find all kinds of new proposals. The ease with which neural networks can handle additional dimensions in model definition and implementation makes them easily applicable to the MTSC problem.

- A clear example of the simplicity of extending a univariate solution to a multivariable environment is the extension of the Long Short Term Memory Fully Convolutional Network (LSTM-FCN) and Attention LSTM-FCN (ALSTM-FCN) [KMDH19] to a multivariate environment. The authors did this conversion through the inclusion of a squeeze-and-excitation block in the fully convolutional block, improving the accuracy of the whole proposal.
- Residual Network (ResNet) [WYO17] can be considered as the baseline method of the MTSC in the Deep Learning approach. It is a simple network composed of three consecutive blocks. Each of these blocks contains three convolutional layers, which are directly connected by residual connections that add the input of each block to its output.
- Inception Time [FLF$^+$20] is an ensemble of 5 different Inception networks initialized randomly. Each Inception network classifier contains two different residual blocks, while ResNet was composed of three. These blocks are named Inception modules [SLJ$^+$15]. In addition, these blocks hold the residual connections and are followed by the global mean and softmax layers. One of the main features of this approach is the use of a bottleneck layer that reduces the dimensionality of the multivariate time series to be processed.

- Time series attentional prototype network (TapNet) [ZGLL20] is a proposal that addresses the MTSC problem in the MTS domain. This proposal is composed of three key steps. First, it randomly selects a reduced number of input MTS variables and evaluates their effects regarding the output class. Second, it applies a reduction of the number of values representing each time series. Finally, TapNet generates, only in the training set, a candidate prototype for each class. The candidate prototype of each class must minimize the distance of the members of its class to it, and the distance between candidate prototypes must be maximized.

2.5 Time Series Classification in Big Data Environments

In recent years, different proposals have emerged in the time series processing field to address the growing complexity and volume of data to be processed. Most of the proposals made have focused on obtaining increasingly efficient algorithms capable of processing the largest amount of data in the shortest possible time while maintaining competitive (accuracy) results. A simple example of this type of proposal is the version of the Dynamic Time Warping (DTW) distance measure capable of processing a trillion time series subsequences in a short lapse of time [RCM+13]. DTW had a high computational complexity up to that moment, so being able to process numerous time series subsequences in a short time when it was thought impossible, was a milestone in the processing of large sets of time series.

From that moment on, more proposals emerged focused on offering more efficient algorithms capable of processing larger sets of time series: the Proximity Forest algorithm [LSP+19], which provides a scalable distance-based classifier for time series, a Gaussian process modeling oriented to process astronomical time series in a fast and scalable way [FMAAA17], or the FastShapelet (FS) proposal [RK13] which offers a significant reduction of the complexity of the original Shapelet proposal [YK09] in exchange for a reduction of the results performance of the obtained from accuracy, among other proposals. The emergence of these new proposals is the answer to the increasing demand for algorithms effective in Big Data. An extension of classical time series classification methods is not usually possible or effective since scalability, namely a linear complexity, is a strong requirement that most methods do not meet.

As mentioned above, there are multiple Big Data Frameworks, with Apache Spark being one of the most popular. However, the overall offer for time series classification methods is severely limited. For example, in MLlib, Apache Spark's scalable machine learning library, we can only find a single streaming linear regression model developed to work with streaming data. Apart from the above algorithm, MLlib has no specific algorithms to perform classification, clustering, or forecasting tasks on time series.

Apart from the MLlib, Spark has an unofficial package repository, spark-packages [Pac19], where, before this thesis development, only one proposal for processing time series could be found: spark-timeseries[1]. This package provides basic tools for modeling time series in a distributed way. These tools are mainly oriented to forecasting tasks.

Based on what we have seen so far, we can assert that the demand for time series processing tools in Big Data environments is anything but satisfied.

[1]`https://spark-packages.org/package/sryza/spark-timeseries`

3 Motivation

As we have discussed in the previous sections, we currently live in an interconnected world that generates and stores large amounts of information over time, which exceeds the processing capabilities of traditional systems. The time component adds an additional variable with enough complexity to differentiate the time series analysis field from the rest. It is obvious that there is a need for time series analysis techniques in Big Data environments. In addition, an interconnected world also increases the number of available information sources, which increases the number of variables that compose the problems to be dealt with, thus increasing their complexity. For this reason, it is necessary to develop techniques capable of processing multivariate time series that can be applied in massive data environments.

Time series analysis, including time series classification tasks, is becoming increasingly important, as it allows to identify what type of event is occurring in a time interval. An increasing number of users are demanding the interpretability of the results. Many users need to know the reasoning behind their decision because they work in particularly critical and sensitive fields. Since most classic proposals are not directly applicable to Big Data environments, the demand for solutions in such environments is growing. The complexity and size of Big Data problems are the main stumbling block in their development.

To boost the effectiveness of the tools for time series analysis, classification and increasing the interpretability of results in Big Data environments, we should address the following issues:

- In Big Data environments, it is necessary to develop new tools to address the time series analysis and classification problems, as there are currently very few options. Due to their complexity, most of the traditional techniques are not applicable to massive data environments. Even their extrapolation from traditional to Big Data environments is a complex task or practically impossible in most cases.
- The interpretability of results is a tool with huge potential. It allows us to understand the results obtained by our model and enables us to explain to others how our model works and how it makes its decisions. In Big Data environments, the interpretability of the results becomes more critical due to the complexity, variability, and dimensionality of the processed data. For these reasons, the development of tools that improve the interpretability of our results and that are applicable in both traditional and massive data environments is a task that is increasingly in demand.
- The increase in the amount of data to process associated with Big Data environments is also related to an increase in the number of variables to be processed and the complexity of the problem. For this reason, it is necessary to develop techniques that can process multivariate time series in Big Data environments.

All these topics are encompassed in the subject of this thesis: Time series analysis in Big Data environments.

4 Objectives

Once the context of the research top has been clearly stated, we proceed to set the objectives. The general objective can be defined as research to design new algorithms for time series analysis in Big Data. This goal can be broken down into some more specific ones:

- **To design new time series classification methods in Big Data.** Because of the relatively small set of proposals to this end, some new algorithms are needed. Our main goals are to provide the first distributed time series classification algorithm in Big Data, and offering the possibility of applying classical vector-based classification algorithms, already existing in Big Data environments, to time series classification problems.
- **To enhance the interpretability of time series models.** Most models built to classify or describe time series are mainly or uniquely targeted at the highest possible performance. Interpretability of models has been frequently not considered. The final goal is to obtain a new time series representation, based on well-known time series features, that captures intrinsical time series' behaviors and increases the results interpretability of interpretable models. The proposed representation lets us apply traditional vector-based classifiers to time series problems, increasing significantly the number of tools to face this type of problem.
- **To design new multivariate time series classification interpretable methods.** Due to the existing proposals in this field have high complexity with low interpretability. The aim is to propose a multivariate time series representation based on the new univariate time series representation proposed by ourselves previously, which lets us use traditional vector-based classifiers on multivariate time series problems, increases the results interpretability of interpretable modes, and provides competitive results compared to the main algorithms of the state-of-the-art.
- **To apply the developed methods to real world problems.** Because of the lack of time series analysis proposals in Big Data. Based on the proposed new time series representation and the multivariate transformation, our aim here is to obtain a competitive, interpretable, and highly scalable proposal built on the MapReduce paradigm, which lets us apply the already available traditional vector-based algorithms in Big Data to time series problems.

5 Methodology

The context and goals set for this theses fall completely within the scientific field. Thus, the traditional scientific method will be applied. This method is composed of the following steps:

1. **Observation**: multiple studies of time series analysis and forecasting tasks as well as the issues of their application in massive data environments and the search for solutions offered by Big Data technologies and distributed processing.

2. **Formulation of hypothesis**: proposal of time series analysis methodologies, new classifiers, and transformations that improve data quality and the subsequent data mining tasks. The new proposals must satisfy the objectives stated in the previous section. Thus, they can be applied in massive data environments.

3. **Compilation of observations**: compilation of the results obtained through proposals implementations on Big Data datasets. Properties such as performance, accuracy, efficiency, and scalability, among others, will be measured and considered in the design.

4. **Hypothesis contrasting**: a comparison of the obtained results with the main proposals of the state-of-the-art in order to evaluate the quality of the new proposals.

5. **Hypothesis confirmation or refutation**: acceptance or rejection and modification, if necessary, of the developed proposals as a consequence of the experiments carried out and the results obtained. The previous steps could be repeated to propose a new hypothesis to be tested.

6. **Scientific thesis**: extraction, redaction, and acceptance of the conclusions obtained throughout the research process performed. All the proposals and results gathered during the development of the research must be collected and synthesized into a memory of the thesis.

6 Summary

In this section, we summarize the publications associated with this thesis. After this section, in section 7, we explain and develop the main results obtained by these proposals. Next, we show a list of the journal publications which contain the research carried out and the results obtained:

- F. J. Baldán, J. M. Benítez. Distributed FastShapelet Transform: a Big Data time series classification algorithm. Information Sciences, 496, 451-463 (2019). DOI: `https://doi.org/10.1016/j.ins.2018.10.028`
- F. J. Baldán, J. M. Benítez. Complexity Measures and Features for Times Series classification. Submitted.
- F. J. Baldán, J. M. Benítez. Multivariate times series classification through an interpretable representation. Information Sciences, 569, 596-614 (2021). DOI: `https://doi.org/10.1016/j.ins.2021.05.024`
- F. J. Baldán, Daniel Peralta, Yvan Saeys, J. M. Benítez. SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments. International Journal of Computational Intelligence Systems, (2021). Accepted.

The remainder of this section is organized following the objectives indicated in Section 4 and their respective publications. First, Section 6.1 details the characteristics of the shapelets-based time series classifier developed for Big Data environments. Section 6.2 presents a set of measures of complexity and well-known time series features capable of extracting information of interest about the behavior of the time series. Section 6.3 discusses the interpretability improvements obtained from the proposed time series feature-based transformation. Finally, Section 6.4 shows the successful performance of the proposed set of features in Big Data environments through the MapReduce paradigm.

6.1 Big Data time series classification algorithm

The leading Big Data frameworks, such as Apache Spark or Apache Hadoop, do not include a wide variety of options for addressing time series classification problems. The main state-of-the-art algorithms for time series classification have a high computational complexity, and most of them have behaviors and restrictions that prevent their application to massive data environments [BLB^{+}17]. The creation of time series classification algorithms is not a trivial task and, in some cases, requires certain concessions in their development to be implementable in Big Data environments. In this way, we can obtain proposals that offer a performance and results very similar to those of the original sequential proposal and that meet the scalability requirement necessary to be used in Big Data environments.

In this work, we have developed the Distributed FastShapelet Transform (DFST) algorithm to address time series classification problems in Big Data environments —programmed in the Apache Spark platform. In this proposal, we combine the low computational complexity of the FastShapelet algorithm with the basic idea of using shapelets as input features to a traditional classifier, as proposed in the Shapelet Transform (ST).

We have evaluated the performance of our proposal by studying both the accuracy and scalability obtained on Big Data synthetic datasets. Since there were no really large size public datasets available, it has been necessary to create such datasets —to synthesize them. To do so, we have followed two approaches. For the first dataset, we have used four different ARIMA models over random sequences of zeros and ones, obtaining a four-class problem. For the second dataset, we have started from six real datasets, assigning a distinct label to the time series of each dataset. In addition, we have introduced a 10% of random noise to each time series from this last dataset to create enough time series. The experimental study results show that DFST achieves a significant improvement in classification accuracy performance and is linearly scalable in Big Data environments.

The journal contribution associated to this part is:

F. J. Baldán, J. M. Benítez. Distributed FastShapelet Transform: a Big Data time series classification algorithm. Information Sciences, 496, 451-463 (2019). DOI: `https://doi.org/10.1016/j.ins.2018.10.028`

6.2 Complexity Measures and Features for Times Series classification

In the time series classification field, we can find three well-differentiated approaches. First, the distance-based approach [BLB+17], which includes methods as simple to understand as K-Nearest-Neighbor (KNN), or complex ensembles composed of multiple classifiers from different domains, such as the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) [LTB18]. Second, there is the feature-based approach [Ful17] with two main sub approaches: on the one hand, we have the sub approach based on the selection of features from a purely experimental point of view, based on the results obtained on a large data set, such as the CAnonical Time-series CHaracteristics (catch22) proposal [LSK+19b]. On the other hand, we have a sub approach focused on selecting features manually, selecting sets of basic features [NAM01], such as mean, typical deviation, kurtosis, among others, or features extracted from the different known typical representations of the time series [BDHL12], power spectrum, autocorrelation function, among others. Finally, we have the approach based on Deep learning [FFW+19], being the Convolutional Neural Networks (CNNs) the most widely used architectures, mainly because of their robustness and relatively short training time. In most cases, the proposals focus on obtaining the best possible performance in terms of accuracy, leaving aside the interpretability of the results.

Nowadays, the use of machine learning models is expanding. These models are being applied in critical processes in which it is necessary to understand their operation and the decisions made [DBH18]. As mentioned above, in the field of time series classification, most proposals leave aside the interpretability of their models and focus on improving their performance. For these reasons, our proposal has been guided by the following concerns:

- First, to offer an alternative representation of the time series that allows the application of the large number of vector-based classification models available, which would otherwise not be applicable to time series problems.
- Second, the representation must improve the interpretability of the decisions made by interpretable models.
- Third, the selected set of measures must be sufficiently representative to reflect the characteristic behaviors of interest of the time series and offer competitive results concerning the main proposals of the state-of-the-art.

In this paper, we have analyzed the main characteristics of time series capable of expressing typical time series behaviors [HYND]. In addition, we have selected measures that evaluate the time series complexity, with the assumption that similar time series will have similar complexities. Based on these two ideas, we have proposed a set of 55 features, composed of complexity measures and time series features, capable of extracting the behaviors of interest from the processed time series. Then, each time series can be represented as a vector of fixed length, where each component corresponds to one of the selected features.

We have evaluated the performance of our proposal through extensive experimentation, using 112 datasets from the University of California, Riverside (UCR) Time Series Classification Archive [DKK+18], which is the main repository in the field of time series classification. We have compared our proposal's performance in terms of accuracy and interpretability. The performance results are competitive —not statistically distinguishable— concerning the state-of-the-art. In addition, this representation allows for improved information extraction from time series.

The paper associated to this part is:

F. J. Baldán, J. M. Benítez. Complexity Measures and Features for Times Series classification. Submitted.

6.3 Interpretable representation for multivariate time series classification

Due to the high interconnectivity of modern world, we are able to record large amounts of information over time, provided from different sources and related to the same process. The relationships between these different sources of information allow us to extract additional information and better understand the problem. In certain cases, if we analyze these sources independently we may not be able to understand the underlying functioning of the process. For this reason, multivariate time series analysis is a task that is gaining more and more attention nowadays.

A major problem with multivariate time series classification proposals is their low interpretability. They face a problem with high complexity and focus on the performance of the results obtained, leaving the interpretability of the models aside.

To face the above problem, we have designed a new multivariate time series representation based on our proposal for univariate time series. Our multivariate proposal concatenates in the same instance all the features of the components of the same time series and uses traditional classification algorithms to find the interrelationships between the different variables of the same multivariate time series.

A thorough empirical study of the merits of this representation has been designed and carried out. On the performance analysis of the results, we found that the classifiers built out of the time series represented with our proposal are competitive with the state-of-the-art methods to such an extent that no statistically significant differences can be found. On the interpretability face, much simpler and easy to cope with and understand models can be obtained. In addition, it is easier to extract relevant information, such as what variables within the time series components are more meaningful to the problem.

The journal contribution associated to this part is:

F. J. Baldán, J. M. Benítez. Multivariate times series classification through an interpretable representation. Information Sciences, 569, 596-614 (2021). DOI: `https://doi.org/10.1016/j.ins.2021.05.024`

6.4 SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments

The advance of technology, the development of new devices, and the reduction of costs are leading us to an increasingly interconnected world. Year after year, increasing amounts of information are generated, and with them, the need for computing capacity to process this information grows. Problems that until now could be treated conventionally have increased to such an extent their volume of data that new processing models are required to be able to treat it. Big Data and distributed processing models based on the MapReduce paradigm were born to address this problem.

If we analyze the tools available for time series analysis in Big Data environments, we can appreciate that the available options are very limited, especially if we compare them with those available for processing traditional vector-based problems. We can find some projects, most of them discontinued, that offer basic time series storage structures, simple statistical analysis tools, a simple regression model, or a single time series classification algorithm.

In this work, we have proposed a scalable and distributed method for Big Data environments, named SCMFTS, which transforms univariate and multivariate time series into vectors of well-known time series features. The proposed method lets us apply the already available traditional vector-based algorithms to time series problems in Big Data environments. Our proposal significantly extends the number of tools available to process time series in Big Data, allowing us to face real world time series classification problems that until now could not be addressed. The experimentation includes the biggest multivariate time series dataset available in the UCI Machine Learning Repository to evaluate the performance of our proposal and multiple synthetic datasets with high dimensions to evaluate its scalability.

The journal contribution associated to this part is:

F. J. Baldán, Daniel Peralta, Yvan Saeys, J. M. Benítez. SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments. International Journal of Computational Intelligence Systems, (2021). Accepted.

7 Discussion of Results

In this section, we will present and discuss the main results reached after the research carried out in this thesis.

7.1 Big Data time series classification algorithm

We proposed the first distributed algorithm for time series classification in Big Data environments. Our proposal combines the interpretability of shapelets with the performance achievable by the Shapelet Transform to offer competitive results. Moreover, it allows applying traditional classifiers already implemented in Big Data to time series classification problems. This opens a host of possibilities for big data time series classification.

In the experimental study, DFST has shown a linear scalability behavior regarding the number of time series on large datasets, which is an important requirement in Big Data environments. In addition, we have verified that our proposal provides better accuracy results than the original FastShapelet algorithm.

The results obtained paved the way to seek the application of new vector-based algorithms —which are more numerous than time-series specific algorithms— on our DFST since it has been demonstrated that this feature-based approach performs well.

To facilitate access to the algorithm, we have published the implementation at spark-packages repository[2], making it available for any interested user.

7.2 Complexity Measures and Features for Times Series classification

In this section, we present a new representation of univariate time series. Each time series is described as a vector of 55 characteristics composed of complexity measures and well-known time series features. Each component of the vector contains the value of a feature. The vector is the base for the new representation, and it allows applying traditional algorithms to time series problems, improving the interpretability of the models used.

But actually, the transcendence of the new representation is much wider since this new time series allows for a completely new view of time series analysis. It enables the reconsideration of every time series analysis task from a different perspective from those considered so far. Each time series is now depicted as a set of measures of meaningful features, some pieces of information readily descriptive for users. In addition, it makes possible and effective the application of classic machine learning tools.

In this paper, we focused on the straight application of the representation for time series classification. Its merits in terms of accuracy and interpretability have been analyzed through an experimental study. The empirical results obtained for our proposal are statistically indistinguishable from the results of the third-best state-of-the-art algorithm with 95% confidence. Moreover, in the case of datasets with at least 500 time series, we can see that our proposal obtains statistically indistinguishable results from those offered by the best state-of-the-art algorithm. This behavior is typical of feature-based methods since they require a *sufficient* amount of data to work properly. Also, we found that interpretable tree-based models are even more interpretable when using our proposed set of features instead of the original time series values. This is because the tree-nodes include well-known time series features rather than specific time series values.

As an additional result of our research, we have published the developed software under an open-source license[3], allowing its use and extension.

[2]Distributed FastShapelet Transform (DFST). `https://spark-packages.org/package/fjbaldan/DFST`
[3]Complexity Measures and Features for Times Series classification. `https://github.com/fjbaldan/CMFTS/`

7.3 Interpretable representation for multivariate time series classification

Here, we have presented a multivariate time series transformation based on well-known time series features. This process allows increasing the interpretability of the problems. Our proposal transforms multivariate time series into traditional feature vectors, obtaining in the same instance all the characteristics of all the variables of a multivariate time series.

The results obtained show that our proposal achieves results statistically indistinguishable from the main state-of-the-art algorithms. So our proposal makes it possible to add vector-based algorithms, not specific to time series, to the repertoire of tools available to face time series classification problems with guarantees. Also, we have verified how, for some cases, our proposal achieves competitive and directly interpretable results using the extracted features and a simple decision tree. In addition, we can see how some features are useful to explain the behavior of the time series and how to identify which of the variables of a multivariate time series contain higher amounts of information of interest for the problem. To sum up, this work explores and extends the results achieved with our feature-based representation for univariate time series to multivariate time series.

A particularly relevant result of our research is the publication of our proposal software as open-source[4], making it available for any user.

7.4 SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments

In this work, we have introduced a scalable and distributed transformation for univariate and multivariate time series based on well-known time series features for Big Data environments (SCMFTS). In other words, this work further extends the application of the new representation for univariate and multivariate time series classification in the Big Data scenario. We have verified that this approach leads to scalable classification methods. This proposal significantly increases the rather small toolset for Big Data time series analysis.

The results obtained from the experimental study indicate that SCMFTS has significantly improved the results obtained by the state-of-the-art on the WESAD dataset. Moreover, our proposal has shown a fully scalable behavior in Big Data environments, being close to the theoretical limit expressed by Amdahl's law [HM08]. The obtained results demonstrate that SCMFTS can address real problems in Big Data environments with solvency in terms of results and scalable performance.

Finally, following an endeavor for reproducible and extendable Science, we have published the software of SCMFTS as open-source[5], allowing easy access and use of our work.

[4]Complexity Measures and Features for Multivariate Times Series classification. `https://github.com/fjbaldan/CMFMTS/`

[5]SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments. `https://github.com/fjbaldan/SCMFTS/`

8 Concluding Remarks

In this thesis, we have addressed several problems with the same purpose: time series analysis in Big Data environments.

The first objective was to deal with time series classification problems in Big Data environments. To face this problem, we have proposed a Big Data time series classification algorithm, Distributed FastShapelet Transform (DFST). DFST combines the low computational complexity of the FastShapelet algorithm with the performance of the results offered by the Shapelet Transform. The main proposal of the state-of-the-art HIVE-COTE is an ensemble composed of classifiers from different domains across five large modules and has quadratic complexity concerning the number of time series, being impossible to apply it in Big Data environments. Our proposal enables the application of any traditional classification algorithm already implemented in Big Data on time series classification problems. DFST has shown excellent performance results and a linear complexity with the number of processed time series, which allows its successful application in Big Data environments.

To address the second objective, to increase the interpretability of time series analysis, we have proposed a new representation of time series based on descriptive features of their behavior (CMFTS). The aim is to be able to explain the behavior of the time series through these features. CMFTS can be applied to supervised and unsupervised problems. In this case, we focus on classification problems since the comparison between different models is simpler and more direct. CMFTS allows obtaining competitive results concerning the main state-of-the-art algorithms, especially when addressing datasets with a large number of time series. Moreover, it is able to improve the interpretability of the built models, as it allows explaining the decisions made through features well known in the field of time series.

The third objective was to provide an alternative representation of multivariate time series. We have developed a proposal based on an extension of the representation previously defined for univariate time series. We calculate the features independently for each variable of each multivariate time series, and we include them in the same instance. The inclusion of all the features of all the variables of a multivariate time series in the same instance allows the applied algorithms to extract relationships between variables and time series that contain relevant information about the problem. The relationships found between the different variables that compose the same time series and the different multivariate time series explain interesting underlying behaviors. Besides improving the interpretability of the problem, our proposal offers statistically indistinguishable results from those provided by the best state-of-the-art algorithms, which optimize their parameters for each dataset.

Our last objective, facing real-world univariate and multivariate time series classification problems in Big Data environments, has been addressed by proposing a scalable and distributed transformation for univariate and multivariate time series based on well-known time series features for Big Data environments (SCMFTS). Our proposal, which has been implemented using the MapReduce Paradigm, has shown great performance in Big Data environments, being able to significantly improve the results obtained by the state-of-the-art on the largest multivariate time series dataset available in the UCI repository. It has also demonstrated a fully scalable behavior, close to the theoretical limit expressed by Amdahl's law, which indicates that its correct application in real-world Big Data environments is guaranteed.

As a final conclusion, we can state that the objectives set have been achieved.

Conclusiones

En esta tesis hemos afrontado diferentes problemas orientados a un mismo objetivo: el analisis de series temporales en entornos Big Data.

El primer objetivo fue afrontar problemas de clasificación de series temporales en entornos Big Data. Para hacer frente a este problema, hemos propuesto un algoritmo Big Data de clasificación de series temporales, Distributed FastShapelet Transform (DFST). DFST combina la baja complejidad computacional del algoritmo FastShapelet con el rendimiento en resultados que ofrece la ShapeletTransform. La propuesta principal del estado del arte HIVE-COTE es un complejo ensemble compuesto por clasificadores de dominios diferentes a lo largo de cinco grandes módulos y tiene una complejidad cuadrática respecto al número de series temporales, siendo imposible aplicarlo en entornos Big Data. Nuestra propuesta permite aplicar cualquier algoritmo de clasificación tradicional ya implementado en Big Data en problemas de clasificación de series temporales. DFST ha mostrado unos resultados de rendimiento excelentes y una complejidad lineal con el número de series temporales procesadas, lo que permite su correcta aplicación en entornos Big Data.

Para abordar el segundo objetivo, incrementar la interpretabilidad del análisis de las series temporales, hemos propuesto una representación nueva de series temporales basada en rasgos descriptivos de su comportamiento (CMFTS). El objetivo es poder explicar el comportamiento de las series temporales a través de estas características. CMFTS puede aplicarse a problemas supervisados y no supervisados. En este caso, nos centramos en los problemas de clasificación, ya que la comparación entre diferentes modelos es más sencilla y directa. CMFTS permite obtener resultados competitivos respecto a los principales algoritmos del estado del arte, especialmente al abordar conjuntos de datos con un gran número de series temporales. Además, es capaz de mejorar la interpretabilidad de los modelos construidos, ya que permite explicar las decisiones tomadas a través de características bien conocidas en el ámbito de las series temporales.

El tercer objetivo fue proporcionar una representación alternativa de las series temporales multivariantes. Hemos desarrollado una propuesta fruto de una extensión de la representación previamente definida para series temporales univariantes. Calculamos las características de forma independiente para cada variable de cada serie temporal multivariable, y las incluimos en la misma instancia. La inclusión de todas las características de todas las variables de una serie temporal multivariable en la misma instancia permite a los algoritmos aplicados extraer relaciones entre variables y series temporales que contienen información relevante sobre el problema. Las relaciones encontradas entre las diferentes variables que componen una misma serie temporal y las diferentes series temporales multivariantes explican interesantes comportamientos subyacentes. Además de mejorar la interpretabilidad del problema, nuestra propuesta ofrece resultados estadísticamente indistinguibles de los proporcionados por los mejores algoritmos del estado del arte, que optimizan sus parámetros para cada conjunto de datos.

Nuestro último objetivo, enfrentarnos a problemas reales de clasificación de series temporales univariantes y multivariantes en entornos Big Data, se ha abordado proponiendo una transformación escalable y distribuida para series temporales univariantes y multivariantes basada en características conocidas de series temporales para entornos Big Data (SCMFTS). Nuestra propuesta, que ha sido implementada usando en el paradigma MapReduce, ha mostrado un gran rendimiento en entornos Big Data, siendo capaz de mejorar significativamente los resultados obtenidos por el estado del arte en el mayor conjunto de datos de series temporales multivariantes disponible en el repositorio de la UCI. También ha demostrado un comportamiento totalmente escalable, cercano al límite teórico expresado por la ley de Amdahl, lo que indica que su correcta aplicación en entornos Big Data del mundo real está garantizada.

Como conclusión final, podemos aseverar que los objetivos marcados han sido alcanzados.

9 Future Work

From the conclusions drawn in this thesis, a number of interesting new lines of research arise. The aim is to improve the proposed models and tools and to apply them to new problems.

- **Analysis of new time series features**: the feature-based approach is becoming more and more important. We can find proposals that use very large sets of features and use reduction techniques to select a specific subset for each problem [ANGAB$^+$21], use of the information contained in segmented time-series with the objective of obtaining comprehensive handcrafted features related to the problem objective [IJF$^+$17], or even the use of hypothesis testing to identify the meaningful features [CBNKL18]. Our main aim is to extend the set of basic features, including additional features potentially more relevant. For sure, not every feature will be relevant for every time series area, but possibly we can identify groups of features supportive for specific areas. We intend to further explore this idea.

- **Types of time series classification problems in Big Data that are not covered for the currently available methods that could arise**: the small number of proposals for time series classification algorithms in Big Data shows an evident deficiency in this field. Apart from the new methods proposed in this thesis, it is practically impossible to find proposals developed with Big Data technologies (Spark, Hadoop). The proposals found are focused on optimizing existing procedures. They use traditional computers not included in Big Data environments [RCM$^+$13]. Several interesting cases can be spotted: for example, proposals based on dictionaries to face problems in which the number of times a pattern appears in a time series is determinant to classify it correctly, where shapelets are not useful, or ensembles that can combine the results of different classifiers to obtain a better final result.

- **Improvements for the feature-based method proposed for multivariate time series**: so far, the features used are independent for each variable of each time series. Adding features that combine information from multiple variables in a single feature or including mechanisms that allow extracting the features of sub-sequences of interest from the processed time series are improvements that require a deep study. Although including these options could significantly improve the obtained results, it would need to be carefully done since it is necessary to preserve the scalability of the proposal in Big Data environments.

- **Applying our proposals in other fields and new problems**: although we have focused on classification problems, the proposed features can be helpful in unsupervised problems where the additional information provided has a great value or the search for new large-scale problems where our proposals can provide competitive results.

Chapter II

Publications

1 Distributed FastShapelet Transform: a Big Data time series classification algorithm

- F. J. Baldán, J. M. Benítez. Distributed FastShapelet Transform: a Big Data time series classification algorithm. Information Sciences, 496, 451-463 (2019).
  - Status: **Published**.
  - Impact Factor (JCR 2019): **5.910**
  - Subject Category: **Computer Science, Artificial Intelligence**
  - Rank: **9/156**
  - Quartile: **Q1**

Distributed FastShapelet Transform: a Big Data time series classification algorithm

Francisco J. Baldán[*]
Department of Computer Science
and Artificial Intelligence
University of Granada, DICITS, iMUDS, DaSCI
Granada, Spain, 18071
`fjbaldan@decsai.ugr.es`

José M. Benítez Sánchez
Department of Computer Science
and Artificial Intelligence
University of Granada, DICITS, iMUDS, DaSCI
Granada, Spain, 18071
`J.M.Benitez@decsai.ugr.es`

Abstract

The classification of time series is a central problem in a wide range of disciplines. In this field, the state-of-the-art algorithm is COTE (Collective of Transformation-Based Ensembles) which is a combination of classifiers of different domains: time, autocorrelation, power spectrum and shapelets. The weakest point of this approach is its high computational burden which prevents its use in massive data environments. Shapelet Transform is one of the multiple algorithms that compose this ensemble. It has been shown to achieve a good performance over many reference datasets. Nevertheless, its computational complexity is also too high to be used in massive data environments. On the other hand, Big Data has emerged as an approach to manage massive datasets, which also applies to time series. We propose an algorithm for time series classification in a Big Data environment, DFST. It is based on a combination of the FastShapelet and Shapelet Transform ideas and it is the first completely scalable algorithm for time series classification. We have shown that our proposal scales linearly with the number of time series in dataset. In addition, the classification accuracy is equal to or higher than that of comparable sequential algorithms.

Keywords Time Series · Big Data · Classification · Shapelet

1 Introduction

Nowadays, we are in the Big Data era. Huge loads of data are created, stored and processed. Their features in many dimensions (volume, velocity, variety, complexity, etc) exceed the computation capabilities of current computers. A new approach to cope with them is necessary.

Advances in computing technologies allow to capture, store and process large amounts of data from varied events and processes: changes in climate, traffic evolution, vital signs, etc. Many of these massive datasets are time series, e.g. the dataset "CCAFS-Climate Data" [7] [20], with up to 6TB of information on temperature, solar radiation, etc, or "Federal Reserve Economic Data - Fred" [16] [21], with up to 20,059 U.S. economic time series.

Time series processing for large datasets is a computationally expensive and complex process [11], specially in the field of time series classification. The leading proposal of the current state-of-the-art is COTE [2] (Collective of Transformation-Based Ensembles). This algorithm is based on the idea that an ensemble made up of time series classifiers from different domains (time, autocorrelation, power spectrum and shapelets) can offer better results than each classifier individually. Through a comprehensive study, a recent work [1] has demonstrated that for a given problem, the simplest algorithms of a domain which adapt to the problem can offer better results than the most complex algorithms of other domains. We can also observe how the different types of time series classification algorithms are described and classified in this study according to the type of processing they perform on the time series.

[*]Corresponding author.

The shapelets and the Shapelet Transform (ST) algorithm [13] have gained prominence in recent years due to the good results obtained in time series classification problems. Specifically, problems based on the search for independent phase patterns. Shapelet primitive [22] was developed in order to obtain interpretable results in the classification of time series, while achieving the same performance in accuracy than the state-of-the-art algorithms. ST is the data transformation method that is used to convert the raw time series data using the shapelets, allowing the use of any traditional classification algorithm on the new transformed data. Shapelet Transform algorithm obtains the best classification results in the field of shapelets, but it has sacrificed some of the interpretability of the results. In addition, it has the highest computational complexity, $O(n^2m^4)$, of the time series classification algorithms based on shapelets. So far, proposals have been made in two directions: improving accuracy and reducing computational complexity. An example of proposals that reduce the computational complexity of the shapelets search is the FastShapelet (FS) algorithm, which is a heuristic algorithm search and classification of time series, which uses a traditional classification tree and that obtains a computational complexity of $O(nm^2)$. Nevertheless, this approach can not be applied to large time series datasets or Big Data problems due to its complexity. A reasonable improvement can be achieved through a Big Data processing approach.

In this paper, we propose a distributed and scalable time series classification algorithm based on the MapReduce paradigm for Big Data environments named Distributed FastShapelet Transform (DFST). DFST is a hybrid algorithm that uses a re-designed and distributed version of the shapelets search mechanism, proposed by the FS algorithm [19]. DFST includes design elements to render it of a reduced computational complexity enabling the application of the ST in Big Data environments and making it the only completely scalable procedure currently available: with the provision of adequate hardware resources it can process a time series dataset of *any* size. Thus DFST enables the processing of time series datasets that cannot be handled by a sequential approach. The major changes in DSFT with respect to FS are focused on four points: the control of exponential growth in the generation of random projections, the simultaneous selection of several shapelets at each thread, the use of a localized computation of information gain and, the outcome, which allows the use of other classification algorithms, removing the tie to classification trees. As a result of all of this, DFST accuracy results are better than those obtained by FS. The practical incarnation of the algorithm has been developed in Apache Spark and it is available in SparkPackages[2].

The rest of this proposal is structured as follows. Section 2 includes related papers and background on shapelets. In Section 3 we explain our distributed proposal. In Section 4 we show the experimental study carried out to test the effectiveness of our proposal. The conclusions of this paper are presented in Section 5.

2 Related and background work

Our work aims to address problems of classifying time series on large datasets. In Section 2.1 we present the shapelet primitive, which provides interpretable results on time series classification problems, the FS algorithm, that is a heuristic shapelet search algorithm with reduced computational complexity, and the ST, which is a transformation that obtains characteristics from the time series. In Section 2.2 we introduce the MapReduce Model used in Big Data environments, on which we have implemented our proposal.

2.1 Shapelet primitive, FastShapelet algorithm and Shapelet Transform

This section is structured as follows. In Section 2.1.1 we explain the primitive shapelet. In Section 2.1.2 we show the concept and operation of the Fast Shapelet algorithm. In Section 2.1.3 we explain the idea of the Shapelet Transform.

2.1.1 Shapelet primitive

The shapelet is a primitive [22] used in time series classification problems. It is composed by a subsequence of the time series from which it comes and a threshold distance. The shapelets are used to create a classification tree, where each internal node is composed by one shapelet. Each internal node separates the training instances depending on whether or not they contain the internal node's shapelet. The objective is to obtain a classification tree in which the leaf nodes have instances of a single class. If the shapelet is included in a time series, that time series is classified as belonging to the class of the time series from which this shapelet comes from. We compute the distance between the time series and the shapelet to know if a shapelet is included in a time series. If that distance is less than the threshold distance we consider that the time series contains the processed shapelet. We compute the distance between a time series and a shapelet as the minimum distance between the shapelet's subsequence and all possible subsequences of equal length of the time series.

[2]Distributed FastShapelet Transform (DFST). `https://spark-packages.org/package/fjbaldan/DFST`

2.1.2 FastShapelet algorithm

The FastShapelet algorithm (FS) [19] was proposed to improve the efficiency of the original extraction algorithm. This algorithm has a complexity of $O(nm^2)$, where n is the number of items or time series to process and m the length of the longest time series. The original shapelet discovery algorithm has a complexity of $O(nm^3)$. FS is a heuristic algorithm that faces the discovery of shapelet by applying a change of representation. For this purpose, FS uses Symbolic Aggregate approXimation (SAX) [12] converting the original real values to discrete values with smaller dimensions. All the extracted sequences are transformed into strings of length 16 with 4 discrete levels per value. This mechanism produces multiple SAX words for each time series. The creation of SAX words has the disadvantage that two subsequences with small differences can generate two different SAX words. Random Masking [6] is used to solve this problem. In this way very similar sequences with different SAX words randomly mask some of their values. After multiple iterations they can be identified as similar even if their SAX words are different. After the generation of random projections a frequency count histogram is built for each class. A score is calculated for each SAX word based on its ability to discriminate between classes by processing the frequency count histogram. The best k SAX words are selected and the actual values of their respective shapelets are retrieved. These shapelets are evaluated according to the following parameters and in this order: the gain of information calculated, the separation achieved between instances of different classes and the number of correctly separated training instances. Once the input dataset has been fully processed, the algorithm returns a decision tree. On this tree, each internal node contains the shapelet and the threshold distance that will be used to classify the new input data.

2.1.3 Shapelet Transform

The Shapelet Transform (ST) [13] does not use the extracted shapelets to classify new time series, but as input characteristics for a classifier. This allows using almost any classification algorithm.

At present, there are different ways to extract the shapelets [22] [15] [19] [3] but once extracted, the operation of the algorithm is the same. For a dataset of n time series and m extracted shapelets, the minimum distance between each time series and each shapelet is calculated, obtaining a new dataset with n instances and m characteristics. This new dataset is the training set of any traditional classification algorithm that you decide to apply. This transformation has to be applied to the test set as well.

It has been shown that this approach improves the results obtained by the classification algorithms based on shapelets [1] in most cases, in addition to maintaining the original interpretability of the Shapelets.

2.2 MapReduce Model

The exponential growth of data generated globally is popularizing the use of Big Data technologies. Most of these technologies are based on the MapReduce framework designed by Google in 2003 [8]. This framework allows creating clusters of common machines that can process large amounts of data. The user does not have to worry about tasks like partitioning the input data, handling machine failures or the inter-machine communication, among others. The MapReduce paradigm is based on two phases:

- The Map phase applies a transformation on each key-value pair of the original dataset locally.
- The Reduce phase unifies the results of the Map phase by means of associative operations and returns a result.

Currently, Apache Hadoop [9] is the most popular framework based on the MapReduce paradigm. This framework suffer from some weaknesses:

- Low memory usage, each executed phase must be written to disk.
- Iterative processes are difficult to implement and have a poor performance.

For these reasons, new alternatives such as Apache Spark [10] have been developed. Apache Spark uses in-memory workloads with memory-intensive usage, increasing the speed of computation by several orders of magnitude. It also allows the implementation of iterative processes in a simple and substantially more efficient way than its predecessors.

Apache Spark [24] is an engine for large-scale data processing. It was developed with the aim of facing tasks that focus on applying parallel operations on a input dataset that is constantly reused. One of its main features is the increase in running speed compared to other options. Spark can run the same program than Apache Hadoop up to 100 times faster. Its distributed processing architecture allows to increase the computing capacities of a cluster by adding new computers transparently to the user. Resilient Distributed Datasets (RDDs) [23] are the data structure on which Apache Spark distributed operations are based. These operations are computed over the local data on each partition. There are two types of operations: Transformations and Actions. The Transformations are not executed until an action is performed. These transformations apply a function on each RDD instance and return a new RDD. The Actions execute all transformations applied on an RDD returning a result. This result depends on the action applied. The RDDs are stored in memory. They are immutable and keep a checkpoint of all the transformations applied to them. This checkpoint is known as "lineage" and it allows recovering any partition in case of error.

3 Distributed FastShapelet Transform proposal

In this section, we present our proposal DFST, for scalable time series classification. Our proposal is based on the MapReduce paradigm, which allows it to be applied to massive time series datasets. In Section 3.1 we explain the computational complexity of our proposal.

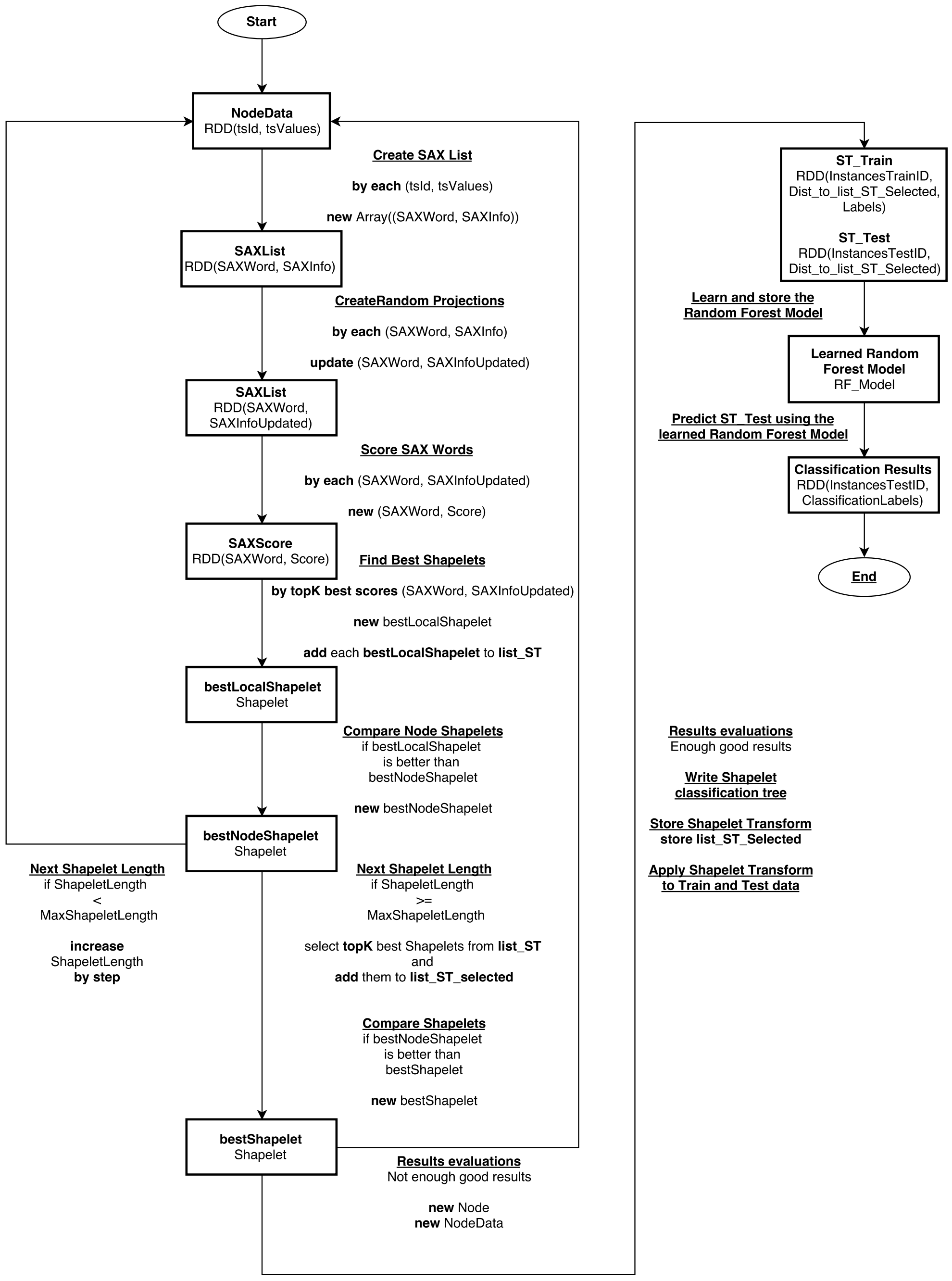
The proposal is expressed in terms of the following Spark primitives:

- *map*: A transformation that applies a function to all elements of the input RDD, returning a new RDD.
- *mapPartitions*: A transformation that applies a function to every partition of the input RDD. This transformation returns a new RDD.
- *reduce*: A transformation that merges the elements of the input RDD using an associative and commutative binary operator.
- *filter*: A transformation that returns a new RDD containing only the elements that satisfy a predicate.
- *sortBy*: A transformation that returns the input RDD sorted by the given key function.
- *count*: An action that returns the number of elements in the input RDD.
- *take*: An action that returns the first `num` elements of the input RDD. *num* is the number of elements to return.
- *lookup*: An action that returns the list of values in the input RDD with the specified key.

The main objective of our proposal is to create an scalable algorithm that can classify time series in Big Data environments. We set as a requirement that the relationship between the increase in running time and the amount of data to be processed must be linear. The second objective is that our proposal must be able to extract and use interpretable features such as shapelets in the best possible way. For this reason, our proposal uses the Shapelet Transform (ST). In addition, our proposal must maintain an accuracy close to the one obtained by similar sequential algorithms. Based on those requirements we have called our proposal Distributed FastShapelet Transform (DFST).

DFST is based on the heuristic shapelet search made in the FS algorithm [18], which is the shapelet search algorithm with the lowest computational complexity of the shapelet search algorithms namely $O(nm^2)$.

Figure 1: Distributed FastShapelet Transform (DFST) Schema


Start
NodeData RDD(tsId, tsValues)
Create SAX List
by each (tsId, tsValues)
new Array((SAXWord, SAXInfo))
SAXList RDD(SAXWord, SAXInfo)
CreateRandom Projections
by each (SAXWord, SAXInfo)
update (SAXWord, SAXInfoUpdated)
SAXList RDD(SAXWord, SAXInfoUpdated)
Score SAX Words
by each (SAXWord, SAXInfoUpdated)
new (SAXWord, Score)
SAXScore RDD(SAXWord, Score)
Find Best Shapelets
by topK best scores (SAXWord, SAXInfoUpdated)
new bestLocalShapelet
add each bestLocalShapelet to list_ST
bestLocalShapelet Shapelet
Compare Node Shapelets
if bestLocalShapelet is better than bestNodeShapelet
new bestNodeShapelet
bestNodeShapelet Shapelet
Next Shapelet Length
if ShapeletLength < MaxShapeletLength
increase ShapeletLength by step
Next Shapelet Length
if ShapeletLength >= MaxShapeletLength
select topK best Shapelets from list_ST and add them to list_ST_selected
Compare Shapelets
if bestNodeShapelet is better than bestShapelet
new bestShapelet
bestShapelet Shapelet
Results evaluations
Not enough good results
new Node
new NodeData
Results evaluations
Enough good results
Write Shapelet classification tree
Store Shapelet Transform
store list_ST_Selected
Apply Shapelet Transform to Train and Test data
ST_Train RDD(InstancesTrainID, Dist_to_list_ST_Selected, Labels)
ST_Test RDD(InstancesTestID, Dist_to_list_ST_Selected)
Learn and store the Random Forest Model
Learned Random Forest Model RF_Model
Predict ST_Test using the learned Random Forest Model
Classification Results RDD(InstancesTestID, ClassificationLabels)
End

The operation of DFTS is shown in Figure 1. The original ideas of the SAX word creation, the generation of random projections and the score of the SAX words of the FS algorithm have been used in the DFST algorithm, but they have been redesigned to work in a distributed fashion. The main issues in which DFST differs from FS are detailed in the following paragraphs.

A first step in the procedure is the application of a SAX transformation for a simplified representation of the time series. Next step is a generation of random projections of the computed SAX words. This process is prone to exponential growth. So when applied to larger datasets the overall set of word size must be controlled. DFST reduces the length of SAX words as a function of the dataset partition that is processed by each thread.

Another effective improvement is that instead of a single shapelet, DFST selects the $topK$ shapelets with the highest scores. Then, for each selected shapelet and locally at each thread, it computes the information gain, the number of misclassified time series, and the gap between the time series that have this shape and those that do not. The search for the best shapelets has been modified so that multiple shapelets are obtained, in each node of the tree, and saved for later use in the ST.

DFST applies the ST to the training and test sets, calculating the distance of each time series to every selected shapelet. This vector of distances is used as a representation of the time series. Thus, at this stage the output is not a classification result but a representation of the time series in terms of the most relevant shape features discovered and with the same length for all. That is, time series are represented in a new way that allows for an easy processing with conventional data mining and machine learning algorithms. In particular, almost any classification algorithm can be used now, removing the tie to the classification tree in the FS algorithm.

The last step in DFST is to learn a model with the transformed training data, ST_Train. It will be used to predict the transformed test data, ST_Test, obtaining the final classification results. So far, we have used Random Forest [5], but this is no restriction of course.

An interesting analysis of the proposal is considering what parts of DFST a sequential and what parts are parallel. In addition, the exchange of data among threads also have a clear impact on the running time. DFST has a first sequential part of reading and delivering the input data. The generation of SAX words and the creation of random projections are performed in a fully parallel fashion, by applying certain transformations independently to each input time series. The score of each shapelet candidate, and its position based on this score, require the overall exchange of information among threads as it is necessary to count the presence or not of each shapelet candidate in all the time series of the dataset and perform this sorting. There is another stage of data exchange in the evaluation of the $topK$ shapelets candidates as it needs to process and collect the calculations of information gain for each of them.

DFST is depicted in Algorithm 1. In Algorithm 2 we present the core procedure of our proposal. In this process the best set of shapelets is selected. We must differentiate between distributed and not distributed variables: as general nomenclature, we represent the distributed variables with their first letter in capital.

In Algorithm 1, all original instances are introduced in node 1 (line 3). The number of instances not correctly classified are initialized with the number of original instances (line 4). The algorithm processes a node as long as the number of instances not correctly classified is greater than the threshold (lines 5-34). For each node, the algorithm selects the instances included in that node (line 6). Then it starts to process subsequences of time series from the minimum to the maximum introduced lengths (line 9-26). This process has 4 principal steps:

First, the algorithm obtains a list of all possible SAX words in the current node data (lines 10-17). We use a map transformation over the node in order to obtain a HashMap that contains all possible SAX word for each time series (lines 10-13). Each SAX word has a usax item associated with information about that SAX word. Then, we use a $reduce$ function (lines 14-17) for combining identical SAX words. The usax item of the first SAX word is updated with extra information about the second SAX word. Secondly, the algorithm computes R random projections for each SAX word and then it counts matches among projections of different SAX words (line 18). The counted values are saved in the match counter of the usax item of the corresponding SAX word. Third, the algorithm computes a score for each SAX word based on their match counter (lines 19-22). Fourth, our proposal selects the $topK$ SAX words that have the higher score and it computes the best shapelet among them (lines 23-24). If the computed shapelet improves the last best case, this shapelet is selected as the current best case (line 25). We increase the subsequence length by the desired $step$ (line 26).

Once the current node has been processed, the algorithm analyzes the results and prepares the next iteration. With the best shapelet of the current node, each time series is assigned to one of the possible next nodes (line 28). DFST obtains the time series incorrectly classified and the stopping threshold of the algorithm (line 29) for the next nodes. The $topK$ shapelets with the highest gain are selected and added to the final list of shapelets (line 30). This list contains the shapelets to be used in the ST. Finally, we increase the node indicator (line 31). After all the relevant shapelets have been extracted, ST is performed on the training dataset. The minimum distance of each time series to each of the selected shapelets is calculated (line 33). Transformed training data is used as input for the classification algorithm, e.g. RandomForest (line 34). Finally, DFST returns the learning model obtained on the training data set and the list of shapelets used in the ST (line 35).

In Algorithm 2, our proposal searchs for the best shapelet of the current SAX word list. We take the $topK$ best scored SAX words to process them (line 1). Each shapelet is processed independently (lines 3-28). The algorithm retrieves information about the original subsequence of the SAX word processed (line 4). It uses this information to obtain the original values of this subsequence (line 5). We use $mapPartitions$ transformation to evaluate independently each shapelet over each partition of the data (lines 6-22). Our proposal computes the local frequency of each class in the

Algorithm 1 DFST Algorithm

```
    Input:
        $OrgData$: RDD with (TsId, LabeledPoint(features, label))
        $minLen$: minimum length of shapelet processed
        $maxLen$: maximum length of shapelet processed
        $step$: the increment of length of shapelet processed
        $topK$: number of best shapelets evaluated by iteration
        $R$: number of random projections computed
    Output:
        $RF\_Model$: RF learned model
        $list\_ST\_selected$: list of shapelet selected for the ST
 1: Node ← 1
 2: bestSh, threshold, list_ST_selected ← 0
 3: NodeTsList[Node] ← getAllTsId(OrgData)
 4: incorrectlyClassifiedData ← count(NodeTsList[Node])
 5: while incorrectlyClassifiedData > threshold do
 6:     NodeData ← filter(OrgData, NodeTsList[Node]==getTsId(OrgData))
 7:     subSeqLen ← minLen
 8:     list_ST ← 0
 9:     while subSeqLen < maxLen do
10:         SaxMapWords ←
11:             map ts ∈ NodeData
12:                 HashMap[saxWord, usax] ← createSaxList(ts, subSeqLen)
13:             end map
14:         SaxMapWordsReduced ←
15:             reduce (SaxListX, SaxListY) ∈ SaxMapWordsReduced
16:                 combineSaxList(SaxListX, SaxListY)
17:             end reduce
18:         RPWords ← createRP(SaxMapWordsReduced, R, subSeqLen)
19:         ScoreList ←
20:             map (word, usax) ∈ RPWords
21:                 (word, calScore(usax, labels(NodeData)))
22:             end map
23:         (sh, list_ST) ← findBestShapelet(topK, subSeqLen, ScoreList,
24:                             NodeData, list_ST)
25:         if (sh > bestSh) then bestSh ← sh end if
26:         subSeqLen ← subSeqLen + step
27:     end while
28:     NodeTsList ← setNextNodes(NodeTsList, Node, OrgData, bestSh)
29:     (incorrectlyClassifiedData, threshold) ← evaluate(Node, OrgData, bestSh)
30:     list_ST_selected ← addBestShapelets(topK, list_ST, list_ST_selected)
31:     Node ← Node + 1
32: end while
33: ST_Train ← calcST(Org_Data, list_ST_selected)
34: RF_Model ← trainRF(ST_Train)
35: return (RF_Model, list_ST_selected)
```

partition (line 9). Then it computes the entropy of local data given the number of instances of the partition and the class frequencies (lines 10-11). We compute the distance between the subsequence selected and each time series in the partition (line 12). These results are sorted in ascending order (line 13) and then it searches on these for the best shapelet (lines 14-21). For this, we compute the information gain, the number of instances in a different position of the correct and the inter-class gap (lines 19-21). Then it takes as the best shapelet of the partition the case that has the maximum gain, the minimum number of differences and the maximum inter-class gap (line 19), respectively. As $DistributedShapeletCalculated$ RDD contains the best shapelet of each partition, we must take the best of these cases following the previous criteria. We take the first of those cases (line 23). Our proposal saves the selected best shapelet for the ST afterwards (line 24). We compare this shapelet with the best case at the moment and we take the best one (line 25). Therefore, the algorithm returns the best shapelet of the $topK$ processed and the list with the node's shapelets for the ST (line 28).

DFST uses the shapelet extraction mechanism proposed in the FS algorithm. This proposal selects the $topK$ shapelets extracted in each internal node as shapelets to be applied in the ST. For example, for a tree with 4 internal nodes and a $topK$ value of 10, our proposal would extract 40 shapelets.

Algorithm 2 $findBestShapelet$

```
Input:
    $topK$: number of better SAX words to consider
    $subSeqLen$: length of shapelet processed
    $ScoreList$: RDD with the scores for word
    $NodeData$: RDD that contains the data of the current node
    $list\_ST$: list with the node's shapelets for the ST
Output:
    $bestSh$: best Shapelet found
    $list\_ST$: updated list with the node's shapelets for the ST
 1: ScoreListLocal $\leftarrow$ take(sortBy(ScoreList, "score", "decrescent"), topK)
 2: k, bestSh $\leftarrow$ 0
 3: while $k < topK$ do
 4:     tsCandInfo $\leftarrow$ ScoreListLocal(k)
 5:     tsCand $\leftarrow$ lookup(Data, tsIndex(tsCandInfo))
 6:     DistributedShapeletCalculated $\leftarrow$
 7:         mapPartitions DataPartition $\in$ NodeData
 8:             localBestSh $\leftarrow$ 0
 9:             localClassFreq $\leftarrow$ countClassLabels(labels(DataPartition))
10:             localClassEntropy $\leftarrow$
11:                 entropyArray(localClassFreq, size(DataPartition))
12:             distTs $\leftarrow$ calcNNDist(DataPartition)
13:             orderedDistTs $\leftarrow$ sortBy(distTs, "NNDist", "ascending")
14:             i $\leftarrow$ 0
15:             while i < (size(orderedDistTs) - 1)
16:                 sh $\leftarrow$ calcShInfo(tsCandInfo, orderedDistTs(i),
17:                         orderedDistTs(i+1), localClassFreq,
18:                             localClassEntropy)
19:                 if (sh > localBestSh) then localBestSh $\leftarrow$ sh end if
20:             end while
21:             localBestSh
22:         end mapPartitions
23:     sh $\leftarrow$ getBestShapelet(DistributedShapeletCalculated)
24:     list_ST $\leftarrow$ addShapelet(sh, list_ST)
25:     if (sh > bestSh) then bestSh $\leftarrow$ sh end if
26:     k $\leftarrow$ k+1
27: end while
28: return (bestSh, list_ST)
```

In DFST, for a dataset with n time series of length m from which it has been extracted s shapelets, once already applied the ST we will obtain a dataset with n instances of length s. This approach has proven to be the most recommended within the field of time series classification by shapelets. The main advantage of this transformation is the possibility of applying any automatic learning algorithms, e.g. decision trees, to classification problems of time series. In our proposal, we have chosen to use the Random Forest algorithm [5] included in the MLlib of Spark with default parameters and 1000 trees.

3.1 Computational Complexity

COTE algorithm is the most popular and widely used algorithm for time series classification. This approach achieves excellent results in most cases. However, COTE is a computationally expensive technique. The computational complexity of COTE is set by the classifier with greater computational complexity that forms part of this ensemble. The algorithm with highest computational complexity included in COTE is the ST, with a computational complexity of $O(n^2m^4)$. The computational complexity limits the number of time series that this approach is capable of processing in conventional computers with limited resources. This high computational complexity order prevents its use in Big Data environments.

The DFST learning computational complexity in time is defined by the sum of the complexity of the FS based search algorithm, the shapelet transformation applied to the training set and the computational complexity of learning a Random Forest model. The computational complexity of the shapelet search algorithm used, which is based on FS, is $O(nm^2)$, being n the number of time series and m the length of time series. The transformed shapelet applied to the training set calculates the distance of each shapelet to each time series. This distance has a computational complexity $O(timeSeries_{length} - shapelet_{length})$, since the minimum distance between the shapelet is calculated to all the sequences of the time series of equal length to that of the shapelet. To simplify, we replace $(timeSeries_{length} - shapelet_{length})$ with p in the following computational complexity equations. This calculation is repeated for each time series of the training set, n, and as many times as shapelets have been extracted, s. For this reason, the computational complexity of the ST applied is $O(pns)$. The computational complexity of the Random Forest learning has been theoretically demonstrated [14] as $O(tk\tilde{n}\log\tilde{n})$, being t the number of randomized trees, k the number of variables randomly included at each node and n the number of samples of the training partition. $\tilde{n} = 0.632n$ due to the 63.2% of unique samples, on average, [4] extracted by bootstrap. Finally, the DFST learning computational complexity in time is: $O(nm^2) + O(pns) + O(tk\tilde{n}\log\tilde{n})$.

The DFST prediction time complexity is defined by the sum of the complexity of the ST applied to the testing set and the computational complexity of Random Forest model for prediction. The computational complexity of the ST applied to the test set is $O(pls)$, being l the number of samples in the test partition. The computational complexity of prediction of the Random Forest is $O(t\log l)$. Finally, the DFST prediction time complexity is: $O(pls) + O(t\log l)$.

Comparing our proposal with FS, we see that our proposal can be applied in Big Data environments and obtain better classification results due to the inclusion of ST in the training and classification phase. The use of this transformation together with traditional classification algorithms has proven to offer better results than the use of a traditional classification tree. If we compare our proposal with the ST, which has a computational complexity $O(n^2m^4)$, we see that DFST obtains a lower order complexity, $O(nm^2) + O(pns) + O(tklog)$, which allows it to be used in Big Data environments with similar accuracy results.

4 Empirical study

To asses the effectiveness and performance of our proposal, we have developed a thorough experimental setup. It is described in this section along with the analysis of the experimental results. In Section 4.1, we present the experimental framework as well as the details of the datasets and the parameters used in the methods. In Section 4.2, we present the results of performance over huge datasets in a Big Data environment. These datasets can not be processed/handled by the original algorithm or typical computers. In Section 4.3, we present the accuracy results of DFST on the datasets used in the previous section. In Section 4.4, we show the utility of shapelets as input characteristics to the classification models created by DFST.

The source code of our proposal, the datasets creation files, results and additional material are available online [3].

4.1 Experimental Framework

Since no publicly available large time series classification dataset could be found, we have created two classification problems[4]. For each problem, multiple datasets have been created with different numbers of instances. The number of instances varies from 100,000 to 20 million. The length of the time series created are 100, for first problem, and 150, for the second problem.

As a first problem, we have simulated 4 ARIMA models with `arima.sim()` function from “stats” package of the R language [17] over random sequences of 0 and 1. Each model has been assigned a classification label, Table 1. For the second problem, we have selected 6 datasets of time series classification problems from the UCR repository: ECG5000, PhalangesOutlinesCorrect, Two_Patterns, Gun_Point, wafer and ElectricDevices. These datasets are representative of the large groups of existing problems: ECG (Electrocardiograms), Image, Motion, Sensor, Simulated and Device, respectively. Each dataset has been assigned a classification label, Table 1.

For our experiments, we have used a Big Data cluster composed of one master node and 20 computing nodes. The computing nodes hold the following characteristics: 2 $\times$ Intel(R) Xeon(R) CPU E5-2620 processors, 6 cores per processor with HyperThreading, 2.00 GHz, 2 TB HDD (1 TB HDFS), 64 GB RAM. We have used the following software configuration: CentOS 6.9, Hadoop 2.6.0-cdh5.4.3 from Cloudera open source Apache Hadoop distribution, Apache Spark and MLlib 1.6.0, 23 threads/node, 1040 RAM GB (52 GB/node).

To obtain the sequential results in a comparable setting we have run the sequential algorithm in one of the nodes.

[3]Additional material on the Distributed FastShapelet Transform (DFST) proposal. `http://dicits.ugr.es/papers/DFST/`

[4]Popular dimensions datasets for benchmark, for example UCR datasets, lacks behind the values that currently qualify as starting Big Data dimensions

Table 1: Classification Problems Data

Classification Problem 1: Random sequences of 0 and 1 processed by 4 ARIMAs

Class	Model	Coeff
0	ARIMA(1,0,1)	AR(0.6), MA(0.1)
1	ARIMA(1,0,2)	AR(0.5), MA(-0.6,0.6)
2	ARIMA(2,0,2)	AR(0.5,-0.7), MA(-0.6,0.5)
3	ARIMA(2,1,2)	AR(0.5,-0.7), MA(-0.6,0.5)
Dataset size: 16,000,000		Time series length: 100

Classification Problem 2: 6 Real Datasets with 10% of White Noise

Class	Dataset
0	ECG5000
1	PhalangesOutlinesCorrect
2	Two_Patterns
3	Gun_Point
4	wafer
5	ElectricDevices
Dataset size: 8,000,000	Time series length: 150

Table 2: Experimentation Configuration Values

Parameter	Value
minLen	10
maxLen	100 (Problem 1) / 150 (Problem 2)
step	10
R	1
topK	10

4.2 Performance in Big Data environments

We are mainly concerned with accuracy and scalability of the proposed approach. Since no other scalable algorithm is available we will compare DFST to the FastShapelet algorithm, because they share some basic ideas.

The experimental configuration used is presented in Table 2. In Table 3 and 4 we show the total running times of our proposal for Big Datasets. In both cases we can see that starting from 500,000 time series onwards our distributed proposal needs lower running times than those obtained by the sequential algorithm. From this point on, as the number of time series processed increases, so does the difference between the two algorithms. Figures 2 and 3 show these results graphically. In both figures we can see the point from which the DFST improves over the sequential algorithm. We can also appreciate a linear relationship between the running time and the number of time series. The expressions of the line that defines the execution time behavior of each algorithm have been included. The execution times of problem 2 show increments in each step greater than those obtained in problem 1. This is mainly due to the fact that the time series of problem 2 have a length of 150, which is 50% greater than that of problem 1, that is 100.

Regarding the amount of data, for the first problem we have done experiments with DFST on the cluster with up to 16 millions of time series with a length of 100. The sequential algorithm can process with up to 2,200,000 time series with a length of 100. For the second problem, DFST on the cluster with up to 8 millions of time series with a length of 150. The sequential algorithm can process with up to 1,100,000 time series with a length of 150. Sequential or iterative algorithms have limitations in terms of the amount of data they can process. DFST has shown to be able to cope with datasets of any size. This is the definition of scalability.

A complementary view of the proposal could be provided through the speedup. Because of different issues this value is not very meaningful in this case. To begin with, the large difference between the size of the dataset processable by the sequential approach with respect to the parallel one. In addition, the effective gain in performance for the parallel

Table 3: Sequential FS and DFST Running Time vs Number of Time Series Problem 1

Number of time series	Sequential FS Running time (s)	DFST Running time (s)
100,000	2,088.06	12,190.91
200,000	4,119.02	12,310.29
300,000	6,699.75	9,748.17
400,000	7,909.84	11,222.16
500,000	11,643.08	9,025.84
600,000	13,407.48	12,057.48
700,000	16,317.37	8,965.22
800,000	18,454.49	9,798.00
900,000	20,764.48	12,226.55
1,000,000	22,513.41	11,855.29
1,100,000	24,622.18	9,651.16
1,200,000	26,634.61	11,536.39
1,300,000	30,618.57	13,686.43
1,400,000	30,524.07	13,741.24
1,500,000	33,556.26	13,075.48
1,600,000	36,532.96	14,814.28
1,700,000	38,695.64	14,998.85
1,800,000	41,606.93	15,583.09
1,900,000	35,099.40	16,625.55
2,000,000	49,435.64	15,128.27
2,100,000	48,421.24	15,692.12
2,200,000	53,036.16	15,667.97
3,000,000	NC	20,560.59
4,000,000	NC	26,823.02
5,000,000	NC	27,244.44
6,000,000	NC	33,295.01
7,000,000	NC	35,709.65
8,000,000	NC	42,389.48
9,000,000	NC	44,831.42
10,000,000	NC	52,685.68
11,000,000	NC	61,208.60
12,000,000	NC	67,723.17
13,000,000	NC	75,183.40
14,000,000	NC	75,973.21
15,000,000	NC	98,655.94
16,000,000	NC	103,155.57

NC indicates cases not computable by sequential algorithm.

approach is better realized for bigger case sizes, where the sequential approach cannot be applied —we would have to wait for it to finish an unacceptably long time or we would run out of main memory.

Next, the difference between the platforms. Big Data platforms are designed with scalability and easiness of use in mind, with high performance at a second level of importance. A completely parallel solution (based on communications through main memory) is scalable up to a limited size. A distributed approach, e.g. based on message passing, is more scalable although the required design and code effort is larger. In addition, its deployment is more troublesome and not feasible for the average programmer. Instead, Big Data platforms have become very popular at the price of lower performance with respect the optimal usage of resources. We selected a Big Data platform to implement DFST with idea of having a wider set of potential users.

Finally, the programming language also has an impact. The sequential algorithm is coded in C++, allowing for an effective usage of the underlying hardware platform. On the other hand, Big Data platforms are coded either on Java or languages that compile to Java Virtual Machine. Their performance falls behind that of C++, and while this effect in terms of complexity would fall under a constant term it is usually a large one.

Figure 2: Sequential FS and DFST Running Time vs Number of Time Series Graph Problem 1

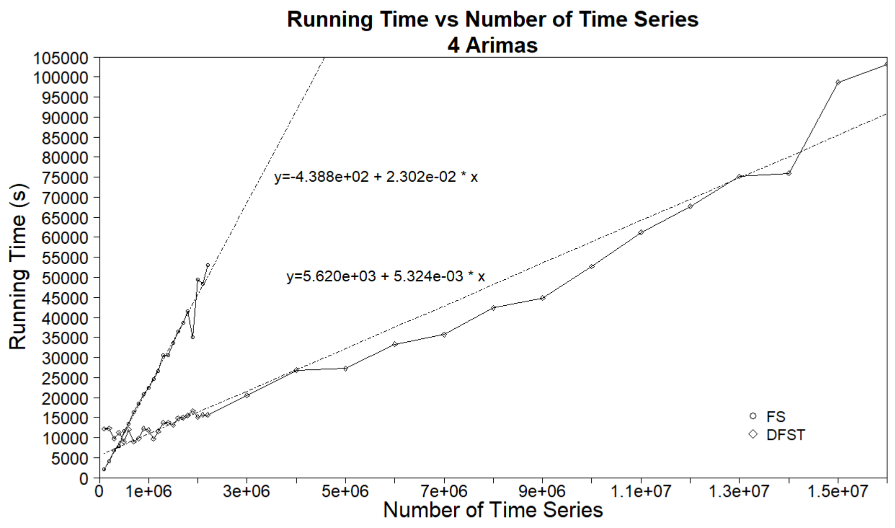


Table 4: Sequential FS and DFST Running Time vs Number of Time Series Problem 2

Number of time series	Sequential FS Running time (s)	DFST Running time (s)
100,000	3,801.29	13,608.42
200,000	8,155.19	24,660.67
300,000	13,564.75	18,829.35
400,000	17,335.96	19,032.37
500,000	22,330.90	20,188.38
600,000	28,371.62	16,507.55
700,000	35,450.22	21,773.58
800,000	36,850.60	16,073.29
900,000	39,122.43	16,882.79
1,000,000	47,092.72	20,669.58
1,100,000	51,151.38	28,544.63
2,000,000	NC	27,774.42
3,000,000	NC	39,545.70
4,000,000	NC	70,036.37
5,000,000	NC	71,690.11
6,000,000	NC	72,842.22
7,000,000	NC	88,590.94
8,000,000	NC	112,239.49

NC indicates cases not computable by sequential algorithm.

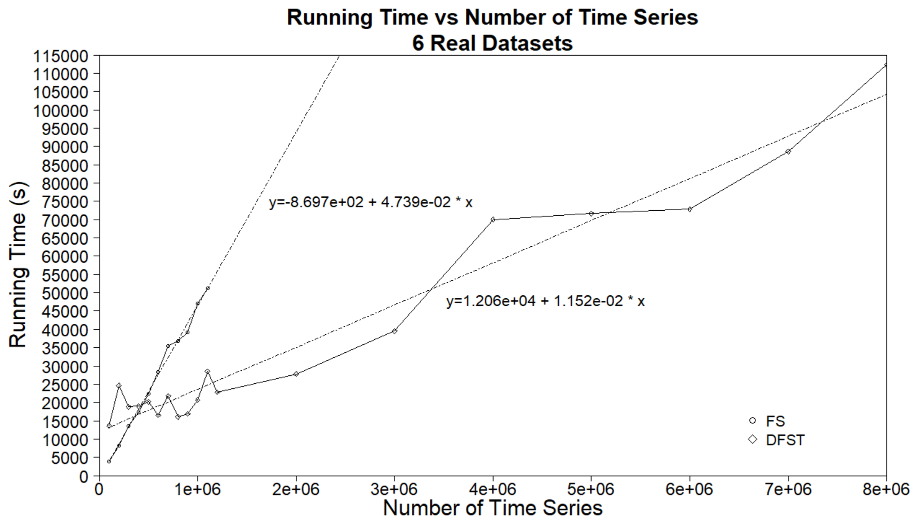
After the considerations made above, we made a computation of the running times ratios sequential vs MapReduce for the largest size that the sequential approach can handle and obtained a value of 4. This entails us to conclude the following advice: when the running time of the sequential approach is acceptable use it, otherwise use DFST.

4.3 Accuracy

To the best of our knowledge, there is currently no other proposal that can be applied to such massive time series datasets. As there are no time series classification algorithms in Big Data environments, the comparison of accuracy is not feasible. Instead, the accuracy of our proposal has been compared with that obtained by the FS algorithm, which has been the basis of the shapelet search process implemented in our proposal. This comparison has only been possible up to the processing limit of this algorithm on current computers.

Accuracy is measured in terms of the rate of correctly classified time series, as a percentage. In the ARIMAs problem, the sequential FS algorithm has obtained an average accuracy of 93.11% for the problems that has been able to process.

Figure 3: Sequential FS and DFST Running Time vs Number of Time Series Graph Problem 2



For these problems DFST has obtained an average accuracy of 99.32%, being higher than average accuracy obtained by the sequential algorithm. Over all processed datasets, 100,000 to 16,000,000,000 time series, DFST has achieved an average accuracy of 99.40%. Finally, DFST has obtained 99.66% accuracy on the largest dataset processed.

For the second problem, the sequential FS algorithm has obtained an average accuracy of 80.07% for the problems that has been able to process. For these problems, DFST has obtained an average accuracy of 82.60%, again higher than average accuracy obtained by the sequential algorithm. Over all processed datasets, 100,000 to 8,000,000,000 time series, DFST has achieved an average accuracy of 81.92%. Finally, DFST has obtained 74.17% accuracy on the largest processed dataset.

4.4 Interpretability of models

The learning model used in our proposal is a Random Forest that uses as input a distance matrix created from the extracted shapelets. Each row refers to an input time series and each column contains the distance from that time series to a shapelet. It is logical that the distance between a shapelet of a class and a time series of the same class is close to 0. On the other hand, the distance between a shapelet and a time series of different classes will be distant to 0. Since we have a large number of shapelets of different classes, there are a large number of combinations that allow a machine learning algorithm to find the relationships between the different shapelets that define the different classes. Figure 4 shows an example of the distances from different shapelets to a time series.

At the top of Figure 4, you can see how different class 0 shapelets are similar to time series of the same class. In this example we obtain two Euclidean distances equal to 0 and one of 1.39. In the lower part of Figure 4, you can see how the shapelets of classes 1 and 2 are quite dissimilar to the time series of class 0, obtaining distances equal to 2.21 and 2.77, respectively. In both examples, the shapelets have been placed at points where the distance between the shapelet and the time series is minimized.

To summarize, shapelets are graphical features that adapt to time series shapes. They are the building blocks (input features) to the classification models built with DFST.

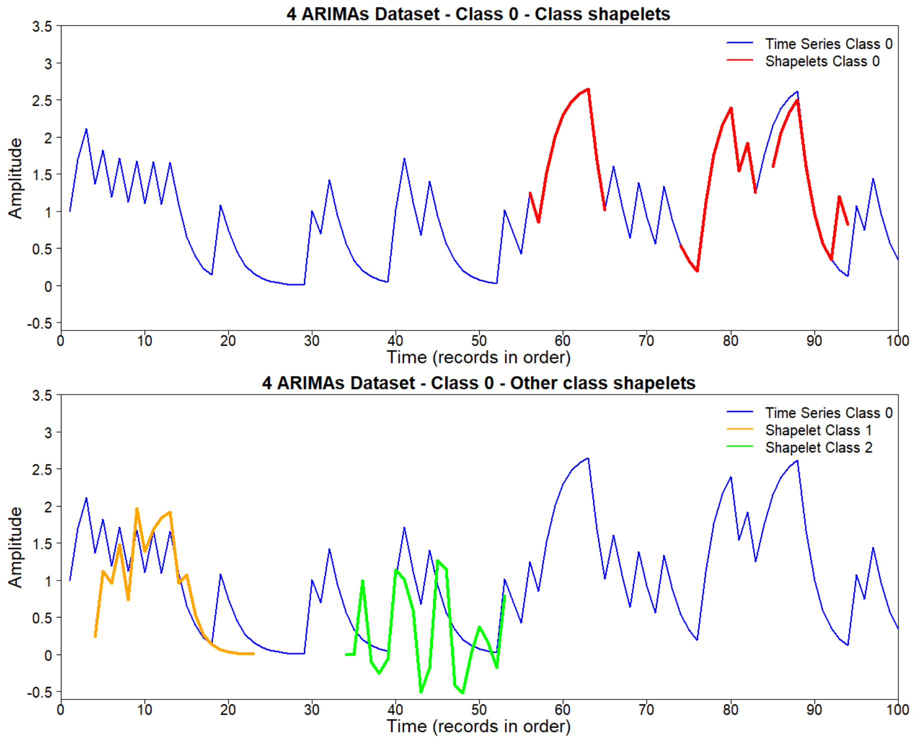
The online resource[5] includes some examples for the different classes of the 2 problems proposed in this paper.

5 Conclusions

In this work, we have proposed the Distributed FastShapelet Transform algorithm (DFST), based on the MapReduce paradigm. It is the first proposal of a completely scalable algorithm for time series classification. The state of the art of time series classification problems is dominated by algorithms such as COTE, very effective but also with a very high computational complexity, i.e. $O(n^2m^4)$. However, complexities this high prevent their application in a Big Data environment. Alternative proposals, like FS, reduce the complexity to the levels of $O(nm^2)$, at the price of a lower

[5]Additional material on the Distributed FastShapelet Transform (DFST) proposal. `http://dicits.ugr.es/papers/DFST/`

Figure 4: Example of shapelets interpretability.



accuracy. DFST addresses both issues providing a linear complexity (with respect to the number of time series) approach with comparable accuracy results. Inspired on the shapelets search procedure of the FS algorithm, a number of decisive steps have been redesigned like the use of gain of information measures, the selection from multiple shapelets, or the generalization to allow the use of different classifiers, not being tied to classification trees.

The proposed algorithm has been implemented in the Apache Spark framework and is now available as an open-source contribution to the MLlib, making it available for any practitioner o researcher to use. We have carried out a thorough empirical study focused on scalability and accuracy. The accuracy of DFST is higher than that obtained by FS in all the cases were both could be applied. The linear complexity of the final algorithm allows for the application of the algorithm on datasets of any size.

6 Acknowledgements

This research has been partially funded by the following grants: TIN2013-47210-P and TIN2016-81113-R both from the Spanish Ministery of Economy and Competitivity, and P12-TIC-2958 from Andalusian Regional Government, Spain. Francisco J. Baldán holds the FPI grant BES-2017-080137 from the Spanish Ministery of Economy and Competitivity.

7 References

References

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

[2] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: the collective of transformation-based ensembles, IEEE Transactions on Knowledge and Data Engineering 27 (9) (2015) 2522–2535.

[3] A. Bostrom, A. Bagnall, Binary shapelet transform for multiclass time series classification, in: Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII, Springer, 2017, pp. 24–46.

[4] L. Breiman, Bagging Predictors, Machine Learning 24 (2) (1996) 123–140.

[5] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[6] J. Buhler, M. Tompa, Finding motifs using random projections, Journal of computational biology 9 (2) (2002) 225–242.

[7] A. CCAFS Climate Change, F. Security, CCAFS-Climate Data Weather Stations, `http://www.ccafs-climate.org/weather_stations/`, accessed: 2018-01-31.

[8] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[9] T. A. S. Foundation, Apache Hadoop, `http://hadoop.apache.org/`, accessed: 2018-01-31.

[10] T. A. S. Foundation, Apache Spark: Lightning-fast cluster computing, `https://spark.apache.org/`, accessed: 2018-01-31.

[11] Y.-S. Jeong, R. Jayaraman, Support vector-based algorithms with weighted dynamic time warping kernel function for time series classification, Knowledge-based systems 75 (2015) 184–191.

[12] J. Lin, E. Keogh, W. Li, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, Data Mining and knowledge discovery 15 (2) (2007) 107.

[13] J. Lines, L. M. Davis, J. Hills, A. Bagnall, A shapelet transform for time series classification, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2012, pp. 289–297.

[14] G. Louppe, Understanding random forests: From theory to practice, arXiv preprint arXiv:1407.7502v3.

[15] A. Mueen, E. Keogh, N. Young, Logical-shapelets: an expressive primitive for time series classification, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2011, pp. 1154–1162.

[16] F. R. B. of ST. Louis, Federal Reserve Bank of ST. Louis, `https://fred.stlouisfed.org/`, accessed: 2018-01-31.

[17] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2015).
URL `http://www.R-project.org/`

[18] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh, Data mining a trillion time series subsequences under dynamic time warping, in: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, AAAI Press, 2013, pp. 3047–3051.

[19] T. Rakthanmanon, 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.

[20] A. W. Services, CCAFS-Climate Data, `https://aws.amazon.com/es/datasets/ccafs-climate-data/?tag=datasets%23keywords%23climate`, accessed: 2018-01-31.

[21] A. W. Services, Federal Reserve Economic Data - Fred, `https://aws.amazon.com/es/datasets/federal-reserve-economic-data-fred/?tag=datasets%23keywords%23economics`, accessed: 2018-01-31.

[22] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 947–956.

[23] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, I. Stoica, Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, USENIX Association, 2012, pp. 2–2.

[24] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster Computing with Working Sets., HotCloud 10 (10-10) (2010) 95.

2 Complexity Measures and Features for Times Series classification

- F. J. Baldán, J. M. Benítez. Complexity Measures and Features for Times Series classification. Submitted.
  - Status: **Submitted**.

Complexity Measures and Features for Times Series classification

Francisco J. Baldán[*]
Department of Computer Science
and Artificial Intelligence
University of Granada, DICITS, iMUDS, DaSCI
Granada, Spain, 18071
`fjbaldan@decsai.ugr.es`

José M. Benítez Sánchez
Department of Computer Science
and Artificial Intelligence
University of Granada, DICITS, iMUDS, DaSCI
Granada, Spain, 18071
`J.M.Benitez@decsai.ugr.es`

Abstract

Classification of time series is a growing problem in different disciplines due to the progressive digitalization of the world. Currently, the state-of-the-art in time series classification is dominated by The Hierarchical Vote Collective of Transformation-based Ensembles. This algorithm is composed of several classifiers of different domains distributed in five large modules. The combination of the results obtained by each module weighed based on an internal evaluation process allows this algorithm to obtain the best results in state-of-the-art. One Nearest Neighbour with Dynamic Time Warping remains the base classifier in any time series classification problem for its simplicity and good results. Despite their performance, they share a weakness, which is that they are not interpretable. In the field of time series classification, there is a tradeoff between accuracy and interpretability. In this work, we propose a set of characteristics capable of extracting information on the structure of the time series to face time series classification problems. The use of these characteristics allows the use of traditional classification algorithms in time series problems. The experimental results of our proposal show no statistically significant differences from the second and third best models of the state-of-the-art. Apart from competitive results in accuracy, our proposal is able to offer interpretable results based on the set of characteristics proposed.

Keywords Classification · Complexity measures · Time series features · Time series interpretation

1 Introduction

At present, large amounts of information are recorded from a wide variety of fields. There is a growing need to analyze and classify these data to obtain useful information, for example, to identify different patterns of electricity consumption in order to adapt prices to consumers [42], to identify cardiac anomalies characteristics of a pathology [17] or search for anomalies in starlight curves [59].

The field of time series classification (TSC) [5] has historically been dominated by proposals that offer good classification results but are hardly interpretable. For example, a simple approach that achieves good average results in the different types of problems is One Nearest Neighbour with Dynamic Time Warping (1NN+DTW) [11] [51]. This approach tells us how similar the time series are to each other, but it does not allow us to extract additional information from the problem. Recently the Collective Of Transformation Ensembles (COTE) [6] has been shown to obtain the best TSC results on the reference time series database collected in the UCR repository [19], in the 2015 version of this repository. This algorithm is composed of 35 classifiers (flat-COTE) which apply cross-validation on the training set. COTE contains reference classifiers in the fields of TSC. These classifiers are evaluated internally with cross-validation, and depending on their results, they are included in the final result. Recently The Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) [3] has been proposed, which improves the classification process carried out by its previous versions. HIVE-COTE is composed of several classifiers of different domains distributed in five large modules. Each

[*]Corresponding author.

module provides a probability estimate for each class and obtains a weighting proportional to the accuracy achieved over the training set. HIVE-COTE combines these estimates in a second layer and obtains the predicted class from the highest weight over all the modules. The HIVE-COTE proposal provides the best results, but its interpretability is very low, and its high computational cost prevents its application in large datasets.

Other more interpretable approaches as decision trees do not usually obtain competitive results in the field of TSC. This behavior is due to their inability to capture the time relationships between the different time instants that make up a time series. These approaches are successfully used in combination with other proposals, such as shapelets, which extract behavioral patterns from time series [63]. These patterns make it possible to differentiate time series belonging to different classes. These proposals have great interpretability since they allow us to identify, in a graphical way, patterns of interest belonging to the different classes that compose the problem. Although, in this case, there are also proposals such as the Shapelet Transform (ST) [40], which transforms these shapelets into features. ST alter the problem of TSC into a traditional, vector-based, classification problem, on which we can apply traditional algorithms, such as Random Forest (RF) [14], and obtain good results. In this way, there are proposals in Big Data such as Distributed FastShapelet Transform [7] that allows us to face TSC problems in massive data environments where traditional TSC algorithms cannot be applied due to their high computational complexity. There is a more recent proposal, which proposes the creation of a weighted ensemble of standard classifiers, such as Random Forest, Naive Bayes, Support Vector Machines, among others, on the transformed data, obtaining very competitive results. This proposal is named Shapelet Transform Classifier (STC) [13].

In the literature, we can find proposals focused on extracting a large number of characteristics from time series [30] [28]. The main idea of these characteristics is any type of mathematical operation applicable over a time series that provide valuable information. The objective of these proposals is to look for an underlying structure that represents the behavior of a time series. These types of studies are ambitious but difficult to interpret over a specific problem due to the high number of characteristics present. Moreover, these studies are oriented to the unsupervised learning environment. Some proposals make a selection of the main characteristics of a time series, from the theoretical point of view that could explain the origin of their behavior [33]. The objective of the previous work is to generate synthetics time series that represent real problem behaviors, so its main target is far from the problem of TSC. On the other hand, CAnonical Time-series CHaracteristics (catch22) [41] proposes a set of 22 characteristics that have been selected based on the classification results obtained on a large set of datasets. For this proposal, a large number of characteristics and their possible combinations have been tested, measuring the classification results obtained. The main criterion for selecting the characteristics is to provide the best possible results, although the execution time and, in some cases, their interpretability is also taken into account. Recently a method has been proposed in this line. This method, called Feature and Representation Selection (FEARS) [12], is based on obtaining different alternative representations such as derivatives, cumulative integrals, power spectrum, among others, of the time series. This method then extracts characteristics of interest using an automatic variable construction technique. As the last step, a Naive Bayes classifier is in charge of learning about the new extracted characteristics. This procedure is repeated several times to obtain the most informative set of characteristics possible.

There are other proposals based on studying the complexity of time series [65] [46]. These proposals use complexity measures that measure the interrelationships between the different values in a time series. A greater number of relationships lead to greater complexity. In the same way that the traditional characteristics of the time series are capable of providing sufficient information about a problem, complexity measures can add useful information to the problem.

In this work, we present a set of characteristics composed of complexity measures and representative features of the time series. This transformation allows the use of traditional learning algorithms on TSC problems. Additionally, this transformation allows interpreting the results obtained by the classification algorithms. The performance of our proposal has been tested on a set of 112 datasets present in the UCR repository. We have applied the most popular and widely used classification algorithms based on trees that allow interpretable results. Our proposal is publicly available as an R package in the online repository[2].

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

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

2 Related work

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

2.1 Distance-based Classification

Patterns searched for in TSC problems may have their origin in different domains. For this reason, there are different types of approaches depending on multiple factors. There are currently six main approaches for dealing with this kind of TSC problems [5], grouped by the type of discriminatory features that the technique attempts to find:

- Proposals that use all the values of the time series: are linked to the use of similarity measures and different types of distance. The reference algorithm of this group is 1NN+DTW, which is simple to apply but has high computational complexity. This algorithm is often used as a benchmark in TSC problems.
- Those using phase-dependent intervals: they use small subsets from each time series, rather than using the entire time series. Proposals like Time Series Forest (TSF) [25] have been proved that extracting characteristics such as mean, variance, or slope from random intervals, and use them as classifier features, works particularly well. Characteristics such as Fourier, autocorrelation, and partial autocorrelation, which are more complex and related to the time series than those mentioned above, are used by more recent proposals such as contract Random Interval Spectral Ensemble (c-RISE) [27], with very competitive results.
- The independent phases, based on shapelets: the shapelets based ones look for substrings of the time series that allow differentiating the time series belonging to each class. They are closely linked to the use of similarity and distance measurements. The first proposals generated simple classification trees capable of differentiating the belonging of a time series to one class or another according to the presence or not of a certain subsequence in it [63] [50] [43]. These approaches offered some interpretability to the results. Recent work on shapelets has shown that they are particularly useful when used as input features to a traditional classification algorithm [40] [13] [7], rather than as part of the classification tree itself.
- Based on dictionaries: in some cases, the presence of a certain pattern in a time series is not enough to identify whether it belongs to one class or another [39] [53]. There are problems in which the number of times the pattern appears in a time series is determinant to classify it correctly. The shapelets are not useful in these cases, and the use of algorithms based on dictionaries is mandatory. These algorithms count both the presence or absence of each subsequence in a time series. They create a classifier based on the histograms obtained from these dictionaries. The way of creating the dictionary is one of the main differences among the proposals of this type. For example, Bag of patterns (BOP) [38] creates the dictionary through the Symbolic aggregate ApproXimation (SAX) [37] words extracted from each window. Symbolic Aggregate approXimation-Vector Space Model (SAXVSM) [56] combines the SAX representation used in BOP with the vector space model commonly used in Information Retrieval and counts the appearance frequencies over the classes and not over the time series. Bag of SFA symbols (BOSS) [53] does not use Piecewise Aggregate Approximation (PAA) [34] in its SAX transformation but uses truncated Discrete Fourier Transform (DFT). Furthermore, it uses the so-called Multiple Coefficient Binning (MCB) technique to discretize the truncated time series, among other differences. Despite the good results, BOSS does not scale well, so it made a proposal called contracted BOSS (cBOSS) [35], which modified the way BOSS classifiers were chosen, indicating construction time limits per classifier and saving the advances during the construction process without significant accuracy changes. Word ExtrAction for time SEries cLassification (WEASEL) [54] is one of the latest proposals made. WEASEL has highly competitive results and differs from the rest by its ability to derive the characteristics obtained, achieving a new, much smaller, and more discriminating set of features.
- Based on models: this approach is mainly oriented to problems with long time series, but with different lengths [4] [18]. These proposals usually fit a model to each time series and measure the similarity between the models. It is an approach that is not sufficiently widespread and is applied to particular problems.
- Combinations or ensembles: this approach works both in time series and traditional classification problems, using the results of different models to make a final decision. In the area of time series, HIVE-COTE [3] is the best proposal to date. It uses models from different approaches and offers the best accuracy results. On the

other hand, it is the approach with the highest computational complexity due to the high number of algorithms it uses and its corresponding computational complexities. Moreover, this large number of algorithms leads to low interpretability of results.

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

2.2 Feature-based Classification

The feature-based approach is focused on a transformation to the time series dataset, obtaining a new dataset composed of different features that explain the behavior of the original time series [28]. The feature-based approach offers multiple advantages over the distance-based approach for dealing with time-series classification problems. This approach allows analysis of time series on different time domains and with different lengths, being more widely applicable because the stationarity properties of the series are not always required. In addition, this approach allows us to use the standard classification and clustering methods that have been developed for non-time series data. In this approach, we can found two different approximations:

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

  Based on this approach, we can find proposals that, with a minimum of four initial characteristics such as mean, typical deviation, skewness, and kurtosis, are able to face the problem of the classification synthetic control chart patterns used in the statistical process control [44]. There are also proposals, focused on the improvement of accuracy, based on the creation of an ensemble for classification, composed of trained classifiers on different representations of the time series [2]: power spectrum, autocorrelation function, and a principal components space. The final classification is obtained from a weighted voting scheme. In the field of clustering, some proposals use characteristics of time series such as trend, seasonality, non-linearity, among others, which are very appropriate to express the behavior of a time series [60].

  In this approach, we also find proposals that aim to generate synthetic time series with a given behavior as close as possible to a real time series [33]. This work contains a selection of the main characteristics of a time series. Its objective is to use them to generate time series with a real behavior with these controllable parameters.

- The second approach focuses on applying a large number of different operations to obtain a great set of descriptive parameters of the time series analyzed. In this approach, the selection of the characteristics of interest resides in the learning algorithm used on the transformed dataset. Having a much greater set of characteristics than the first approach allows us to capture a higher number of behaviors of interest, improving the results of the algorithms applied afterward. But it is hard to extract useful information because there are a large number of characteristics to analyze. In addition, it is possible that a large part of the selected characteristics is not as explanatory as the characteristics with a strong theoretical base such as trend, seasonality, among others.

  In this area, we can find different proposals. For example, the use of 8,651 operations on a set of 875 time series [30], coming from different fields, with the aim of extracting the different possible structural behaviors. Another of its objectives is to find possible interrelations between time series coming from different fields. Given the rearrangement of the rows (original time series) in the final matrix of characteristics, based on the similarity between the different operations calculated, this work can be included within the field of clustering. Another objective of the previous work would be to find a shared underlying structure between time series belonging or not to the same scope.

In a more controlled environment, within the reference problems of classification of time series, we found a similar proposal to the previous one. In this case, the authors seek to obtain the best classification results by working on the transformed dataset [29]. It has almost 9,000 characteristics, being of special importance the way to select the variables of interest. This proposal opted for the selection of the combination of variables that offers the best classification results, using the following procedure:

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

2.3 Deep Learning Classification

The approach based on deep learning has gained popularity recently [26]. Although it is usually related to the processing of images, it has very interesting proposals in the field of TSC [62]. We can distinguish between two main groups inside this approach: Generative Models and Discriminative Models.

- In the Generative Models, there is usually a previous step of unsupervised training to the learning phase of the classifier. Depending on the approach, two subgroups can be differentiated: Auto Encoders and Echo State Networks. In the case of Auto Encoders, there are a large number of proposals, for example, to model the time series before the classifier is included in an unsupervised pre-training phase such as Stacked Denoising Auto-Encoders (SDAEs) [10]. A Recurrent Neural Network (RNN) Auto Encoder [49] was designed to generate time series first and then use the learned representation to train a traditional classifier. After that, it predicts the class of the new input time series. A model based on Convolutional Neural Networks (CNN) [57] was proposed where the authors introduced a deconvolutionary operation followed by an upsampling technique that helps to reconstruct a multivariate time series. In the case of the Echo State Networks, these networks were used to reconstruct time series and use the representation learned in the space reservoir for classification. They were also used to define a kernel on the learned representations and apply an MLP or SVM as a classifier.
- In the case of Discriminative Models, these are a classifier or regressor that learns the mapping between the input values of the time series and returns the probability distribution over the class variable of the problem. In this case, we can differentiate two subgroups: Feature Engineering and End-to-End. The typical case of use of Feature Engineering is the transformation of the time series into images, using different techniques such as recurrence plots [31] and Markov transition fields [61], and introduce that information in a deep learning discriminating classifier [45]. In contrast, the End-to-End approach incorporates feature learning while adjusting the discriminative classifier.

If we look at the TSC problem, we see that the CNNs are the most used architectures, mainly due to their robustness and their relatively short training time, compared to other types of networks. One of the best-known architectures is the Residual Networks (ResNets) [62]. This proposal adds linear shortcuts for the convolutional layers, potentially improving the accuracy of the model.

3 Time series complexity measures and features

The complexity of a time series represents the interrelationship that exists between its different elements. A greater number of interrelations between the elements of a time series indicates a greater complexity. Once these interrelations have been found and understood, we can try to find the mechanisms that produce this complexity. In this way, it is possible to explain the behavior of a time series based on these mechanisms. In other words, these interrelations are characteristic of the time series.

The features of a time series explain certain behavioral characteristics of the time series itself. The features that traditionally have been used in the process of analysis of a time series as seasonality, trend, stationarity, among others, are well documented [33] [8]. These types of characteristics can describe the behavior of a time series efficiently. There are other types of characteristics that provide small pieces of information about the behavior of a time series, such as mean, maximum value, minimum value, variance, among others. Although the latter is not usually employed in the analysis process, they are features that may be especially useful depending on the problem. For example, in a classification problem where time series of different classes have significant differences in their value ranges, the mean can be very helpful.

This work presents a novel ensemble of complexity measures and features of time series, aimed at solving problems of classification of time series by applying traditional classification algorithms. It also aims to obtain interpretable results. The characteristics selected in this paper, composed of complexity measures and time series features, are based on information theory and seek to provide greater knowledge about the underlying structure of the processed time series. A set of characteristics, based on measures of complexity, is summarized in Table 1.

In addition to the features mentioned above, we have added a set of time series features. It has been selected based on its theoretical basis, also taking into account its historical importance in the field of time series and its interpretability [33]. This set of measures, based mostly on typical characteristics of time series, is summarized in Table 2.

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

The objective of using such characteristics is to obtain an alternative and interpretable representation of the behavior of a time series. This representation allows us to use traditional classification algorithms and obtain interpretable results.

Table 1: Complexity measures selected.

Char.	Name	Description	Ref.
C_1	lempel_ziv	LempelZiv (LZA)	[36]
C_2	aproximation_entropy	Aproximation Entropy	[47]
C_3	sample_entropy	Sample Entropy (DK Lake in Matlab)	[52]
C_4	permutation_entropy	Permutation Entropy (tsExpKit)	[9]
C_5	shannon_entropy_CS	Chao-Shen Entropy Estimator	[16]
C_6	shannon_entropy_SG	Schurmann-Grassberger Entropy Estimator	[55]
C_7	spectral_entropy	Spectral Entropy	[64]
C_8	nforbiden	Number of forbiden patterns	[1]
C_9	kurtosis	Kurtosis, the "tailedness" of the probability distribution	[23]
C_{10}	skewness	Skewness, asymmetry of the probability distribution	[20]

This way, if a classification algorithm is applied that offers an interpretable model, we can explain the classification based on characteristics that describe the behavior of the processed time series. We can obtain information beyond the simple visual behavior of a time series.

The theoretical explanation of each of the measures has not been included in this paper due to space constraints. For the convenience of the reader, they are available online in the web resource [3] associated with this work.

Our proposal consists of a set of characteristics that allow us to classify in a better way the time series and to obtain interpretable results. The pseudocode in Algorithm 1 shows how our proposal works.

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

Although most of the proposed characteristics are specially designed to be applied over time series, in some cases, these characteristics may not be defined for some specific time series. In these cases, undesirable values are produced, and we must process them. In the first place, we differentiate between the cases in which we obtain infinite values and those we do not. For this reason, the results obtained are filtered, detecting the cases of noninfinite values and transforming to the same value (lines 2-5) for subsequent elimination or imputation. On the training set, we check for each column (operation applied) that the amount of these values is less than 20% of the total. In other cases, the column is removed from both the training set and the test set (lines 6-11). Infinite values are identified as positive or negative and replaced by the maximum or minimum value of the corresponding column, respectively, ignoring the infinite values in these calculations (lines 12-22). Imputation of missing values based on the mean is then applied to each column (lines 23-25), eliminating any presence of unwanted values in the datasets.

Since one of our objectives is to obtain interpretable results, in the second part of our proposal, we have selected the main classification algorithms based on trees: C5.0, C5.0 with boosting [48], Rpart [58] and Ctree [32]. We have selected this type of algorithms by the interpretability of the generated models. The accuracy of the models obtained on the test set is an objective indication of the quality or fidelity of the representation obtained by the set of selected features. We initialize a variable that contains the results obtained for each one of the processed models (line 26). In the final part, we calculate each selected model, make the corresponding prediction, and calculate the accuracy. Finally, all these results are stored (lines 27-32). Our proposal returns these results together with the training and test sets with the new calculated characteristics (line 33).

Figure 1 shows, in a graphic form, the process of calculating the characteristics of the time series.

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

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

Table 2: Time series features selected.

Char.	Name	Description
C_{11}	x_acf1	First autocorrelation coefficient
C_{12}	x_acf10	Sum of squares of the first 10 autocorrelation coefficients
C_{13}	diff1_acf1	Differenced series first autocorrelation coefficients
C_{14}	diff1_acf10	Differenced series sum of squares of the first 10 autocorrelation coefficients
C_{15}	diff2_acf1	Twice differenced series first autocorrelation coefficients
C_{16}	diff2_acf10	Twice differenced series sum of squares of the first 10 autocorrelation coefficients
C_{17}	max_kl_shift	Maximum shift in Kullback-Leibler divergence between two consecutive windows
C_{18}	time_kl_shift	Instant of time in which the Maximum shift in Kullback-Leibler divergence between two consecutive windows is located
C_{19}	outlierinclude_mdrmd	Calculates the median of the medians of the values, while adding more outliers
C_{20}	max_level_shift	Maximum mean shift between two consecutive windows
C_{21}	time_level_shift	Instant of time in which the maximum mean shift between two consecutive windows is located
C_{22}	ac_9	Autocorrelation at lag 9
C_{23}	crossing_points	The number of times a time series crosses the median line
C_{24}	max_var_shift	Maximum variance shift between two consecutive windows
C_{25}	time_var_shift	Instant of time in which the maximum variance shift between two consecutive windows is located
C_{26}	nonlinearity	Modified statistic from Teräsvirta's test
C_{27}	embed2_incircle	Proportion of points inside a given circular boundary in a 2-d embedding space
C_{28}	spreadrandomlocal_meantaul	Mean of the first zero-crossings of the autocorrelation function in each segment of the 100 time-series segments of length l selected at random from the original time series
C_{29}	flat_spots	Maximum run length within any single interval obtained from the ten equal-sized intervals of the sample space of a time series
C_{30}	x_pacf5	Sum of squares of the first 5 partial autocorrelation coefficients
C_{31}	diff1x_pacf5	Differenced series sum of squares of the first 5 partial autocorrelation coefficients
C_{32}	diff2x_pacf5	Twice differenced series sum of squares of the first 5 partial autocorrelation coefficients
C_{33}	firstmin_ac	Time of first minimum in the autocorrelation function
C_{34}	std1st_der	Standard deviation of the first derivative of the time series
C_{35}	stability	Stability variance of the means
C_{36}	firstzero_ac	First zero crossing of the autocorrelation function
C_{37}	trev_num	The numerator of the trev function, a normalized nonlinear autocorrelation, with the time lag set to 1
C_{38}	alpha	Smoothing parameter for the level-alpha of Holt's linear trend method
C_{39}	beta	Smoothing parameter for the trend-beta of Holt's linear trend method
C_{40}	nperiods	Number of seasonal periods (1 for no seasonal data)
C_{41}	seasonal_period	Seasonal periods (1 for no seasonal data)
C_{42}	trend	Strength of trend
C_{43}	spike	Spikiness variance of the leave-one-out variances of the remainder component
C_{44}	linearity	Linearity calculated based on the coefficients of an orthogonal quadratic regression
C_{45}	curvature	Curvature calculated based on the coefficients of an orthogonal quadratic regression
C_{46}	e_acf1	First autocorrelation coefficient of the remainder component
C_{47}	e_acf10	Sum of the first then squared autocorrelation coefficients
C_{48}	walker_propcross	Fraction of time series length that walker crosses time series
C_{49}	hurst	Long-memory coefficient
C_{50}	unitroot_kpss	Statistic for the KPSS unit root test with linear trend and lag one
C_{51}	histogram_mode	Calculates the mode of the data vector using histograms with 10 bins (It is possible to select a different number of bins)
C_{52}	unitroot_pp	Statistic for the PP unit root test with constant trend and lag one
C_{53}	localsimple_taures	First zero crossing of the autocorrelation function of the residuals from a predictor that uses the past trainLength values of the time series to predict its next value
C_{54}	lumpiness	Lumpiness variance of the variance
C_{55}	motiftwo_entro3	Entropy of words in the binary alphabet of length 3. The binary alphabet is obtained as follows: Time-series values above its mean are given 1, and those below the mean are 0

Algorithm 1 Main procedure

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

4 Empirical Study

In this section we evaluate the performance of our proposal. In order to do this, we first show the experimental design carried out followed by the results obtained with their corresponding analysis.

4.1 Experimental Design

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

The source code of our proposal and experimentation has been developed in R 3.4.4 and can be found in the online repository [4].

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

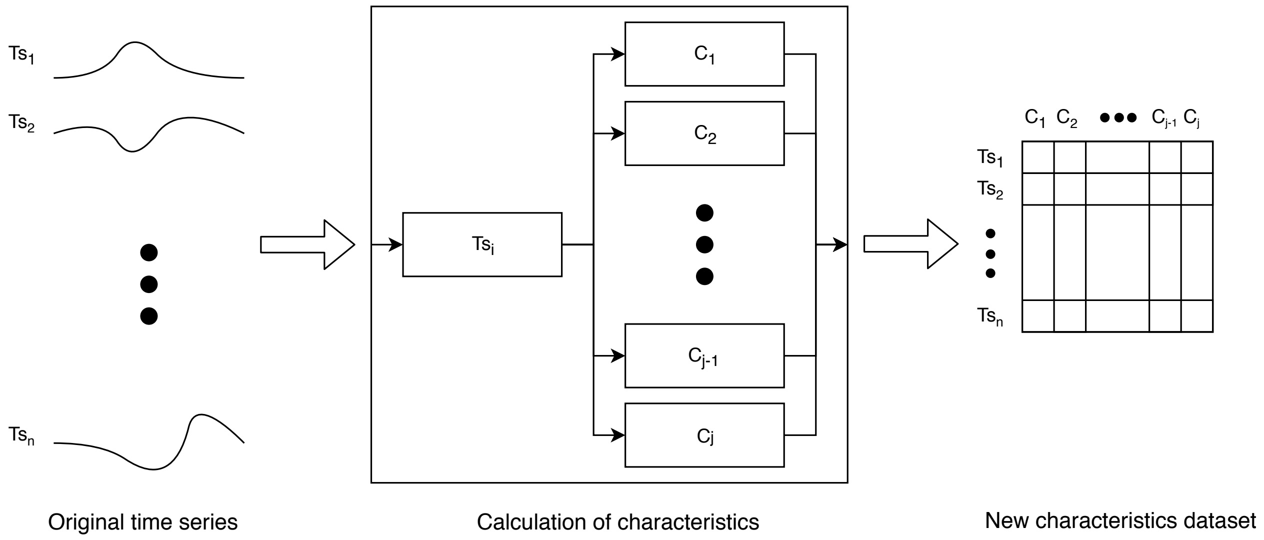


Figure 1: Characteristics calculation workflow.

4.1.1 Performance measures

We have chosen accuracy as a basic measure of performance. Accuracy is calculated as the number of correctly classified instances in the test set divided by the total number of cases in the test set. We use the average rank to compare the performance of the different models against each other. Having over a large number of datasets, very different from each other, a relative performance measure like the rank is one of the best options to make the desired comparison. Since the results can vary greatly from one dataset to another we have chosen to use the Critical Difference diagram (CD) [24]. CD allows making a comparison of results, between the different models, from a statistical point of view. In this diagram, the models linked by a bold line can be considered to have no statistically significant differences in their results at a given confidence level α. In this paper, we have chosen a 95% confidence level setting an α of 0.05. We have used the R *scmamp* package to calculate average rank and the CD. In addition, we include the Win/Loss/Tie ratio to be able to observe in a direct quantitative way the performance of each model in comparison with the rest.

4.1.2 Datasets

The used datasets have been extracted from the UCR repository [22], which is the reference repository in the field of TSC. It is composed of 128 datasets. The authors of the repository have run the main algorithms of the state of the art of TSC on 112 of the 128 datasets. They eliminated 15 datasets because of containing time series of different lengths and the Fungi dataset because it contains only one instance per class in the training data. Given the great number of algorithms run on these datasets, we can consider the 112 selected datasets as the state of the art in TSC datasets.

4.1.3 Models

The main tree classification algorithms have been selected based on their interpretability: C5.0, C5.0 with boosting (C5.0B) [48], Rpart [58], and Ctree [32]. 1NN+ED, 1NN+DTW(w=100) and 1NN+DTW(w_learned) applied over the original time series have been included as benchmark methods since they are the benchmark TSC methods. The new representation of time series that we propose in this work offers an additional information about these series that can also be used by less interpretable algorithms to improve the obtained results. For this purpose, classification algorithms with greater complexity and better accuracy performance have been selected like RF [14], and SVM [21]. We have also added 1NN+ED applied to the proposed features as a benchmark method. We name the models based on the features proposed in this work following the CMFTS+Model pattern, for example, CMFTS+RF, CMFTS+C5.0, etc.

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

4.1.4 Hardware

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

4.2 Results

In this section, we show and evaluate the results obtained by our proposal both in terms of performance (Section 4.2.1) and interpretability (Section 4.2.2). Since the complete empirical results are too extensive to include in the paper, we have put just a summary. The complete set is available at web resource[5] associated to this work.

4.2.1 Performance results

Table 3 shows the results obtained for the 112 datasets processed. We show the average accuracy, average rank, and Win/Loss/Tie Ratio, for all the feature-based learning models (CMFTS) proposed in this paper and the benchmark models in TSC.

Table 3: Comparative results of the proposed feature-based models (CMFTS) and the TSC benchmark models. The best results are stressed in bold.

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

If we look at the results of the average rank, Table 3, we see that the CMFTS+RF model obtains the best results, followed by CMFTS+SVM, CMFTS+C5.0B, and 1NN-DTW (learned_w). This shows that more complex models such as RF, C5.0B, SVM, and 1NN-DTW (learned_w) offer better results than more simple models such as C5.0, Rpart, and Ctree. This behavior is also visible in the Win/Loss/Tie Ratio, where CMFTS+RF is the best model, with 48 wins, followed by 1NN-DTW (learned_w) with 23 wins. The third, fourth and fifth places are taken by 1NNN-DTW (w=100) (16 wins), CMFTS+SVM (14 wins), and CMFTS+C5.0B (12 wins), respectively.

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

Once the best models of our proposal have been identified, we will compare them with the best models of the state of the art. The best models of our proposal selected for this comparison are CMFTS+RF, CMFTS+SVM, CMFTS+C5.0B, and CMFTS+1NN-ED. CMFTS+RF and CMFTS+SVM are the models that obtain the best results, although their interpretability is reduced. CMFTS+C5.0B is the most interpretable model with the best results if we compare it with the rest of the tree-based models. CMFTS+1NN-ED is a simple model that we can use as a benchmark. As in the previous case, for a first analysis, we use a table with the results of average accuracy, average rank, and Win/Loss/Tie Ratio, Table 4. In addition, to carry out an analysis from a statistical point of view we use the CD, Figure 3.

In Table 4, we see the HIVE-COTE algorithm has the best results in average rank, average accuracy, and win/loss/tie ratio. This algorithm should be used whenever possible. STC is the second method with the lowest average rank

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

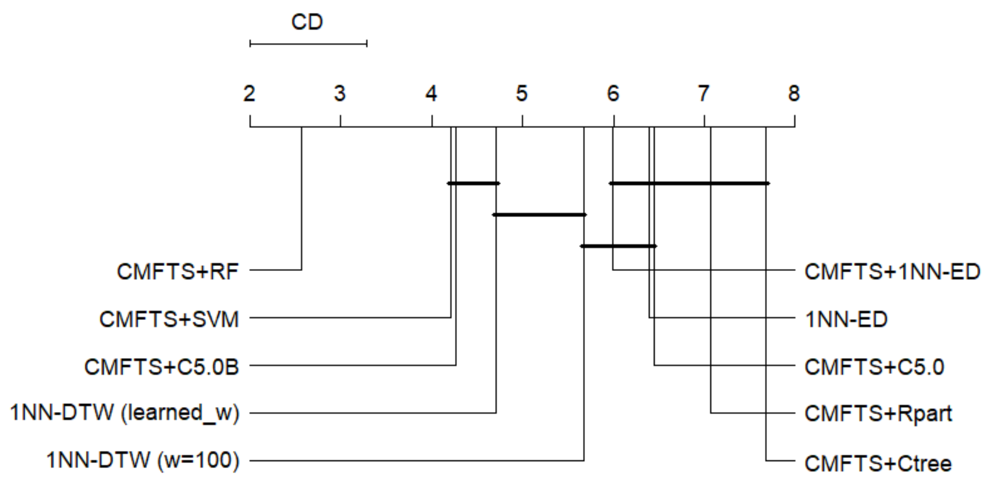


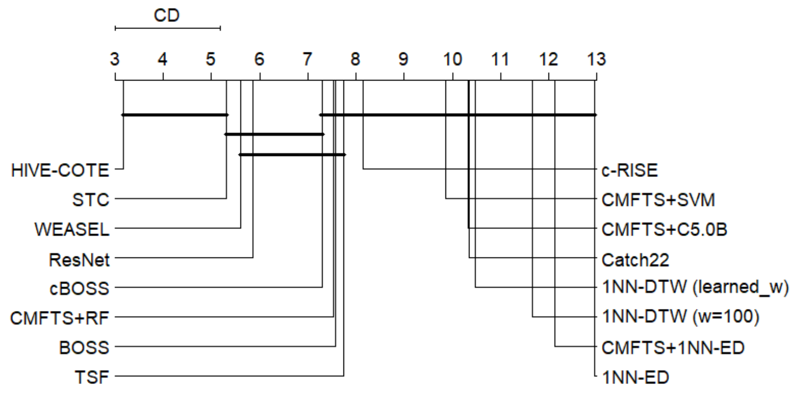
Figure 2: Critical Difference diagram between the proposed feature-based models (CMFTS) and the TSC benchmark models, confidence level of 95%.

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

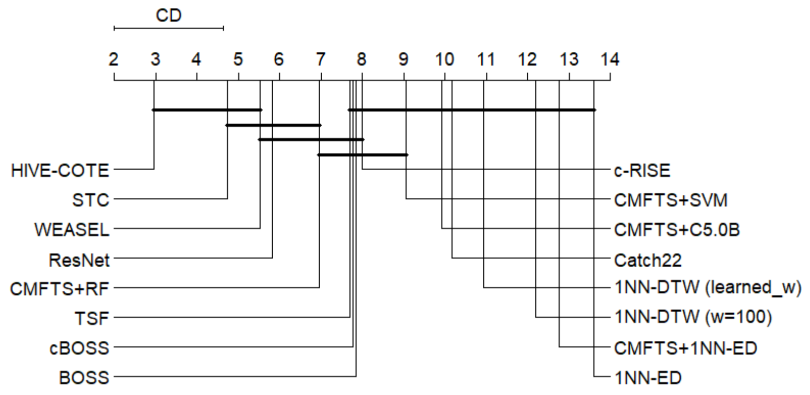
Table 4: Comparative results of the proposed feature-based models (CMFTS) and the TSC state of the art models. The best results are stressed in bold.

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

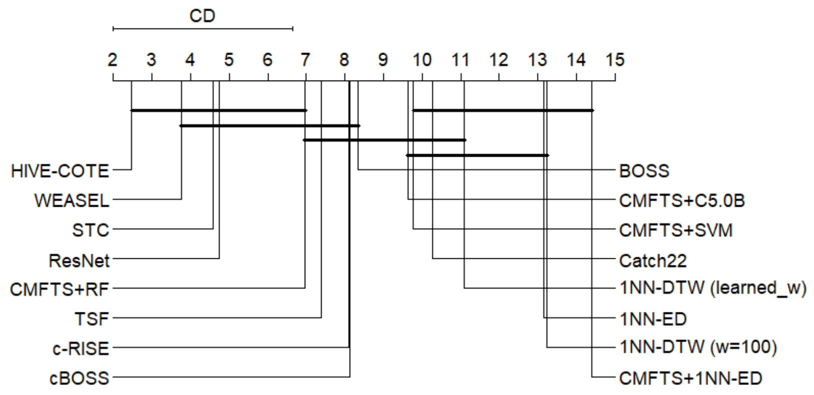
If we analyze Figure 3, we can observe statistical relationships between our proposal CMFTS+RF and the algorithms HIVE-COTE, STC, and WEASEL, with some conditions. In Figure 3a, there are four principal subgroups of proposals without statistical differences between their results over the 112 selected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



(a) Full UCR repository, 112 datasets.



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



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

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

4.2.2 Interpretability

In this section, we analyze the interpretability of the results obtained by our proposals. We also see the advantages of our proposal in terms of the robustness of results.

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

In Figure 4b, we see the first classification tree C5.0 obtained by our proposal, CMFTS+C5B. In this tree, we observe how two features like the *stability*, as the variance of the means obtained from tiled windows, and the *shannon entropy SG*, with Bayesian estimates of the bin frequencies using the Dirichlet-multinomial pseudo-counting model, can differentiate a large part of the cases that belong to a class. If we compare these results with Figure 4c, where the values of some instants of time are the ones that determine if a case belongs to different classes, we can see how our proposal offers a robust behavior to problems as simple as the desynchronization of the temporal series.

The interpretability of the results is strongly linked to the importance given by each algorithm to each of the input features, whenever it is possible. For this reason, we have selected our best proposal, CMFTS+RF, that measures the importance of each feature through the Gini Index [15]. We have analyzed the accumulated importance of each feature over the 112 datasets and the importance of each feature in each dataset.

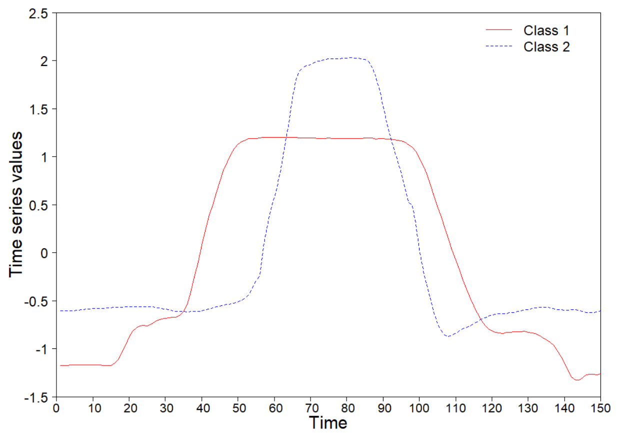
Figure 5 shows the mean results of the importance of the features obtained on the 112 datasets used. We see how characteristics related to entropy, such as *sample_entropy* and *aproximation_entropy* achieve the highest valuation in importance. Interpretable characteristics such as *linearity*, *curvature*, *spike*, and *skewness* would occupy the following positions of importance. On the other hand, we can see two characteristics that have zero importance: *nperiods* and *seasonal_period*. Given the high number of datasets, a no-preprocessing of the data approach has been chosen, specifying a zero frequency for every time series. This causes the calculation of *nperiods* and *seasonal_periods* to always get the same value. In a real case, different parameters can be specified that would allow different values to be obtained in these characteristics. The previous characteristics are especially interesting in the field of time series, so we have decided to keep them in the CMFTS package.

We use a heat map to be able to analyze the importance of each feature on each dataset, Figure 6. As we can see in Figure 6, there are a lot of differences in the feature importance scores between different datasets. It means that each problem has very several characteristics and behaviors, so we need different features to extract the right information on each dataset. We can differentiate into two big groups of datasets. The first one which we need a small number of features to obtain the desired information. So, our models can obtain good enough results with this small subset of features, even if these results are not the best. The second one which our model uses a lot of features. In this case, it might be because the problem is very complex, and we need a lot of information to obtain good results. Or the features are not good enough to obtain the needed information to resolve the problem, and the model uses a lot of them trying to obtain good results. If we sort the datasets from Figure 6 in an increasing way based on the accumulated importance of the features, we can observe both groups in an easy way, Figure 7. At the top of the heat map, we can see the datasets in which our model uses a small set of features. At the bottom, we are able to see the datasets in which our proposal needs to use a lot of features. On the datasets in the order of Figure 7, if we calculate the difference between the best case of each dataset and our best model (CMFTS+RF), Figure 8, we see that this difference is lesser in the datasets at the top of Figure 7. That means that in the cases in which our model uses a small subset of features, it is able to obtain very close results to the best algorithm. These results reinforce the original idea of this proposal to obtain competitive results with simple and interpretable models.

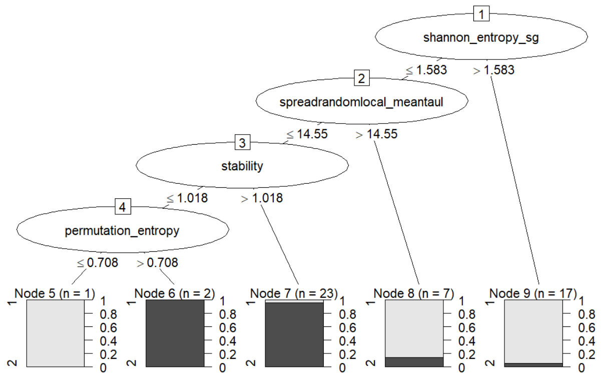
5 Conclusion

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

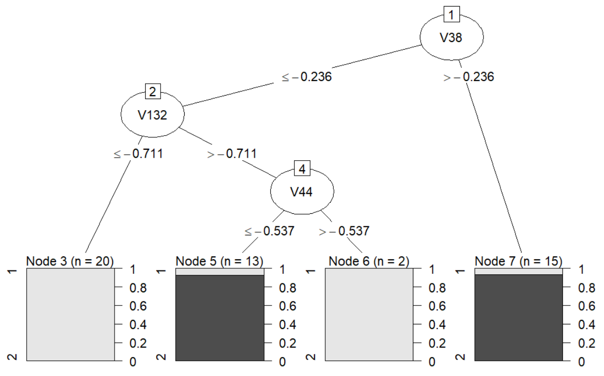
We have published our proposal software to make it accessible and usable for any practitioner or researcher to use. We have published all the results obtained throughout the work to make it fully reproducible. The functioning of our proposal has been tested on 112 datasets obtained from the UCR repository.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



(a) GunPoint classes example.



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



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

Figure 4: Interpretability GunPoint dataset example.

proposal obtains results statistically indistinguishable from those obtained by the best state-of-the-art algorithm. This result reinforces the original idea that feature-based methods require a larger number of time series to perform correctly.

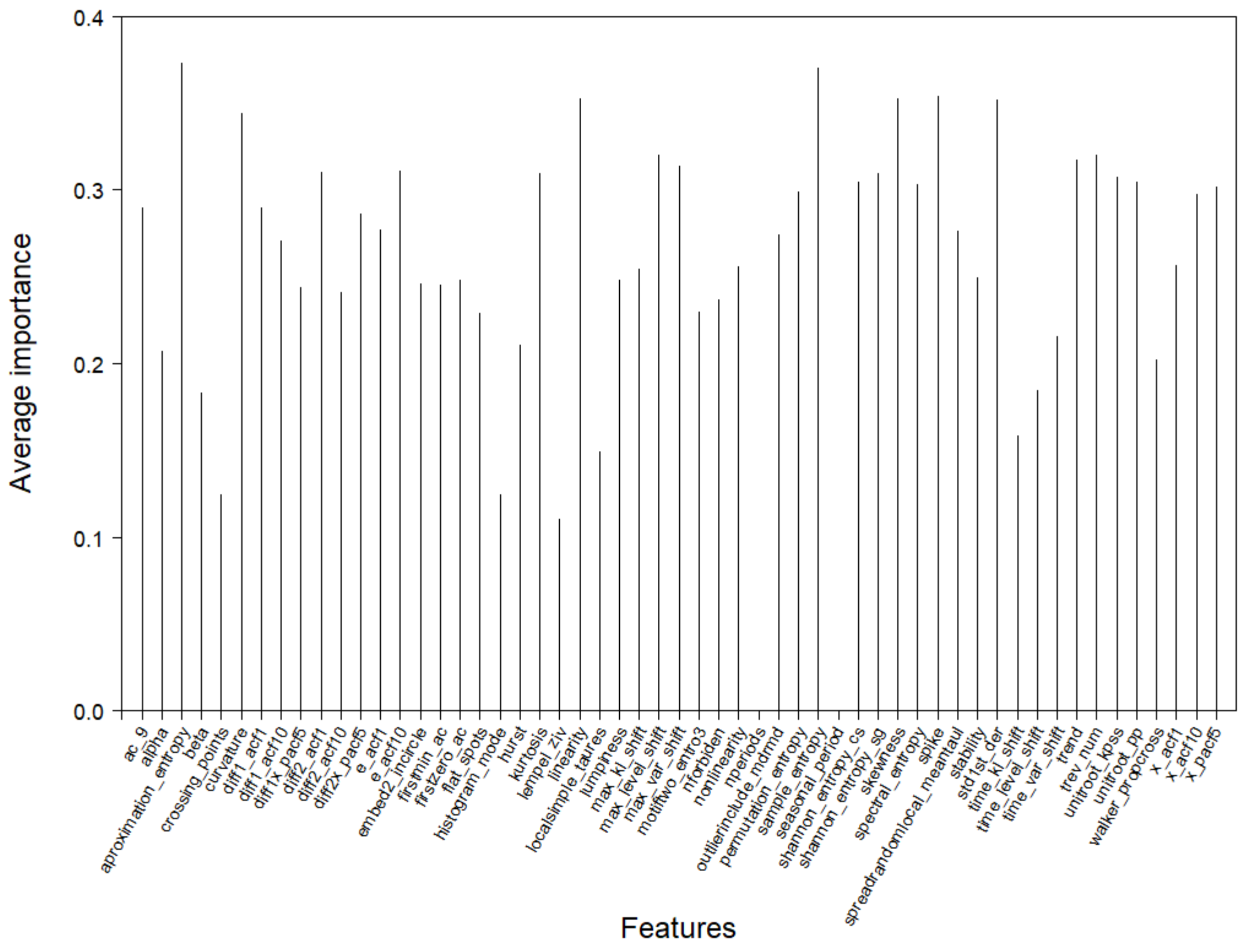


Figure 5: Average importance of features above all datasets.

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

6 Acknowledgment

This research has been partially funded by the following grants: TIN2013-47210-P and TIN2016-81113-R both from the Spanish Ministry of Economy and Competitiveness, and P12-TIC-2958 from Andalusian Regional Government, Spain. Francisco J. Baldán holds the FPI grant BES-2017-080137 from the Spanish Ministry of Economy and Competitiveness.

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

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

References

[1] J. Amigó, Permutation complexity in dynamical systems: ordinal patterns, permutation entropy and all that, Springer Science & Business Media, 2010.

[2] A. Bagnall, L. Davis, J. Hills, J. Lines, Transformation based ensembles for time series classification, in: Proceedings of the 2012 SIAM international conference on data mining, SIAM, 2012, pp. 307–318.

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

[4] A. Bagnall, G. Janacek, A run length transformation for discriminating between auto regressive time series, Journal of classification 31 (2) (2014) 154–178.

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

[6] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: the collective of transformation-based ensembles, IEEE Transactions on Knowledge and Data Engineering 27 (9) (2015) 2522–2535.

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

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

[9] C. Bandt, B. Pompe, Permutation entropy: a natural complexity measure for time series, Physical review letters 88 (17) (2002) 174102.

[10] Y. Bengio, L. Yao, G. Alain, P. Vincent, Generalized denoising auto-encoders as generative models, in: Advances in neural information processing systems, 2013, pp. 899–907.

[11] D. J. Berndt, J. Clifford, Using Dynamic Time Warping to Find Patterns in Time Series, in: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, AAAIWS'94, AAAI Press, 1994, pp. 359–370.

[12] A. Bondu, D. Gay, V. Lemaire, M. Boullé, E. Cervenka, FEARS: a Feature and Representation Selection approach for Time Series Classification, in: Asian Conference on Machine Learning, 2019, pp. 379–394.

[13] A. Bostrom, A. Bagnall, Binary shapelet transform for multiclass time series classification, in: Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII, Springer, 2017, pp. 24–46.

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

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

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

[17] S. Chauhan, L. Vig, S. Ahmad, ECG anomaly class identification using LSTM and error profile modeling, Computers in biology and medicine 109 (2019) 14–21.

[18] H. Chen, F. Tang, P. Tino, X. Yao, Model-based kernel for efficient time series analysis, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2013, pp. 392–400.

[19] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The UCR Time Series Classification Archive, `www.cs.ucr.edu/~eamonn/time_series_data/` (2015).

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

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

[22] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, E. Keogh, The UCR time series archive, IEEE/CAA Journal of Automatica Sinica 6 (6) (2019) 1293–1305.

[23] L. T. DeCarlo, On the meaning and use of kurtosis, Psychological methods 2 (3) (1997) 292.

[24] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.

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

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

[27] M. Flynn, J. Large, T. Bagnall, The contract random interval spectral ensemble (c-RISE): the effect of contracting a classifier on accuracy, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2019, pp. 381–392.

[28] B. D. Fulcher, Feature-based time-series analysis, in: Feature Engineering for Machine Learning and Data Analytics, CRC Press, 2018, pp. 87–116.

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

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

[31] N. Hatami, Y. Gavet, J. Debayle, Classification of time-series images using deep convolutional neural networks, in: Tenth international conference on machine vision (ICMV 2017), vol. 10696, International Society for Optics and Photonics, 2018, p. 106960Y.

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

[33] Y. Kang, R. J. Hyndman, F. Li, et al., Efficient generation of time series with diverse and controllable characteristics, Tech. rep., Monash University, Department of Econometrics and Business Statistics (2018).

[34] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Locally adaptive dimensionality reduction for indexing large time series databases, in: Proceedings of the 2001 ACM SIGMOD international conference on Management of data, 2001, pp. 151–162.

[35] J. Large, A. Bagnall, S. Malinowski, R. Tavenard, On time series classification with dictionary-based classifiers, Intelligent Data Analysis 23 (5) (2019) 1073–1089.

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

[37] J. Lin, E. Keogh, L. Wei, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, Data Mining and knowledge discovery 15 (2) (2007) 107–144.

[38] J. Lin, R. Khade, Y. Li, Rotation-invariant similarity in time series using bag-of-patterns representation, Journal of Intelligent Information Systems 39 (2) (2012) 287–315.

[39] J. Lin, Y. Li, Finding structural similarity in time series data using bag-of-patterns representation, in: International Conference on Scientific and Statistical Database Management, Springer, 2009, pp. 461–477.

[40] J. Lines, L. M. Davis, J. Hills, A. Bagnall, A shapelet transform for time series classification, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2012, pp. 289–297.

[41] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, N. S. Jones, catch22: CAnonical Time-series CHaracteristics, CoRR abs/1901.10200.

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

[43] A. Mueen, E. Keogh, N. Young, Logical-shapelets: an expressive primitive for time series classification, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2011, pp. 1154–1162.

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

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

[46] A. R. S. Parmezan, G. E. Batista, A study of the use of complexity measures in the similarity search process adopted by knn algorithm for time series prediction, in: Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on, IEEE, 2015, pp. 45–51.

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

[48] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[49] D. Rajan, J. J. Thiagarajan, A generative modeling approach to limited channel ECG classification, in: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, 2018, pp. 2571–2574.

[50] T. Rakthanmanon, 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.

[51] C. A. Ratanamahatana, E. Keogh, Making time-series classification more accurate using learned constraints, in: Proceedings of the 2004 SIAM International Conference on Data Mining, SIAM, 2004, pp. 11–22.

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

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

[54] P. Schäfer, 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.

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

[56] P. Senin, 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.

[57] W. Song, L. Liu, M. Liu, W. Wang, X. Wang, Y. Song, Representation learning with deconvolution for multivariate time series classification and visualization, in: International Conference of Pioneering Computer Scientists, Engineers and Educators, Springer, 2020, pp. 310–326.

[58] T. M. Therneau, E. J. Atkinson, et al., An introduction to recursive partitioning using the RPART routines (1997).

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

[60] X. Wang, K. Smith, R. Hyndman, Characteristic-based clustering for time series data, Data mining and knowledge Discovery 13 (3) (2006) 335–364.

[61] Z. Wang, T. Oates, Spatially encoding temporal correlations to classify temporal data using convolutional neural networks, arXiv preprint arXiv:1509.07481.

[62] Z. Wang, W. Yan, 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.

[63] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 947–956.

[64] A. Zhang, B. Yang, L. Huang, Feature Extraction of EEG Signals Using Power Spectral Entropy, in: 2008 International Conference on BioMedical Engineering and Informatics, vol. 2, 2008, pp. 435–439.

[65] R. Zhou, C. Yang, J. Wan, W. Zhang, B. Guan, N. Xiong, Measuring complexity and predictability of time series with flexible multiscale entropy for sensor networks, Sensors 17 (4) (2017) 787.

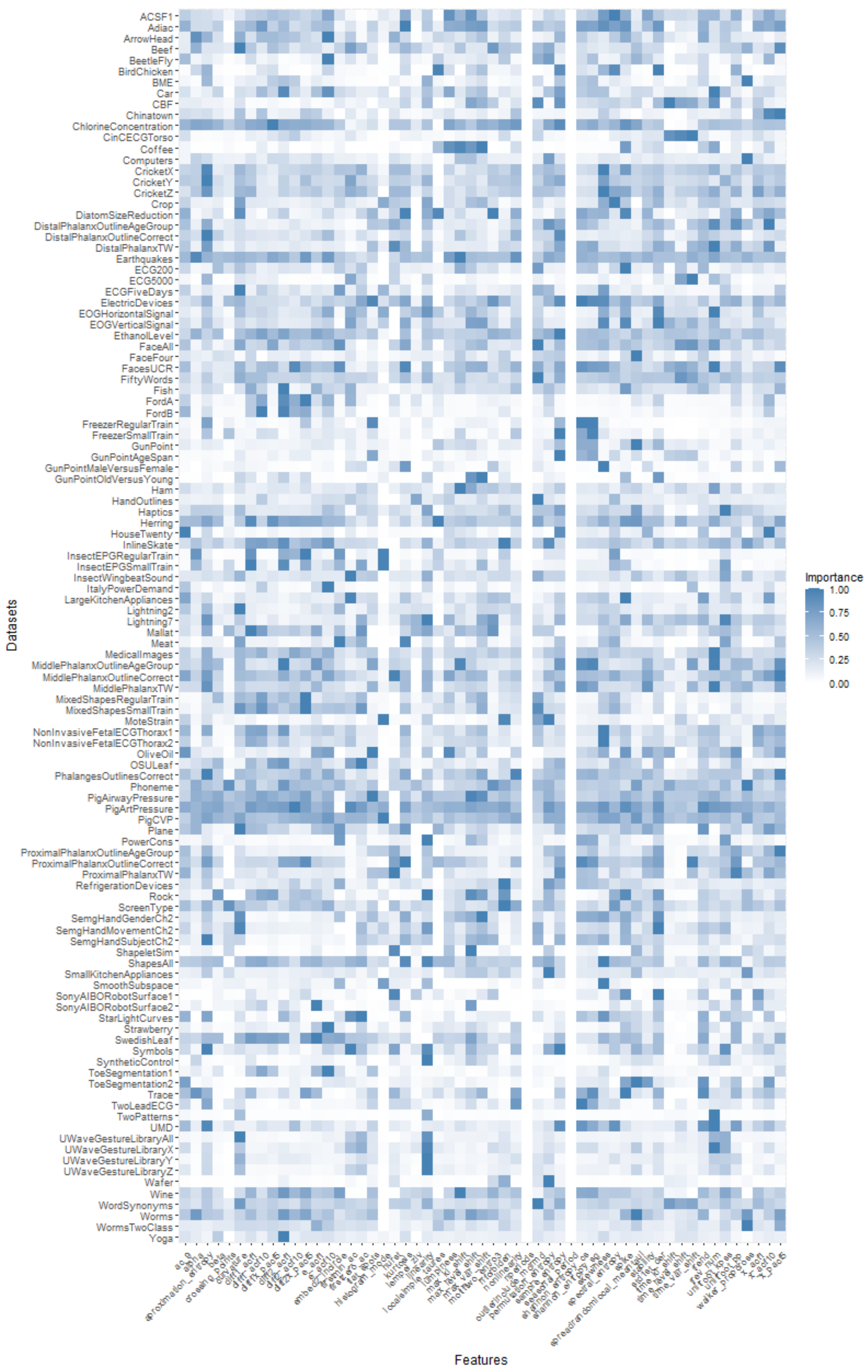


Figure 6: Heat map of the importance of characteristics by dataset.

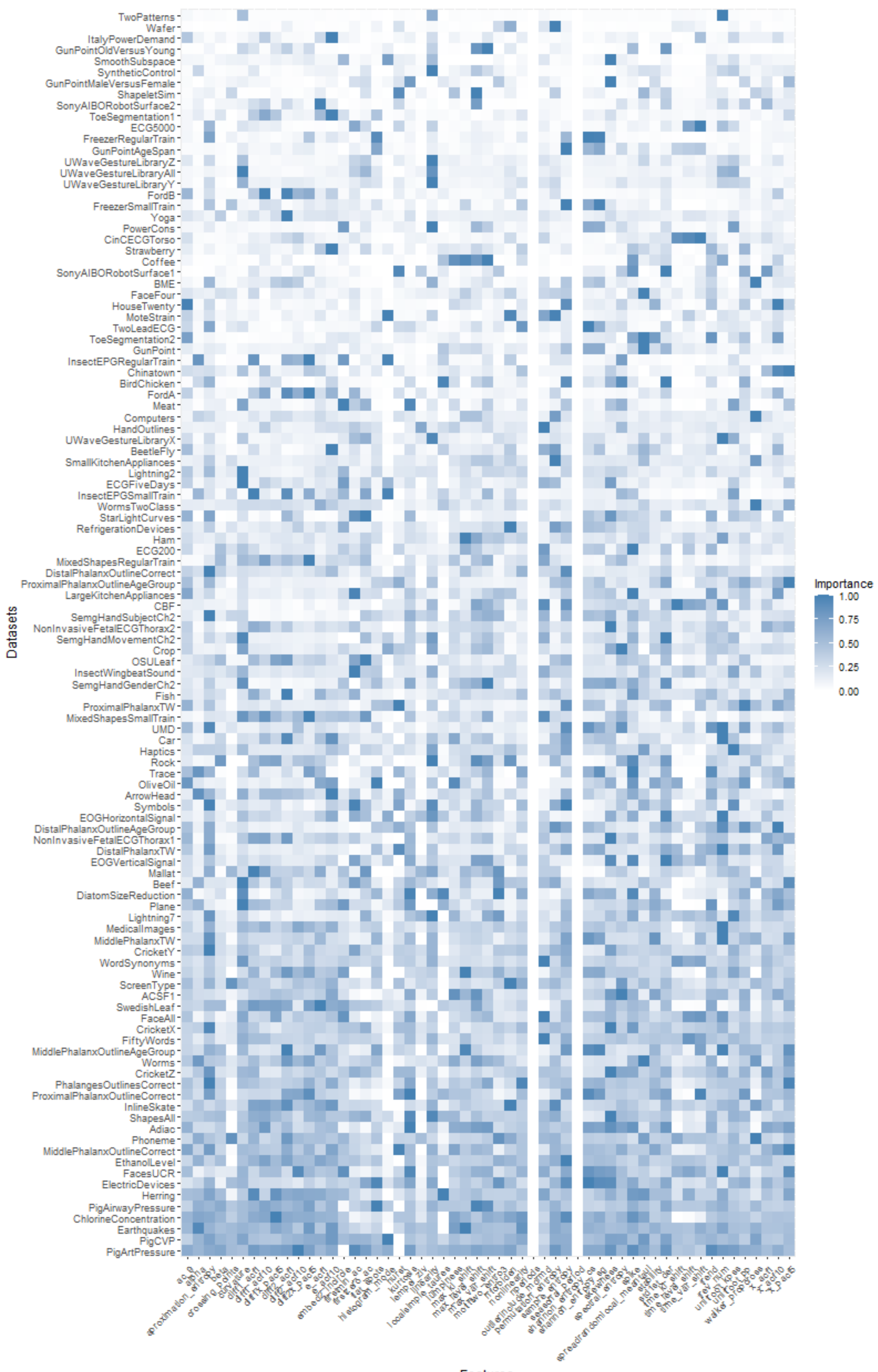


Figure 7: Heat map of the importance of characteristics by dataset, sorted by accumulated importance.

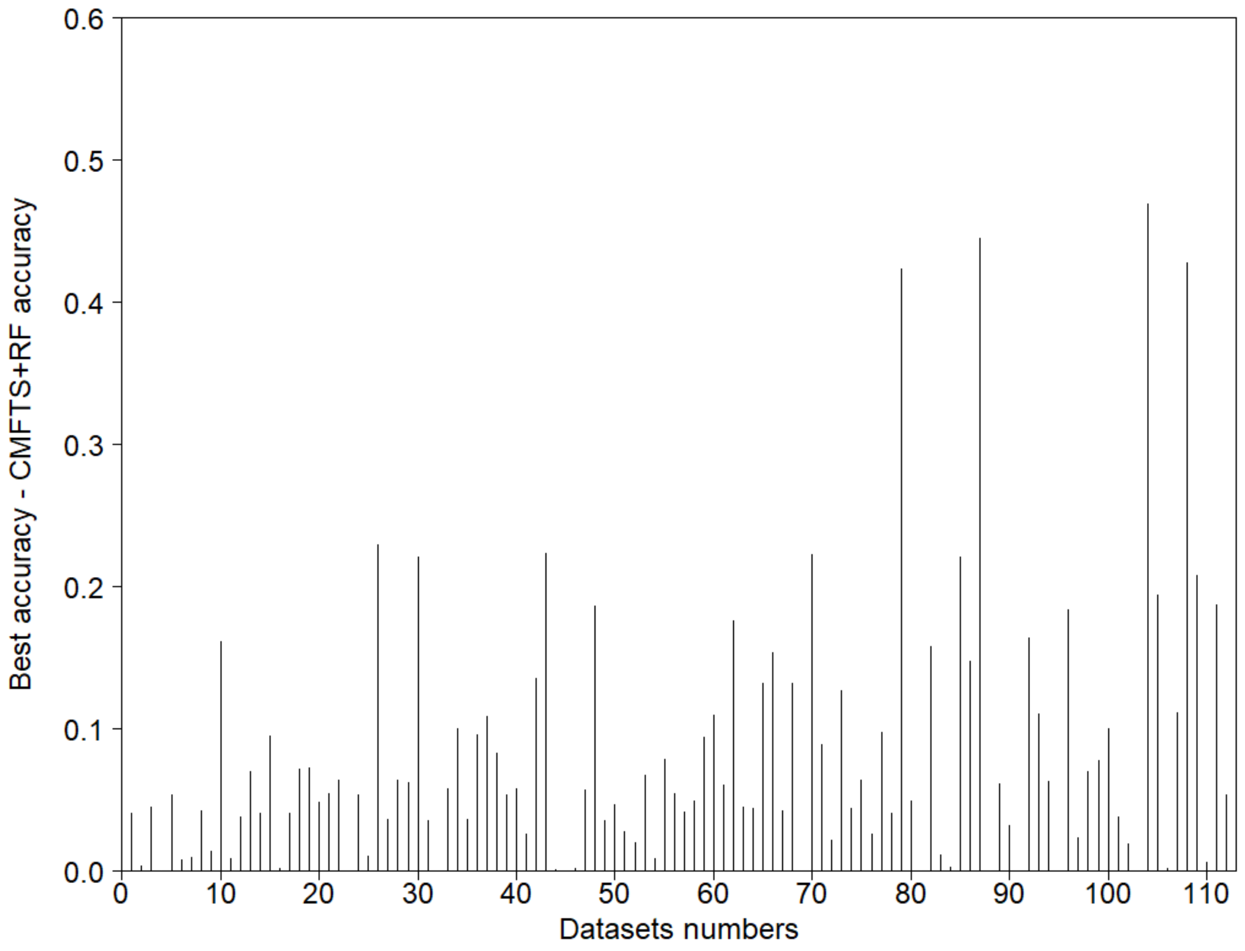


Figure 8: Accuracy differences between CMFTS+RF and the best algorithm on each dataset. The datasets are sorted like Figure 7.

3 Multivariate times series classification through an interpretable representation

- F. J. Baldán, J. M. Benítez. Multivariate times series classification through an interpretable representation. Information Sciences, 569, 596-614 (2021).
  - Status: **Published**.
  - Impact Factor (JCR 2020): **6.795**
  - Subject Category: **Computer Science, Information Systems**
  - Rank: **18/162**
  - Quartile: **Q1**

Multivariate times series classification through an interpretable representation

Francisco J. Baldán[*]
Department of Computer Science
and Artificial Intelligence
University of Granada, DICITS, iMUDS, DaSCI
Granada, Spain, 18071
`fjbaldan@decsai.ugr.es`

José M. Benítez Sánchez
Department of Computer Science
and Artificial Intelligence
University of Granada, DICITS, iMUDS, DaSCI
Granada, Spain, 18071
`J.M.Benitez@decsai.ugr.es`

Abstract

Nowadays the classification of multivariate time series (MTSC) is a task with increasing importance due to the proliferation of new problems in various fields (economy, health, energy, transport, crops, etc.) where a large number of information sources are available. Direct extrapolation from methods that traditionally worked in univariate environments cannot frequently be applied to obtain the best results in multivariate problems. This is mainly due to the inability of these methods to capture the relationships between the different variables that conform a multivariate time series. The multivariate proposals published so far now offer competitive results but are hard to interpret. In this paper we propose a time series classification method that considers an alternative representation of time series through a set of descriptive features taking into account the relationships between the different variables of a multivariate time series. Then applying traditional classification algorithms interpretable while still competitive results can be obtained.

Keywords Multivariable · Time series features · Complexity measures · Time series interpretation · Classification

1 Introduction

Nowadays, large amounts of data are generated. Everything is increasingly interconnected, more and more sensors are included in everything around us, and these monitor the behavior of any event of interest over time. These sensors generate lots of data as multivariate time series (MTS). A key task in the analysis and mining of these data is multivariate time series classification (MTSC), which aims to give an accurate response to a large number of problems: e.g. from detecting when a patient is sick or has an anomaly in his heart behavior [28], or if a driver is in optimal condition to drive [26], the recognition of human activities [37], the occupation of an office room based on environmental information [12], the wind speed forecasting [35] or how to adapt energy production based on particular circumstances [24].

The field of MTSC can be divided into two main types of work. Firstly, applied works that seek to obtain a better solution for a given problem, offering ad-hoc proposals considering the peculiarities of the treated problem [11][25][31]. Secondly, proposals that deal with MTS in a general way but taking into account possible interrelations between the different variables available [2][17][33][43]. The proposals in the later group are usually based on strong theoretical foundations. A relatively large number of proposals for MTSC can be found in the literature [7][16][19][36]. Most of them are guided towards obtaining increasing levels of accuracy. However, eXplainable Artificial Intelligence (XAI) [14] is a topic enjoying a growing level of interest. Its goal is to build accurate intelligent system for complex tasks, but also paying special attention to their interpretability. The built systems or the way they make decisions are required to be easy to understand for human beings. Thus top accuracy is no longer the only objective and interpretability receives higher attention. This also applies to solutions for classification problems.

[*]Corresponding author.

In the field of MTSC, there are few proposals that pay attention to the interpretability of results [18]. Given the complexity of the problem, most proposals are focused on obtaining the best results in terms of accuracy. Even the proposals based on shapelets [4][42], which are interpretable from their univariate origins, have chosen to use the Transformed Shapelets in multivariate environments [9] or proposals that are even less interpretable [20], giving priority to accuracy results over interpretability. One possible way to pave the path towards easier to understand solutions to MTSC is expressing time series in different domains. Perhaps in terms of descriptive features instead of the raw time domain values.

In this paper, we present a new MTSC approach based on the representation of time series through a set of features and measures. This approach allows transforming the original MTSC problem into a traditional classification problem, enabling to apply the whole set of the traditional classification algorithms. The proposal is mainly focused on obtaining interpretable classifiers, so that, an end user can understand and feel more confident with the obtained systems. In addition, the approach allows to obtain acceptable accuracy results with respect to the main techniques of the state-of-the-art.

The remainder of this paper is organized as follows: Section 2 introduces the state of the art in MTSC. Section 3 describes in depth our proposal. Section 4 shows the experimental study conducted, and the results obtained. Section 5 discusses the interpretability of our proposal. Finally, Section 6 concludes the paper.

2 Related work

In the field of MTSC, proposals from methods that have demonstrated good behavior in univariate cases predominate. Some of the first proposals for MTSC were multivariate extensions of the distance-based algorithm 1NN-DTW [22][40], given its simplicity and good results in univariate environments. These proposals are a good starting point, but they carry the limitations they already had in univariate environments, such as high computational complexity and low interpretability, since they only indicate how much the instances are similar to each other. To these limitations, we must add that in a multivariable environment, the first proposals of 1NN-DTW processed each variable of each time series independently, so they were not able to extract information from the relationship between the different variables that make up each multivariate time series. With this in mind, we can say that multiple proposals for a multivariate DTW have been made, such as dependent (DTW_D) and independent (DTW_I) warping, both having the same performance [38]. Other proposals such as Mahalanobis Distance-based Dynamic Time Warping measure (MDDTW) [32] seek to give a general answer to this problem. MDDTW is able to precisely calculate the relationship between the different variables that compose an MTS. This, together with the alignment obtained by DTW, allows obtaining very competitive results.

The feature-based approach has multiple proposals, giving special importance to the extraction of additional information and to the speed of processing, especially when compared to similarity-based techniques. In this field we can differentiate between proposals based on shapelets and bag-of-words. In the field of shapelets, Generalized Random Shapelets Forests (gRSF) [21] is considered the state-of-the-art, obtaining better results than its direct competitor, Ultra Fast Shapelets (UFS) [41]. gRSF is based on the creation of a set of shapelet-based decision trees from a random extraction of the shapelets. In the field of bag-of-words, Word ExtrAction for time Series cLassification plus Multivariate Unsupervised Symbols and dErivatives (WEASEL+MUSE) [36] is considered the state-of-the-art, as it obtains the best results against its direct competitors: Learned Pattern Similarity (LPS) [8], Autoregressive forests for multivariate time series modelling (mv-ARF) [39], Symbolic representation for Multivariate Time Series classification (SMTS) [7], and gRSF. All of them have been tested on one of the first reference MTS database collected from [6], with a total of 20 MTSC datasets. WEASEL+MUSE extracts a vector of features by applying a sliding-window to each variable of the MTSC and filtering out non-discriminative features, finally a classifier analyses these data.

In the field of deep learning, the extension of the Long Short Term Memory Fully Convolutional Network (LSTM-FCN) and Attention LSTM-FCN (ALSTM-FCN) [19] to a multivariate environment, including a squeeze-and-excitation block in the fully convolutional block that improves accuracy. This proposal improved the WEASEL+MUSE results over the original database of 20 datasets [6] extended with 10 datasets from the UC Irvine Machine Learning Repository (UCI) [15] and 6 datasets used by Pei et al. [34]. Also, we find proposals that pay attention to detect natural features of time series such as trend [27].

A new proposal has recently emerged, Local Cascade Ensemble for Multivariate Data Classification (LCE) and its extension for Multivariate Time Series (LCEM) [16]. LCE and LCEM are a hybrid ensemble method with 2 major objectives. The first one is to handle the bias-variance tradeoff by an explicit boosting-bagging approach. The second one is to individualize classifier errors on different parts of the training data by an implicit divide-and-conquer approach. This proposal is outlined as the new state-of-the-art in MTSC by obtaining better results than the previous state-of-the-art MLSTM-FCN and WEASEL+MUSE, on the University of East Anglia (UEA) repository [3], a new repository for MTSC composed of 30 datasets that is becoming increasingly important.

In contrast to state-of-the-art methods, we propose a method that obtains essential features of each variable and each MTS and applies a transformation to the MTS dataset, obtaining a traditional classification problem based on attributes. All traditional classification algorithms can be applied to this new dataset, and depending on the applied algorithms, interpretable results can be obtained to explain the problem or results of higher accuracy.

3 Multivariable times series classification through an interpretable representation

In this work we propose a method that allows the calculation of complexity measures to be applied to MTSC problems. Our proposal, namely Complexity Measures and Features for Multivariate Time Series (CMFMTS), is based on the idea that a time series can be faithfully represented with a set of complexity measures and descriptive features [5]. Furthermore, these features preserve most of the information content of the series to such an extend that they can be used to classify the series.

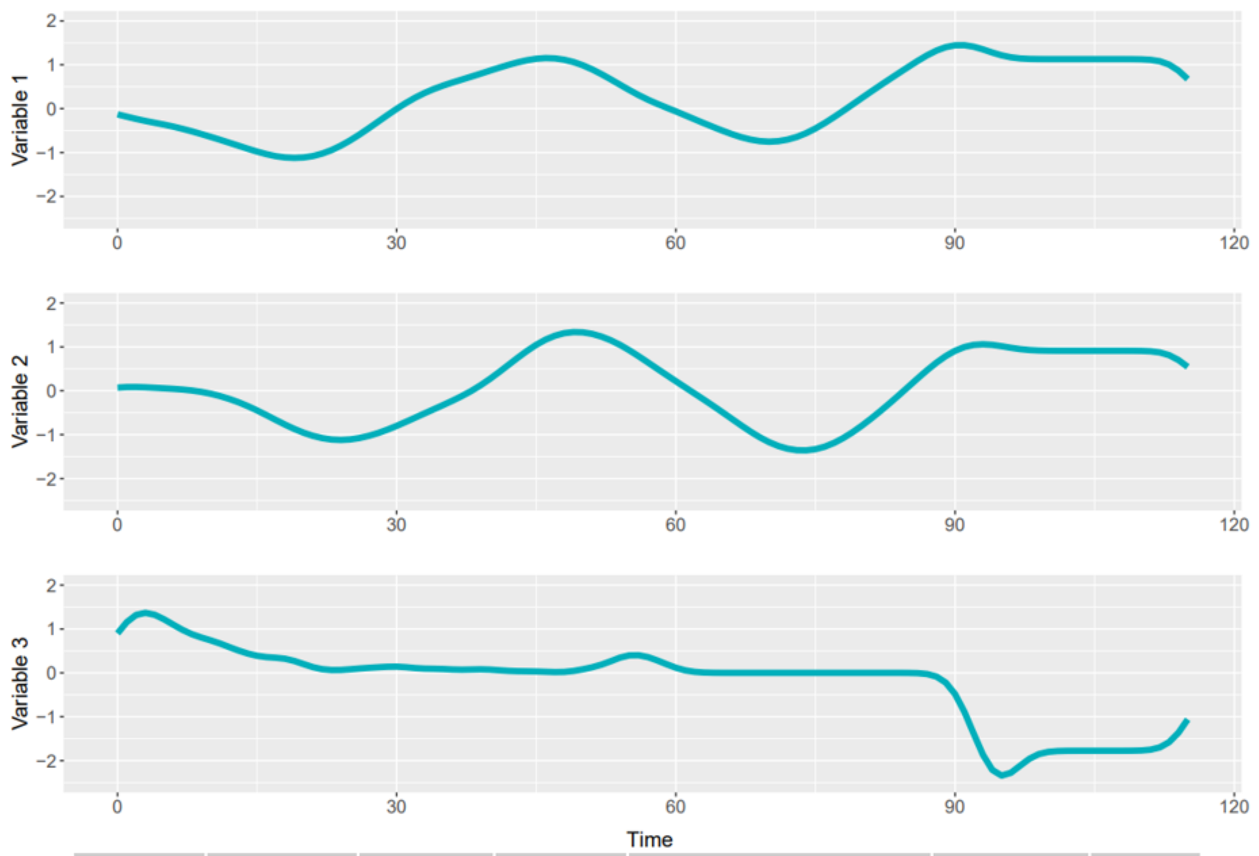
The following is an example of the calculation of some features on an MTS with three variables. In Table 1, we show some features highly related to the nature of the time series and its range of possible values. In Figure 1, we show a simple example of the feature computation used and its interpretability. In the first place, we can see how variables 1 and 2 are similar, so we can expect values of the features also similar to each other. This is reflected in the values of kurtosis and skewness. If variables 1 and 2 have similar values their probability distribution will be similar and therefore their values of kurtosis and skewness. We can appreciate a significant difference concerning variable 3. In the three variables, we can see the existence of a single trend, for this reason, the trend values are close to 1 in all cases. The oscillations present in the variables 1 and 2 seem more typical of seasonal patterns that do not affect the trend of the time series. To evaluate the Chao-Shen shannon entropy (shannon_entropy_cs) we have to appreciate the evolution of variables 1, 2, and 3. Variables 1 and 2 show a certain pattern, while variable 3 shows a long period without changes with a final reduction of the value never seen before. For this reason, it obtains a higher value in shannon entropy very far from the one obtained by variables 1 and 2. We also analyze the values of curvature and linearity. Given the evolution and shape of the three variables and the perceptible linear relationship between the current values of variables 1 and 2 with their corresponding past values, it is logical to expect positive and similar values of curvature and linearity for variables 1 and 2. On the other hand, variable 3 does not show these forms or a linear relationship between its present and past values, so it obtains negative values that are far from curvature and linearity concerning what is obtained by variables 1 and 2.

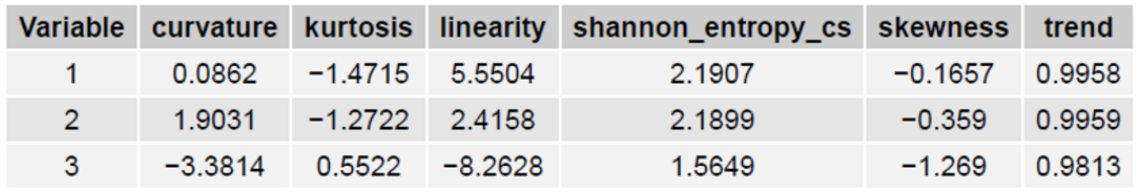
Table 1: Example of some features used.

Char.	Name	Description	Range
F_1	curvature	Calculated based on the coefficients of an orthogonal quadratic regression	$(-\infty, \infty)$
F_2	kurtosis	The “tailedness" of the probability distribution	$(-\infty, \infty)$
F_3	linearity	Calculated based on the coefficients of an orthogonal quadratic regression	$(-\infty, \infty)$
F_4	shannon_entropy_cs	Chao-Shen entropy estimator	$[0, \infty)$
F_5	skewness	Asymmetry of the probability distribution	$(-\infty, \infty)$
F_6	trend	Strength of trend	$[0, 1]$

Figure 2 shows the workflow of our proposal. First, a set of n multivariate time series is assumed, each consisting of m variables (Figure 2.1). Individually, each one of the variables that compose each time series is processed, obtaining the j features for each variable (Figure 2.2). A dataset is obtained with $n \times m$ rows and j columns, where each row is composed of the set of features processed on each time series that compose each MTS (Figure 2.3). Finally, this dataset is processed by placing all the variables of the same MTS in the same row (Figure 2.4). In this way, a new (transformed) dataset is obtained where all the features of all the variables that compose the same MTS are placed in the same row, forming part of the same instance. This enables the search for patterns and relationships of interest among the different variables that compose the same MTS (Figure 2.5).

Although the feature calculation based approach can be applied to all types of automatic learning problems as well as supervised, unsupervised, semi-supervised learning, etc. In the supervised case, simple and fast comparisons can be made with respect to the main state-of-the-art algorithms. Due to the great variety of the processed time series, it is possible that undesired values are obtained for some of the proposed features. For example, to calculate the autocorrelation coefficient function (ACF) [5] concerning the values delayed 10 instants of time it is necessary that our time series has a minimum length of 11, otherwise, we will obtain an Not Available (NA). Time series with a single value are another problematic case since features like kurtosis and skewness are not defined for these cases and would return Not a Number (NaN) values. Time series containing NA generate problems internally in some of the features used





Variable	curvature	kurtosis	linearity	shannon_entropy_cs	skewness	trend
1	0.0862	−1.4715	5.5504	2.1907	−0.1657	0.9958
2	1.9031	−1.2722	2.4158	2.1899	−0.359	0.9959
3	−3.3814	0.5522	−8.2628	1.5649	−1.269	0.9813

Figure 1: Example of feature extraction from an MTS with 3 variables.

(acf, kurtosis, skewness, shannon_entropy_cs, etc.) returning NA values in those features. Finally, there are features that can obtain values in the range $(-\infty, \infty)$. Extreme values close to the limits are considered as undesirable since they generate several problems in the different algorithms applied later. To deal with these cases, we have specified a preprocessing stage, following the calculation of the features and their correct ordering, which solves the possible inconveniences generated by these cases. The whole process is depicted in Algorithm 1.

The starting point is the calculation of the proposed features in the training and test sets (Line 1). For any of the cases mentioned above in which an undesired value has been obtained, these values are unified under a single NA identifier (Lines 2-4). We check on the training set if any column lacks interest because it is full of undesired values. If so, this column is removed from both the training set and the test set (Lines 5-10). In order to simplify the treatment of missing values, we have chosen to impute these values with the average of their respective column (Lines 11-15). There are better imputation techniques, but we do not address that task in this paper and the considered one has proved to be effective enough. To avoid the use of variables without information, we analyzed the training set looking for variables with a single value. If any variable with this condition is found, it is eliminated from both the training set and the test set (Lines 16-21). Finally, each of the specified models is processed, obtaining the desired model fit, its prediction on the test set and the accuracy achieved (Lines 23-28). These data are returned to the user, together with the datasets transformed to the features of our proposal (Line 29).

Finally, once we have explained our proposal and its application in a real environment, we can list the main advantages offered by this approach:

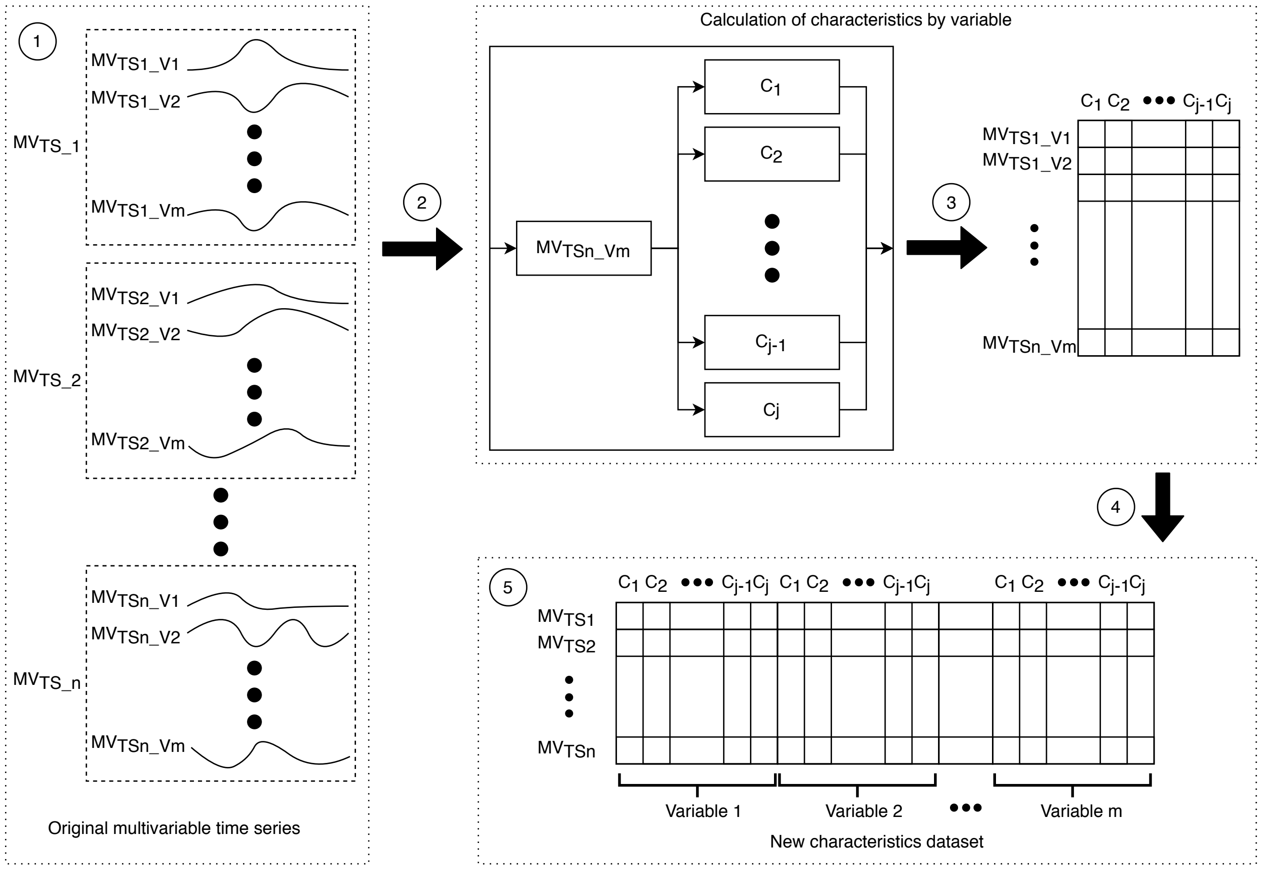


Figure 2: Features calculation workflow.

- Allows the use of the application of any vector-based classification method, since after the applied transformation, we obtain a traditional dataset where each instance is represented by its corresponding attributes (features).
- Allows the use of machine learning methods based on different paradigms: supervised, semi-supervised, self-supervised, unsupervised, etc., since it obtains a vector-based dataset, where each instance is composed of different attributes.
- Handles easily datasets of time series with varying lengths, as it processes each time series individually.
- Decisions made can be easily understood by human experts, since the features used explain the behavior of the time series. In addition, the represented concepts by the selected features are interpretable by the users.

3.1 Computational Complexity

Our proposal is composed of two main steps: the calculation of the feature value set and the construction of the classifier. Since the steps have to be executed sequentially, the complexity of the whole process can be computed by simply adding the complexity of each one. For the sake of convenience, the variables used in the formulas are: f, number of features computed; n, number of time series in a dataset; v, the number of variables in a multivariate time series; and l, the length of the time series.

The computing of the feature values is defined as a sequential process, each feature computing at one time, although if enough threads are available, all of them can be computed in parallel. Thus the overall complexity, stated in O notation, is equal to the feature with the highest complexity. The computation of the complexity of all of them is quite straightforward. Two of them have the greatest complexity, namely approximation entropy and sample entropy. For a univariate time series, those features have a computational time complexity of O(l^2) [29][30], each one. Based on the feature computation process shown in Figure 2, we can conclude that our proposal has a computational complexity of O$(n \cdot v \cdot f \cdot l^2)$. As for the classification models used in our proposal, each one has a different computational complexity. In Table 2, we show the computational complexity of typical models. The final computational complexity

Algorithm 1 Preprocessing procedure

```
Input:
    $train$: train dataframe with (Ts_id, Ts_dimId, Ts_class, Ts_values)
    $test$: test dataframe with (Ts_id, Ts_dimId, Ts_class, Ts_values)
    $models$: list of models to be processed
Output:
    $output\_data$: a triplet that contains the fitted models, the vectors with the predicted labels and the
    accuracies obtained
    $mvf\_train$: features train dataframe
    $mvf\_test$: features test dataframe
1: mvf_train, mvf_test ← calc_mvcmfts((train, test), all)
2: for each value in (mvf_train, mvf_test) do
3:     if (is.na(value) || is.nan(value) || is.infinite(value)) then value ← NA end if
4: end for
5: for each column in mvf_train do
6:     if sum(!is.na(colum.values)) == 0) then
7:         mvf_train.delete(column)
8:         mvf_test.delete(column)
9:     end if
10: end for
11: for each column in (mvf_train, mvf_test) do
12:     for each value in column do
13:         if is.na(value) then value ← mean(column) end if
14:     end for
15: end for
16: for each column in mvf_train do
17:     if (length(unique(column)) <= 1) then
18:         mvf_train.delete(column.index)
19:         mvf_test.delete(column.index)
20:     end if
21: end for
22: output_data ← NULL
23: for each model in models do
24:     fit ← train.model(mvf_train, train.Ts_class)
25:     pred ← fit.predict(mvf_test)
26:     acc ← accuracy(pred, test.Ts_class)
27:     output_data.add(fit, pred, acc)
28: end for
29: return (output_data, mvf_train, mvf_test)
```

Table 2: Computational complexity of typical models. Notation: n, number of samples, t, number of randomized trees; k number of attributes randomly included at each node; h, height of a tree; m, number of attributes; c, number of classes.

Models	Time complexity
Random Forest	O$(0.632 \cdot n \cdot t \cdot k \cdot \log(0.632 \cdot n))$ [4]
C5.0 with Boosting	O$(h \cdot m \cdot (n \cdot c + n \cdot \log(n))$ [23]
Support Vector Machine	O(n^3) [1]
1-Nearest Neighbor	O$(n \cdot v \cdot f)$

of our proposal results from the addition of the feature computation complexity, seen above, and the complexity of the model used, as shown in Table 2.

The computational complexity of our proposal is very similar to the one offered by the main state-of-the-art algorithm. LCEM has a computational complexity O$(N \cdot s \cdot d \cdot D \cdot 2^D \cdot T_{Base})$, where d is the number of attributes, d' is the number of attributes in RF subset of attributes, D is the maximum depth of a tree, s is the number of samples, N is the number of trees, and T_{Base} is the time complexity of the base classifier. To this complexity, we must add the complexity of the applied transformation, which linearly grows in complexity with the number of samples.

One of the main advantages of our proposal is that the time complexity of the feature extraction process scales linearly with the number of time series to be processed and is trivial to parallelize. In addition, we have not considered hyper-parameter optimization for feature computation, unlike LCEM proposal or the models used, so the procedure is kept simple and does not increase the computational complexity. Furthermore, the feature extraction process is applied independently to each multivariate time series, so we can compute these characteristics directly as the time series are received, and we do not need to have a complete set of time series to start processing them. Again, the process can be trivially parallelized. This allows us, in the best case, to reduce the computational complexity of our proposal to the complexity of the classification model used.

4 Empirical Study

To assess the effectiveness of our proposal, we have developed a detailed empirical study. We start by explaining the experimental design (Section 4.1), followed by the results obtained (Section 4.2). The analysis of the interpretability of the models is important enough so that a complete section is devoted to them, namely Section 5.

4.1 Experimental Design

We describe the performance measures used to evaluate our proposal (Section 4.1.1), followed by the datasets used (Section 4.1.2) and the machine learning models selected (Section 4.1.3). Finally, we describe the hardware and software used in the development of our proposal (Section 4.1.4).

4.1.1 Performance measures

Since the datasets come from very different fields, we have opted for a ranking performance measure. We have selected the average rank as a comparative method from the accuracy calculation on the original training and test sets. The accuracy has been calculated as the number of instances correctly classified divided by the total number of instances of the test set. To obtain robust results from a statistical point of view, we have executed each experiment 100 times, —using different random seeds—, and calculated the mean of the obtained values. Also, we have included the Win/Loss/Tie ratio to quantify the number of cases in which each model and approach wins, loses, or ties concerning the best case. Since the range of possible results is wide, we have opted to include a Critical Difference diagram (CD) [13], and the Wilcoxon Signed-Rank test. CD shows the results of a statistical comparison between all models in pairs based on average ranks. Models that are connected by a bold line do not have a statistically significant difference for a particular confidence level. The Wilcoxon Signed-Rank test allows us to assess whether two models offer statistically distinguishable results at a particular confidence level, compared between all models in pairs based on the accuracies. In our case, we have set an α of 0.05 for a 95% confidence level. The average rank and the CD were obtained using the R *scmamp* package. The Wilcoxon Signed-Rank test results were obtained using the R *stats* package.

4.1.2 Datasets

To evaluate the performance of our proposal on problems of all kinds, we have selected the main repository of MTSC problems, the UEA multivariate time series classification archive. In Table 3, we show the characteristics of the 30 datasets of the UEA repository: number of instances of the training and test sets, length of the time series, number of variables of each MTS, and number of classes. Some of these datasets are composed of time series of different lengths, so the repository chose to pad with NA values. In our case, we have removed those values. We have processed the values of the time series that contain information without affecting the original values of each time series.

4.1.3 Models

For our proposal, we have selected a set of traditional models with two main approaches: to obtain interpretable results and to obtain the best classification results by sacrificing interpretability [5]. These models are C5.0 with boosting (C5.0B), Random Forest (RF), Support Vector Machine (SVM), and 1-Nearest Neighbors with Euclidean Distance (1NN-ED). For this last model, we have applied a normalization between [0, 1]. This set of models will be applied to the set of time series features obtained by our proposal. The final models of our proposal are obtained from the union of the transformed datasets with the four models previously commented. These proposals are: CMFMTS+C5B, CMFMTS+RF, CMFMTS+SVM, and CMFMTS+1NN-ED. We have simplified the CMFMTS nomenclature by CMFM due to space limitations in later tables. On the other hand, we have selected the main state-of-the-art MTSC models:

- 1-Nearest Neighbor classifier with Euclidean distance (1NN-ED), with and without normalization.
- 1-Nearest Neighbor classifier based on multi-dimensional points (DTW-1NN-D) [38], with and without normalization.
- 1-Nearest Neighbor classifier based on the sum of DTW distance for each dimension (DTW-1NN-I) [38], with and without normalization.

Table 3: Datasets information from the UEA repository.

Dataset	Train	Test	Length	Dims	Class
ArticularyWordRecognition	275	300	144	9	25
AtrialFibrillation	15	15	640	2	3
BasicMotions	40	40	100	6	4
CharacterTrajectories	1422	1436	60-182	3	20
Cricket	108	72	1197	6	12
DuckDuckGeese	50	50	270	1345	5
EigenWorms	128	131	17984	6	5
Epilepsy	137	138	206	3	4
ERing	30	270	65	4	6
EthanolConcentration	261	263	1751	3	4
FaceDetection	5890	3524	62	144	2
FingerMovements	316	100	50	28	2
HandMovementDirection	160	74	400	10	4
Handwriting	150	850	152	3	26
Heartbeat	204	205	405	61	2
InsectWingbeat	25000	25000	2-22	200	10
JapaneseVowels	270	370	7-29	12	9
Libras	180	180	45	2	15
LSST	2459	2466	36	6	14
MotorImagery	278	100	3000	64	2
NATOPS	180	180	51	24	6
PenDigits	7494	3498	8	2	10
PEMS-SF	267	173	144	963	7
PhonemeSpectra	3315	3353	217	11	39
RacketSports	151	152	30	6	4
SelfRegulationSCP1	268	293	896	6	2
SelfRegulationSCP2	200	180	1152	7	2
SpokenArabicDigits	6599	2199	4-93	13	10
StandWalkJump	12	15	2500	4	3
UWaveGestureLibrary	120	320	315	3	8

- Multivariate LSTM Fully Convolutional Networks for Time Series Classification (MLSTM-FCN) [19] with the settings specified by their authors: 128-256-128 filters, 250 training epochs, a dropout of 0.8, and a batchsize of 128.
- Word ExtrAction for time SEries cLassification plus Multivariate Unsupervised Symbols and dErivatives (WEASEL+MUSE) [36] with the settings specified by their authors: SFA word lengths l in [2,4,6], windows length in [4:max(MTSlength)], chi=2, bias=1, p=0.1, c=5 and a solver equals to L2R LR DUAL.
- Local Cascade Ensemble for Multivariate data classification (LCEM) [16], optimized hyper-parameters for each dataset (Windows (%), Trees, and Depth). The results have been obtained from the published work of the authors.
- Random Forest for Multivariate (RFM) algorithm, from the sklearn library, applied to the transformation proposed in the LCEM paper [16].
- Extreme Gradient Boosting for multivariate (XGBM), Extreme Gradient Boosting algorithm, from the xgboost library, applied to the transformation proposed in the LCEM paper [16].

The results of the algorithms mentioned above have been obtained from [16].

4.1.4 Hardware and Software

The experimentation carried out in this work was performed in a server with the following hardware: 4 $*$ Intel(R) Xeon(R) CPU E5-4620 0 @ 2.20GHz processors, 8 cores per processor with HyperThreading, 10 TB HDD, 512 GB RAM. The server software configuration comprises Ubuntu 18.04 and R 3.6.3.

The source code of our proposal can be found in the online repository [2].

4.2 Results

We start by analyzing the accuracy and the average rank results on the 30 processed datasets. Table 4 shows the accuracy results obtained by our proposal against the main state-of-the-art algorithms. The NA values refer to cases in which for any reason (memory overflow, libraries limitations, etc.), a model has not been obtained correctly, and it has been impossible to perform the desired classification. As we can see in Table 4, our proposal CMFMTS+RF, called CMFM+RF for simplification, obtains the best results among the four models we have proposed: CMFM+C5.0B, CMFM+RF, CMFM+SVM, and CMFM+1NN-ED. The CMFM+C5.0B model is especially interesting for cases where a simple and easy to interpret classifier is required, and that offers results close to the optimum ones as it happens in the datasets: Epilepsy and LSST. We can find cases in which CMFM+RF does not offer the best results among these four models, and it may be interesting to try other combinations as it happens in the datasets: AtrialFibrilation, EthanolConcentration, FaceDetection, HandMovementDirection, etc.

Table 4: Accuracy results on the UEA repository datasets: accuracy (%), average accuracy, median, average rank, and Win/Loss/Tie Ratio. The best results are stressed in bold.

Part 1:

Datasets	CMFM + C5.0B	CMFM + RF	CMFM + SVM	CMFM + 1NN-ED	LCEM	XGBM	RFM	MLSTM - FCN
ArticularyWordRecognition	92.0	98.8	97.3	98.7	**99.3**	99.0	99.0	98.6
AtrialFibrillation	6.7	19.1	26.7	26.7	**46.7**	40.0	33.3	20.0
BasicMotions	90.0	98.2	97.5	97.5	**100.0**	**100.0**	**100.0**	**100.0**
CharacterTrajectories	94.2	97.0	97.0	93.3	97.9	98.3	98.5	**99.3**
Cricket	86.1	97.7	95.8	98.6	98.6	97.2	98.6	98.6
DuckDuckGeese	54.0	51.0	42.0	46.0	37.5	40.0	40.0	**67.5**
EigenWorms	84.7	89.5	84.7	77.9	52.7	55.0	**100.0**	80.9
Epilepsy	99.3	**99.9**	97.8	96.4	98.6	97.8	98.6	96.4
ERing	80.7	**93.1**	93.0	90.4	20.0	13.3	13.3	13.3
EthanolConcentration	22.1	26.0	26.6	22.8	37.2	42.2	**43.3**	27.4
FaceDetection	55.7	55.7	58.3	50.3	61.4	**62.9**	61.4	55.5
FingerMovements	50.0	50.1	46.0	49.0	59.0	53.0	56.0	**61.0**
HandMovementDirection	17.6	24.5	28.4	17.6	**64.9**	54.1	50.0	37.8
Handwriting	17.2	27.4	18.7	23.4	28.7	26.7	26.7	54.7
Heartbeat	76.1	76.8	72.7	62.4	76.1	69.3	**80.0**	71.4
InsectWingbeat	NA	**67.7**	10.0	26.6	22.8	23.7	22.4	10.5
JapaneseVowels	79.5	83.7	77.8	69.5	97.8	96.8	97.0	**99.2**
Libras	82.2	84.7	81.7	80.0	77.2	76.7	78.3	**92.2**
LSST	65.2	**67.3**	65.6	50.4	65.2	63.3	61.2	64.6
MotorImagery	49.0	50.3	50.0	40.0	**60.0**	46.0	55.0	53.0
NATOPS	87.2	83.5	80.0	76.1	91.6	90.0	91.1	**96.1**
PEMS-SF	91.3	**99.9**	66.5	78.0	94.2	98.3	98.3	65.3
PenDigits	93.7	95.2	95.9	93.7	97.7	95.1	95.1	**98.7**
PhonemeSpectra	22.8	28.4	24.7	17.1	**28.8**	18.7	22.2	27.5
RacketSports	73.0	80.6	80.9	69.7	**94.1**	92.8	92.1	88.2
SelfRegulationSCP1	81.6	82.0	77.1	70.0	83.9	82.9	82.6	**86.7**
SelfRegulationSCP2	48.3	41.8	45.0	46.1	**55.0**	48.3	47.8	52.2
SpokenArabicDigits	93.3	97.6	97.9	91.5	97.3	97.0	96.8	**99.4**
StandWalkJump	26.7	36.3	26.7	20.0	40.0	33.3	**46.7**	**46.7**
UWaveGestureLibrary	65.0	77.5	72.8	74.4	89.7	89.4	90.0	85.7
Mean	62.8	**69.4**	64.5	61.8	69.1	66.7	69.2	68.3
Median	74.6	79.1	72.8	69.6	70.7	66.3	**79.2**	69.5
Average Rank	10.25	7.1	9.25	11.37	**4.23**	6.67	5.18	5.33
Win/Loss/Tie Ratio	0/30/0	5/25/0	0/30/0	0/30/0	8/22/2	2/28/1	5/25/2	**11/19/2**

[2]Complexity Measures and Features for Multivariate Times Series classification. `https://github.com/fjbaldan/CMFMTS/`

Part 2:

Datasets	WEASEL + MUSE	ED-1NN	DTW-1NN-I	DTW-1NN-D	ED-1NN (norm)	DTW-1NN-I (norm)	DTW-1NN-D (norm)
ArticularyWordRecognition	**99.3**	97.0	98.0	98.7	97.0	98.0	98.7
AtrialFibrillation	26.7	26.7	26.7	20.0	26.7	26.7	22.0
BasicMotions	**100.0**	67.5	**100.0**	97.5	67.6	**100.0**	97.5
CharacterTrajectories	99.0	96.4	96.9	99.0	96.4	96.9	98.9
Cricket	98.6	94.4	98.6	**100.0**	94.4	98.6	**100.0**
DuckDuckGeese	57.5	27.5	55.0	60.0	27.5	55.0	60.0
EigenWorms	89.0	55.0	60.3	61.8	54.9	NA	61.8
Epilepsy	99.3	66.7	97.8	96.4	66.6	97.8	96.4
ERing	13.3	13.3	13.3	13.3	13.3	13.3	13.3
EthanolConcentration	31.6	29.3	30.4	32.3	29.3	30.4	32.3
FaceDetection	54.5	51.9	51.3	52.9	51.9	NA	52.9
FingerMovements	54.0	55.0	52.0	53.0	55.0	52.0	53.0
HandMovementDirection	37.8	27.9	30.6	23.1	27.8	30.6	23.1
Handwriting	53.1	37.1	50.9	**60.7**	20.0	31.6	28.6
Heartbeat	72.7	62.0	65.9	71.7	61.9	65.8	71.7
InsectWingbeat	NA	12.8	NA	11.5	12.8	NA	NA
JapaneseVowels	97.8	92.4	95.9	94.9	92.4	95.9	94.9
Libras	89.4	83.3	89.4	87.2	83.3	89.4	87.0
LSST	62.8	45.6	57.5	55.1	45.6	57.5	55.1
MotorImagery	50.0	51.0	39.0	50.0	51.0	NA	50.0
NATOPS	88.3	85.0	85.0	88.3	85.0	85.0	88.3
PEMS-SF	NA	70.5	73.4	71.1	70.5	73.4	71.1
PenDigits	96.9	97.3	93.9	97.7	97.3	93.9	97.7
PhonemeSpectra	19.0	10.4	15.1	15.1	10.4	15.1	15.1
RacketSports	91.4	86.4	84.2	80.3	86.8	84.2	80.3
SelfRegulationSCP1	74.4	77.1	76.5	77.5	77.1	76.5	77.5
SelfRegulationSCP2	52.2	48.3	53.3	53.9	48.3	53.3	53.9
SpokenArabicDigits	98.2	96.7	96.0	96.3	96.7	95.9	96.3
StandWalkJump	33.3	20.0	33.3	20.0	20.0	33.3	20.0
UWaveGestureLibrary	**90.3**	88.1	86.9	**90.3**	88.1	86.8	**90.3**
Mean	64.3	59.1	63.6	64.3	58.5	57.9	62.9
Median	67.8	58.5	63.1	66.5	58.5	61.7	66.5
Average Rank	5.93	10.25	8.83	7.65	10.6	9.27	8.08
Win/Loss/Tie Ratio	3/27/3	0/30/0	1/29/1	3/27/2	0/30/0	1/29/1	2/28/2

If we compare our proposal with the rest of the state-of-the-art algorithms, we can see how CMFM+RF obtains an average rank of 7.1, close to the one obtained by LCEM (4.23), RFM (5.18), MLSTM-FCN (5.33), and WEASEL+MUSE (5.93). We have included two decimals for the average rank so that the differences shown in Figure 3 can be better appreciated. If we observe the Win/Loss/Tie ratio, we can see that MLSTM-FCN obtains the best results in 11 datasets, followed by LCEM, which wins in 8 datasets, and CMFM+RF and RFM, which obtains the best results in 5 datasets. These behaviors are reflected in the CD diagram shown in Figure 3. This diagram shows that there is no statistically significant difference, for an α of 0.05, between the previously mentioned models, in addition to the DTW-1NN-D model. In Table 6, we include the p-values obtained by the Wilcoxon Signed-Rank test of all the accuracies pairs of our models and the best models of the state-of-the-art in the first union line in Figure 3. For our CMFM+RF model, all the p-values are higher than the significance level 0.05, so we cannot reject the null hypothesis, in which the results of each model come from the same population. Those results indicate that our CMFMTS+RF proposal offers results that are statically indistinguishable from those obtained by the main state-of-the-art algorithms. For the rest of our models, we can see that the null hypothesis is rejected except in the cases CMFM+C5.0B with DTW-1NN-D and DTW-1NN-D (norm), and CMFM+SVM with WEASEL+MUSE, DTW-1NN-D, and DTW-1NN-D (norm). These results reinforce CMFM+RF as our best model. If we analyze the median and average accuracy values of Table 4, we can see that our proposal obtains competitive results. The NA values have been transformed to 0 for the calculation made.

Analyzing the results obtained for some specific cases, we can appreciate significant differences between the different proposals. For example, in the DuckDuckGeese dataset, the MLSTM-FCN algorithm obtains 7.5 points of difference with the next best result. LCEM and similar proposals obtain results with significant differences concerning the rest of

Table 6: Wilcoxon Signed-Rank test p-values obtained by the Wilcoxon Signed-Rank test of all the accuracies pairs of our models and the best models of the state-of-the-art in the first union line in Figure 3.

Models	LCEM	XGBM	RFM	MLSTM - FCN	WEASEL + MUSE	CMFM + RF	DTW-1NN-D	DTW-1NND (norm)
CMFM+RF	0.1650	0.9672	0.1294	0.0977	0.2848	-	0.5237	0.4224
CMFM+C5.0B	0.0061	0.0410	0.0012	0.0012	0.0111	0.0001	0.1442	0.2177
CMFM+SVM	0.0027	0.0379	0.0011	0.0047	0.0643	0.0045	0.4732	0.7499
CMFM+1NN	0.0020	0.0036	0.0004	0.0032	0.0049	$6.9e^{-06}$	0.0373	0.0397

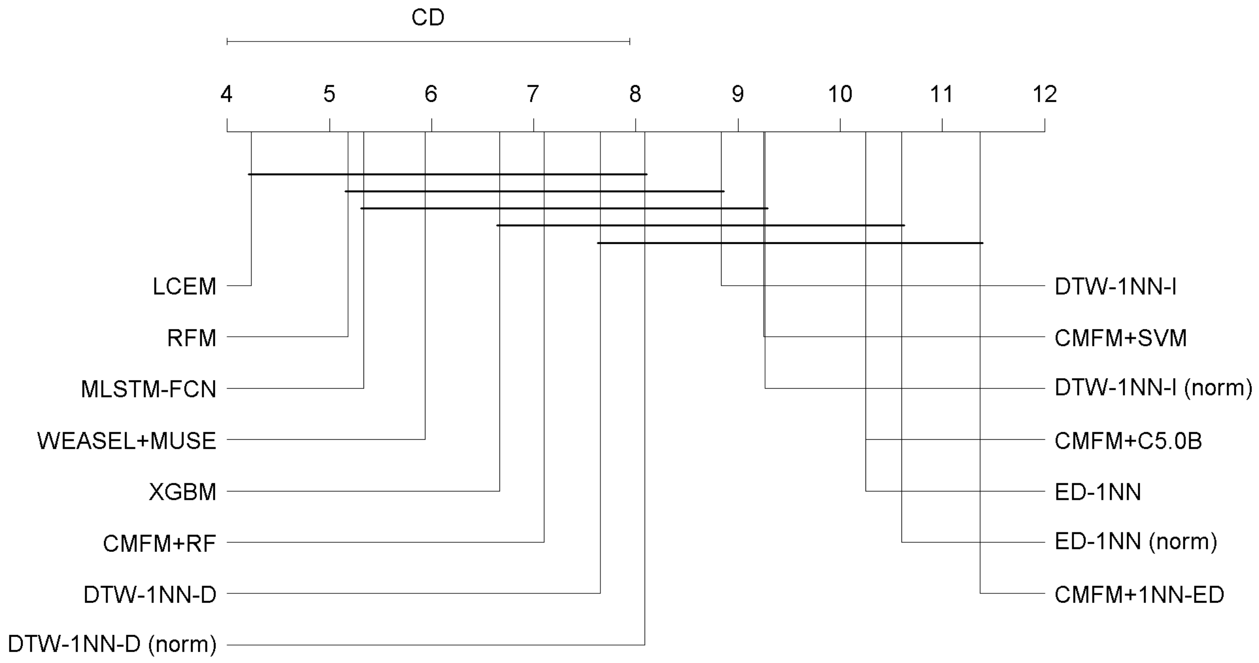


Figure 3: Critical Difference diagram, $\alpha = 0.05$.

the methods, as can be seen in the HandMovementDirection dataset. In the ERing dataset, we can see a big difference between our CMFM+Any proposals and the rest of the algorithms. These cases confirm the idea that in the field of CMTS, the results are strongly linked to the data itself and the approach used. It is especially complicated to find an approach that is able to face all kinds of problems with optimal results or close to them.

Based on the results shown in this section, we can conclude that:

- CMFM+RF is the model of our proposal that offers the best results among the four proposed models.
- The CMFM+RF model offers, with its default configuration, competitive results that are statistically indistinguishable from the main state-of-the-art algorithms, which optimize their parameters for each dataset.
- The CMFM+C5.0B model offers for some datasets results close to the best ones. This behavior turns it into a very interesting model because of its high interpretability.
- The accuracy of the classifiers obtained with the proposal of this work has proven to be competitive over a wide range of datasets.

5 Analysis of the interpretability

As clearly stated by many authors, interpretability has become a key property of machine learning systems. It relates to the easiness of understanding the decision-making process of the system.

In this section, we analyze the interpretability of the classification models that can be built following our proposal and how relevant knowledge can be derived from them. We begin by discussing the interpretability contributed by the considered features (Section 5.1). Then we consider the interpretability provided by trees (Section 5.2). Finally, we analyze how to extract knowledge from the models built upon the features (Section 5.3). First, we study how to assess the overall importance of the selected features as well seeking even more simplified models. Then the variable importance within the multivariate time series is addressed.

5.1 Interpretability of our proposal

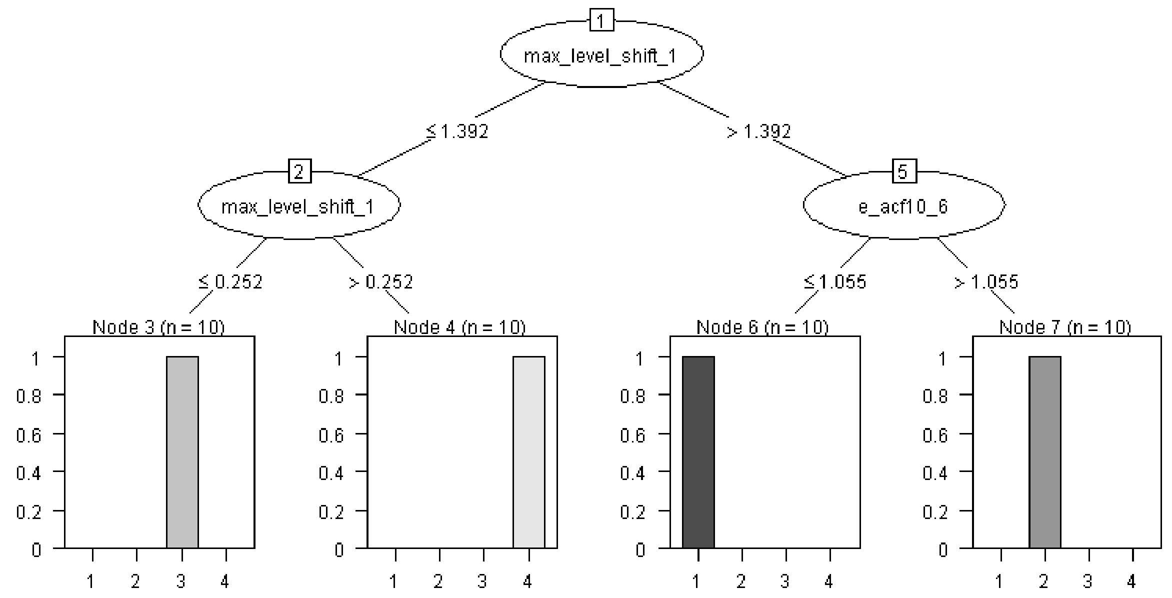
The interpretation of a classifier depends mainly on features upon which it is based (inputs) and the actual algorithm (the technique). The representation features have been chosen following, among other criteria, interpretability. These features express different behaviors of each time series. It is expected that the time series that belong to the same class show similar behaviors. Representing the time series through these features allows us to evaluate, numerically, these behaviors, compare them, and draw conclusions. For example, the value distribution of the time series can define the difference between the two classes of one problem that compose it. Features such as kurtosis and skewness allow us to differentiate time series with different value distribution. Also, the variability level or chaos of the time series may allow us to differentiate between the distinct classes. In this case, entropy related features allow us to quantify and compare this behavior between different time series. Usually, we find these differentiating patterns based on different features, obtaining a greater expressiveness and interpretability of the results. The overall point is that most of these features are easier to understand than the original time series values. The user can grasp a better understanding of the time series behavior through these well stated features. The interpretability of them is usually greater than a vector of lagged-values. Furthermore, they are independent of the classifier to be used.

On the other hand, classifiers offer different interpretability levels depending on the structure and nature of the connections between inputs and outputs that they build and on their number. For example, it is more useful to know that the difference between two classes depends on the linearity of the time series, on its distribution of values, or the stability of its values, among others, than on particular values in certain instants of time. So, for example, classification trees are easier to understand than deep neural networks. In addition, they can be easily translated into a set of rules. Analyzing the models considered in this work, we can observe that the C5.0B model offers us a simple decision tree based on the features used, although as we saw in Table 4, its accuracy results are not the best. On the other hand, an RF offers competitive results in exchange for sacrificing part of their interpretability, although RF is able to offer an assessment of the importance of each feature in the final model that can be very useful. In contrast, models such as 1NN-ED lack interpretability since they work on how much one instance resembles another, and SVMs are really complex to interpret since weights can be affected by external components unrelated to the underlying importance of each variable. Since tree-based models offer different interpretability tools, we will focus on them in this section.

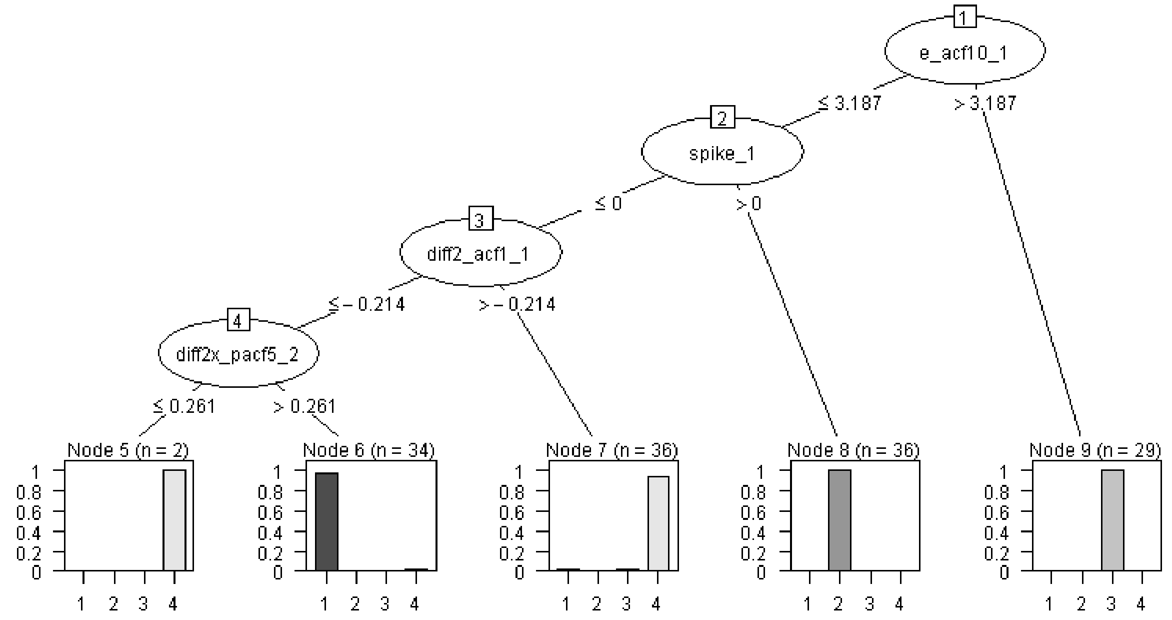
5.2 C5.0 with Boosting model interpretability

Decision trees offer very simple and straightforward interpretability. The C5.0 model with boosting allows us to explain its results using the rules included in the tree. This, together with the use of well-known features, such as those used in our proposal, allows us to understand the decisions of the model based on well-defined behaviors of the time series.

For illustrative purposes, we have included two simple examples. In Figure 4, we show two examples of a single C5.0B tree for the BasicMotions and Epilepsy datasets. BasicMotions is a dataset with 4 classes and MTS with 6 variables. As we can see in Figure 4a, our approach allows us to solve this problem with a simple tree composed of 3 nodes. Two of these nodes refer to features of variable 1, and the remaining one refers to variable 6. According to this tree, we can say that a time series belongs to class 3 if the maximum mean shift between two consecutive windows, for variable 1, is less than 0.252. On the other hand, we can know that a time series belongs to class 2 if the maximum mean shift between two consecutive windows, for variable 1, is greater than 1.392 and the sum of the first ten squared autocorrelation coefficients, for variable 6, is greater than 1.055. Although the results obtained for this dataset are not the best possible ones, it is remarkable how you can obtain acceptable results with such a simple decision tree. Next, we consider the dataset Epilepsy which has 4 classes and MTS with 3 variables. In Figure 4b, we see a tree with 4 nodes: three of them refer to variable 1, and another to variable 2. In this dataset, if the sum of the first ten squared autocorrelation coefficients, for variable 1, is higher than 3.187, we know that the time series belongs to class 3. Otherwise, if the spikiness variance of the leave-one-out variances of the remainder component of the time series is positive, we can say that the time series belongs to class 2. Another relevant conclusion is that the number of features actually used to describe the classifier is rather small, making it easier to comprehend the process.



(a) BasicMotions example of a single C5.0B tree with time series features.



(b) Epilepsy example of a single C5.0B tree with time series features.

Figure 4: Interpretability CMFM example.

5.3 Feature and variable importance

The set of 41 features selected for the representation of time series enable a diverse and extensive description of the characteristics of the time series. For the purpose of classification, however, not all of them are always necessary. This fact has been observed in the classifiers obtained in the empirical analysis detailed in section 4 and is clearly illustrated in the examples depicted in subsection 5.2. It is obvious that simplicity in the classifiers leads to enhanced interpretability. This simplification can be achieved for each classifier by including a feature selection stage in the preprocessing of each workflow. However, we endeavor to seek a more general approach. So, in this section, we describe the work —and attained results— reached in the trek of a smaller subset of features. This trek has been guided through the relevance of the proposed features. An analogue approach can be followed to identify the importance of the different variables composing an MTS.

While different definitions of feature relevance are published, and the interpretability of the features has already been discussed, we turn an eye now towards accuracy. Since Random Forest is the model of our proposal that obtains the best accuracy results, we are going to analyze the importance given to each variable in this section. The importance measure used for each variable by this model is the total decrement with respect to the node impurities, resulting from splitting on the variable, averaged over all trees. In classification problems, the node impurity is measured by the Gini Index [10]. We have performed three different analyses that allow us to extract the desired information:

1. Analysis of the importance of each feature in each dataset (Section 5.3.1). This allows us to know which features have a greater contribution to the final result. Based on the features with the highest contribution, we can determine which behaviors, represented in those features, are the ones that define each type of time series.
2. Analysis of the accumulated importance of each feature over a large set of datasets (Section 5.3.2). This analysis allows us to identify which features are representative of most problems and which ones are uninteresting.
3. Analysis focused on the cumulative importance of features for each variable composing the MTS (Section 5.3.3). In this way, we could identify which variables contain a greater amount of information about a given problem. These variables would be the most interesting ones to solve the problem.

5.3.1 Feature importance by dataset

Since we are working with MTS, each feature is calculated for each variable of a MTS. Therefore, each feature of the original set offers a different result for each variable of the same MTS. For this, it is normal obtaining different values of *importance* for the same feature in different variables. To easily compare the importance of the original features in MTS, we need to simplify the importance values of each feature over each variable to a global importance value per feature. For this reason, we have calculated the mean of the importance of each feature over all the variables. For example, for the approximation entropy feature in a 7-variables MTS, we get 7 different values of importance (1 for each variable to which its corresponding approximation entropy feature is calculated). We add these 7 values and divide them by the number of variables of our MTS. In this way, we also penalize the importance of any features that could not be calculated in any variable. Finally, we normalize these last values between 0 and 1 for each dataset. This produces a normalized measure that is best explored in a graphical way. Figure 5 shows a heatmap of the importance of each feature in the RF classifier for each dataset, where the datasets have been ordered by the accumulated importance of the 41 features.

In Figure 5, we can see significant differences among the datasets. We can differentiate two bands: the upper band where only some features accumulate great importance, and the lower band where multiple features have great importance. In the upper band, we can see datasets for which the classifiers are *dominated* by up to four features with high importance (SelfRegulationSCP1, InsectWingBeat, SpokenArabicDigits, NATOPS, BasicMotions, DuckDuckGeese, PEMS-SF, among others). In these cases, two categories are observed: the set of features used is sufficiently expressive to address the problem satisfactorily with competitive results (RF: InsectWingBeat, and PEMS-SF), or the selected features are not sufficient and other approaches achieve significantly better results (MLSTM-FCN: BasicMotions, DuckDuckGeese, NATOPS, SelfRegulationSCP1, and SpokenArabicDigits). In the lower band, we can identify cases where all the features are necessary (HandMovementDirection, PhonemeSpectra, Handwriting, FingerMovements, MotorImagery, EthanolConcentration, ArticularyWordRecognition, among others). In these cases, we can observe two behaviors. On the one hand, the differentiating capabilities of the features are not enough for some of them to stand out from the rest, so the classifier assigns similar importance to a large number of features. On the other hand, in datasets with a complex problem, it is not possible to find a reduced subset of features capable of explaining the problem. In these cases, more complex solutions are obtained, with a high number of features, capable of offering results very close to the best ones (ArticularyWordRecognition and PhonemeSpectra).

5.3.2 Accumulated feature importance over set of datasets

Another particularly interesting analysis to be carried out is related to the importance at the feature level. In Figure 6, we show the average importance of each feature throughout all the datasets. We have ordered the features in decreasing order of the Average Gini Index. These values have been obtained from the results shown in Figure 5. The average value of the importance of each feature has been calculated over the 30 datasets processed. We can see that there is a group of three distinguished features that frequenly reach the highest importance values, namely, *curvature*, *linearity*, and *spike*. This group has values of importance far superior to the rest. As a fast check experiment, we have built a C5.0B model for every dataset restricted to use only these three variables and have observed an average improvement of accuracy of 2.2%. A second distinguished breakpoint in the importance curve leads to a second group of relevant

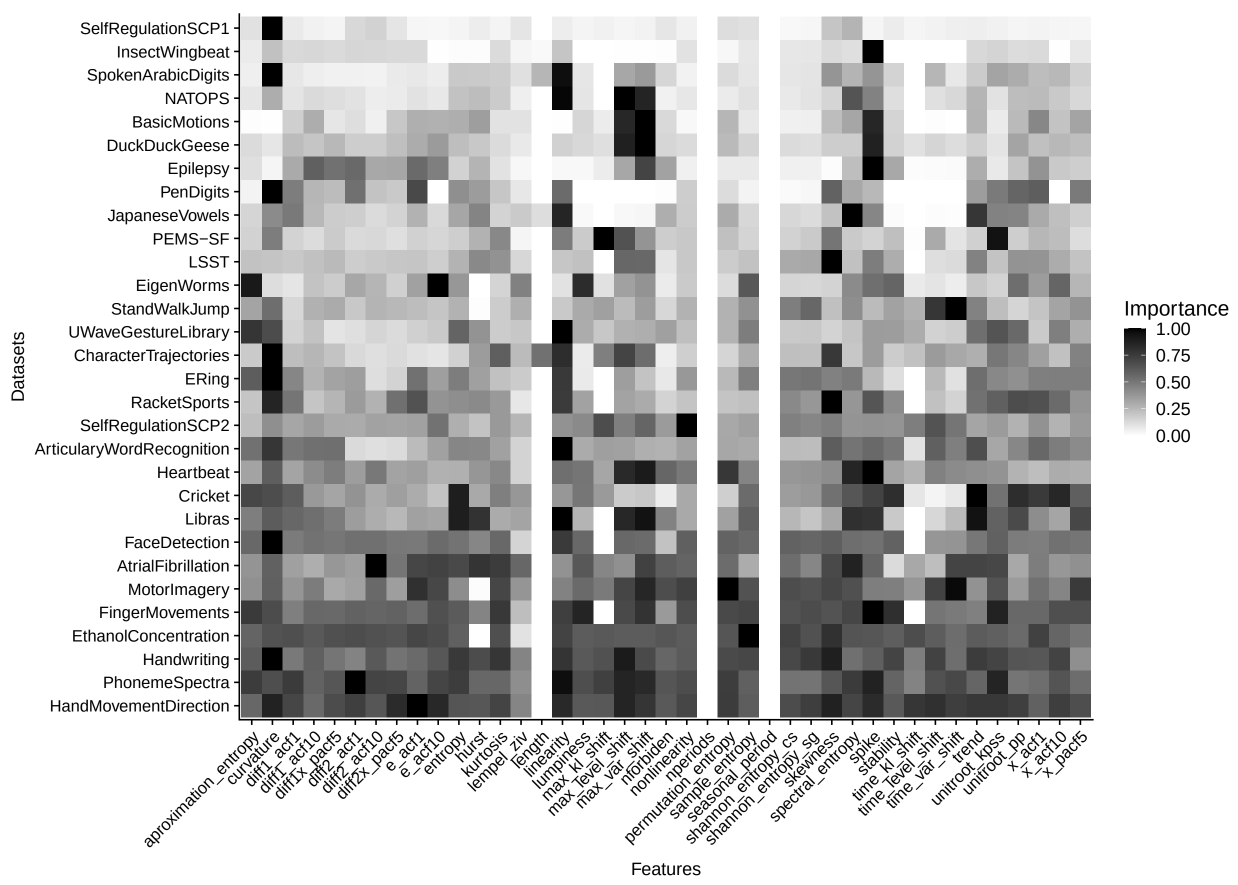


Figure 5: Ordered features importance heatmap.

features: *max_var_shift*, *max_level_shift*, *skewness*, *spectral_entropy*, and *trend*. All of them are assigned importance values greater than 0.45. Even further, it is interesting to realize that the features related to the complexity of a time series get the highest importance values. Higher values achieved by features such as *trend* confirm that the components of the time series are very descriptive and useful when extracting information from them. Other features such as *curvature*, *linearity*, and *spike*, shown as characteristic behaviors of the time series, are especially useful in describing them.

On the other hand, there are also features with particularly low values of importance: *nperiods*, *seasonal_periods*, and *length*. The feature *length* is not of high importance because, in the UEA repository, the vast majority of datasets are composed of MTS of equal length. The features *nperiods* and *seasonal_periods* have importance values of 0 because we have processed all the time series with a frequency of 1, making the processing as general as possible. If all the time series have the same frequency, the features *nperiods* and *seasonal* always return the same value, which does not provide distinguishing information for the problem. In the case that the best results are sought, and a detailed analysis of the time series is carried out in which data on seasonality is available, these measures can be very useful. Furthermore, features such as *max_kl_shift* and *lempel_ziv* have obtained low average importance values, although they are particularly explanatory. If we look at Figure 5, we find some datasets like PEMS-SF and EigenWorms in which *max_kl_shift* and *lempel_ziv* have a high importance, respectively. In this case, even if we identify features that are generally not interesting, they may be relevant for specific problems. These cases reinforce the idea that the selection of a representative set of features must be supported by theoretical knowledge about the structure of time series and by different analyses of results performed on large sets of datasets.

5.3.3 Variable importance

Finally, we analyze an important point in the field of MTSC, the existence of components or variables that contain a major part of the information on the problem. To assess the importance of a variable in a given dataset, we have computed the sum of the importance of the 41 features for each variable of the problem in question. Then, this value is normalized by dividing it by the maximum value of each sum. In this way, the importance value of any variable

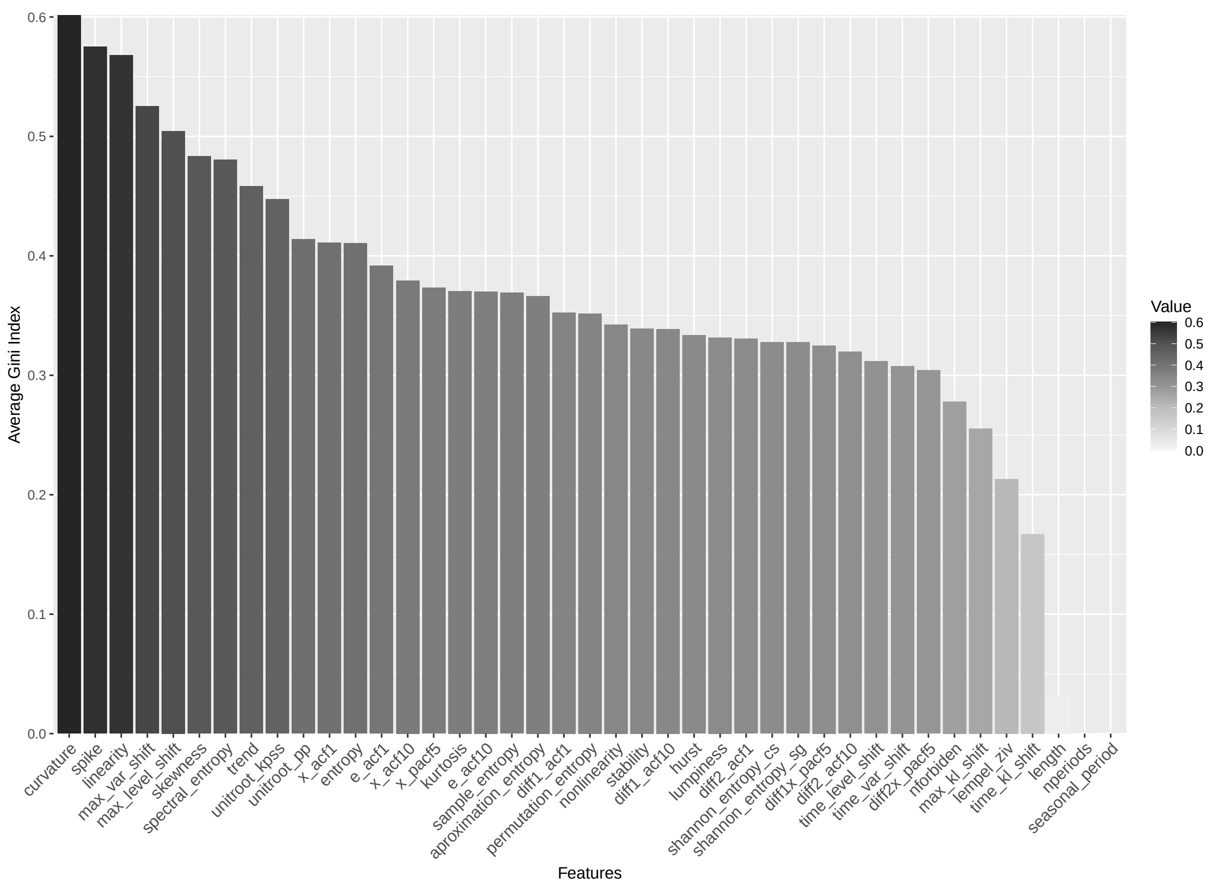


Figure 6: Average importance of each feature in the UEA repository.

belongs in $[0, 1]$, with 1 being the maximum. Several statistics for the variable importance of the considered datasets have been gathered in Table 7. "Sum" refers to the addition of the variable importance of all the variables of the dataset. Conversely, "Max", "Min", "Mean", "Median", and "SD" corresponds to the maximum, minimum, mean, median, and standard deviation of the variable importance.

Some observations can be made from the table. For example, in the PhonemeSpectra dataset, all 11 variables are of similar importance, and CMFM+RF was close to the best results obtained. This means that all variables contain information of interest. On the other hand, in NATOPS, there is a relevant dispersion among variable importance. To further illustrate diversity in variable importance, we have selected some datasets and plotted a histogram of their respective variable importance —see Figure 7.

For the BasicMotions dataset, in Figure 7a, we can see 2 variables with much higher importance than the rest, together with a third variable that also stands out. These variables are, in decreasing order of importance: 2, 6, 1, 3, 5, and 4. If we compare these values of importance —derived from RF— against the C5.0B tree, shown in Figure 4a, we can realize that two of the tree nodes have features of the variable 1, and the remaining node has a feature of the variable 6, which are the second and third most important variables. In the case of dataset Epilepsy —see Figure 7b—, we can see 2 variables significantly distinguished from the rest. These variables are, in decreasing order of importance: 1, 2, and 3. Figure 4b shows a C5.0B tree for this dataset. Three of its nodes have features of the variable 1, the most important variable, while the remaining node has a feature of the variable 2, the second most important variable according to the RF. These examples show a certain relation between the variables with more importance according to the RF and those used by a simple classifier such as C5.0B.

In the case of the NATOPS dataset, the most relevant information is expressed in just 24 of the variables. In Figure 7c, we can see that it is more difficult to obtain well-differentiated groups of variables according to their importance. In this case, we can see that the 3 variables with the greatest importance are significantly distanced from the rest, with importance values higher than 0.70. Depending on the information sought and the difficulty of the problem, we could decide to lower the threshold, create different groups of variables, etc. These histograms are especially interesting for

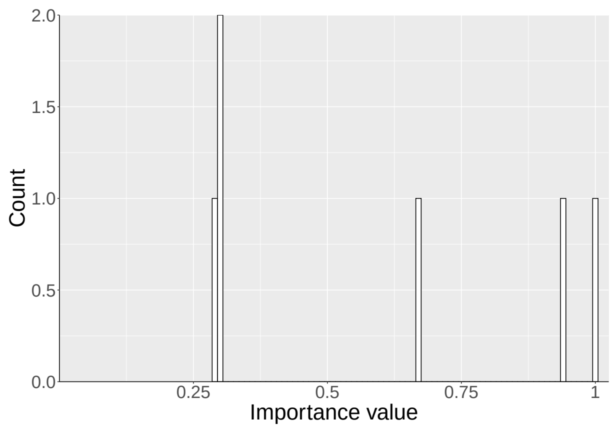
Table 7: Statistics of the variable importance.

Datasets	Sum	Max	Min	Mean	Median	SD	Variables
ArticularyWordRecognition	4.816	1	0.190	0.535	0.504	0.263	9
AtrialFibrillation	1.720	1	0.720	0.860	0.860	0.198	2
BasicMotions	3.495	1	0.291	0.582	0.485	0.332	6
CharacterTrajectories	2.120	1	0.359	0.707	0.762	0.324	3
Cricket	4.726	1	0.627	0.788	0.797	0.131	6
DuckDuckGeese	143.003	1	0	0.106	0.074	0.115	1345
EigenWorms	3.739	1	0.367	0.623	0.594	0.208	6
Epilepsy	2.295	1	0.352	0.765	0.943	0.359	3
ERing	3.356	1	0.659	0.839	0.848	0.152	4
EthanolConcentration	2.971	1	0.980	0.990	0.991	0.010	3
FaceDetection	100.234	1	0.557	0.696	0.692	0.061	144
FingerMovements	19.528	1	0.547	0.697	0.676	0.097	28
HandMovementDirection	9.141	1	0.795	0.914	0.927	0.079	10
Handwriting	2.701	1	0.818	0.900	0.883	0.092	3
Heartbeat	30.558	1	0.307	0.501	0.470	0.162	61
InsectWingbeat	59.976	1	0.195	0.300	0.221	0.159	200
JapaneseVowels	9.279	1	0.563	0.773	0.789	0.138	12
Libras	1.901	1	0.901	0.950	0.950	0.070	2
LSST	5.213	1	0.718	0.869	0.873	0.138	6
MotorImagery	50.752	1	0.588	0.793	0.791	0.097	64
NATOPS	9.634	1	0.140	0.401	0.334	0.215	24
PEMS-SF	30.718	1	0	0.032	0.013	0.078	963
PenDigits	1.684	1	0.684	0.842	0.842	0.223	2
PhonemeSpectra	10.917	1	0.984	0.992	0.993	0.004	11
RacketSports	4.493	1	0.304	0.749	0.810	0.263	6
SelfRegulationSCP1	4.044	1	0.532	0.674	0.629	0.175	6
SelfRegulationSCP2	6.673	1	0.901	0.953	0.950	0.037	7
SpokenArabicDigits	5.962	1	0.162	0.459	0.325	0.302	13
StandWalkJump	3.614	1	0.758	0.904	0.928	0.112	4
UWaveGestureLibrary	2.782	1	0.877	0.927	0.904	0.064	3

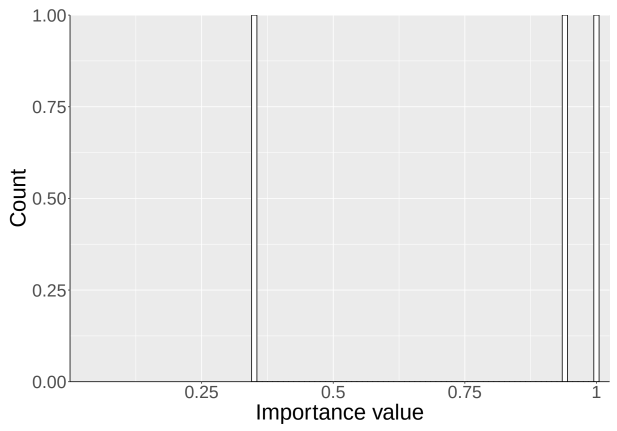
datasets with a large number of variables. For example, in the PEMS-SF dataset, Figure 7d, only some of the 963 variables have a high importance, giving residual importance to the rest. In this case, there are only 5 variables with an accumulated importance value higher than 0.75. These variables are, in decreasing order of importance: 212, 55, 604, 172, and 187.

To better understand the variable importance distribution, we have calculated the percentage of cumulative importance of each variable in each dataset. For this, we have calculated the cumulative importance of the set of features for each variable and dividing these values by the sum of the importance of all the features. In this way, we can see which variables contain a greater amount of useful information, e.g., for a dataset composed of MTS with 3 variables, we have calculated the sums of the importance of the 41 features used for each variable and divided those values by the total of the sum of the importance of the 41 features for the 3 variables. In Figure 8, we show the percentage of cumulative importance for each variable for some processed datasets: it is easy to spot the differences in importance by comparing the relative lengths of the colored pieces into which each bar is divided. Each of these pieces represents a single variable. For example, in the ArticularyWordRecognition dataset, variables 1, 4, 7, and 9 are of less importance; ERing dataset shows a great importance accumulated in variables 1 and 4; Epilepsy dataset has much of its useful information in variables 1 and 2; and so forth. With these results, the preprocessing of the data can be modified in such a way as to improve the recording of the data of these variables or to give them greater importance in the learning process.

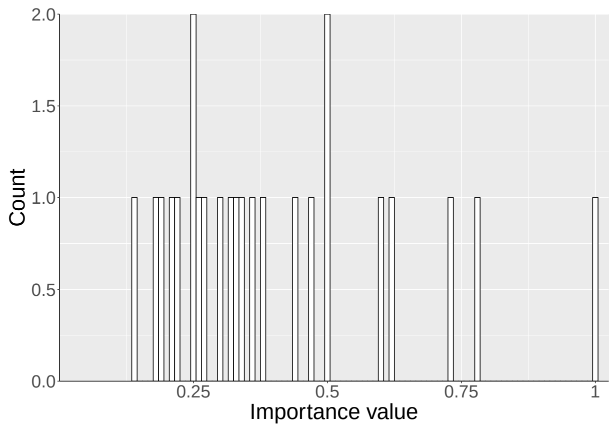
Based on the study conducted in this section, we can conclude that:



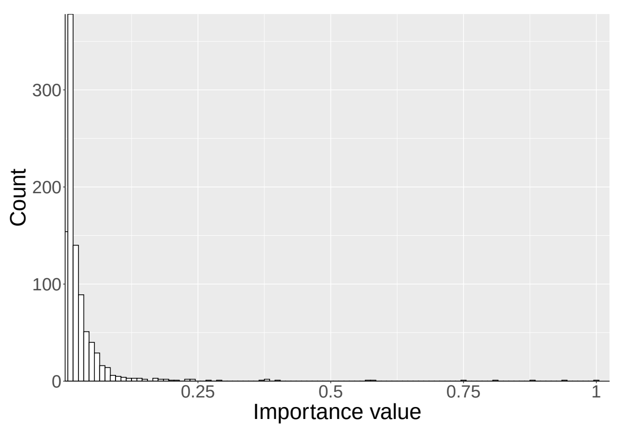
(a) BasicMotions dataset.



(b) Epilepsy dataset.



(c) NATOPS dataset.



(d) PEMS-SF dataset.

Figure 7: Histograms of variable importance values.

- Combining easily interpretable models with understandable time series features, the user can better understand and explain the decision-making process. The CMFM+C5.0B model is especially interesting in this respect.
- CMFM+RF allows us to define a feature importance measure that can be used to reduce the set of considered features, leading to simpler and more interpretable models.
- The feature importance can be used to assess variable importance within multivariate time series. In addition, based on the feature importance, we have also defined a variable importance measure that allows us to identify the most relevant variables and thus guide or prioritize the time series caption, storage, and processing.

6 Conclusions

In this paper, we have presented a method to represent multivariate time series in terms of a set of interpretable features. This method enables the use of conventional classification algorithms on MTSC problems, considerably expanding the tools available to deal with this type of problem. The main benefit of this approach is to obtain interpretable classifiers so that the decision-making process can be better understood. We have designed and executed a thorough empirical study, based on the main repository of the state-of-the-art, composed of 30 datasets. The accuracy results of the built classifiers remain competitive with respect to the state-of-the-art results. In particular, no statistically relevant differences can be found between our CMFMTS+RF proposal and the most accurate already known methods.

The interpretability of the built classifiers has been extensively analyzed. Measures for feature importance and variable importance have been defined, allowing to derive relevant knowledge for each particular problem addressed with the proposed method. In addition, we have verified the existence of a set of features that maintains high importance

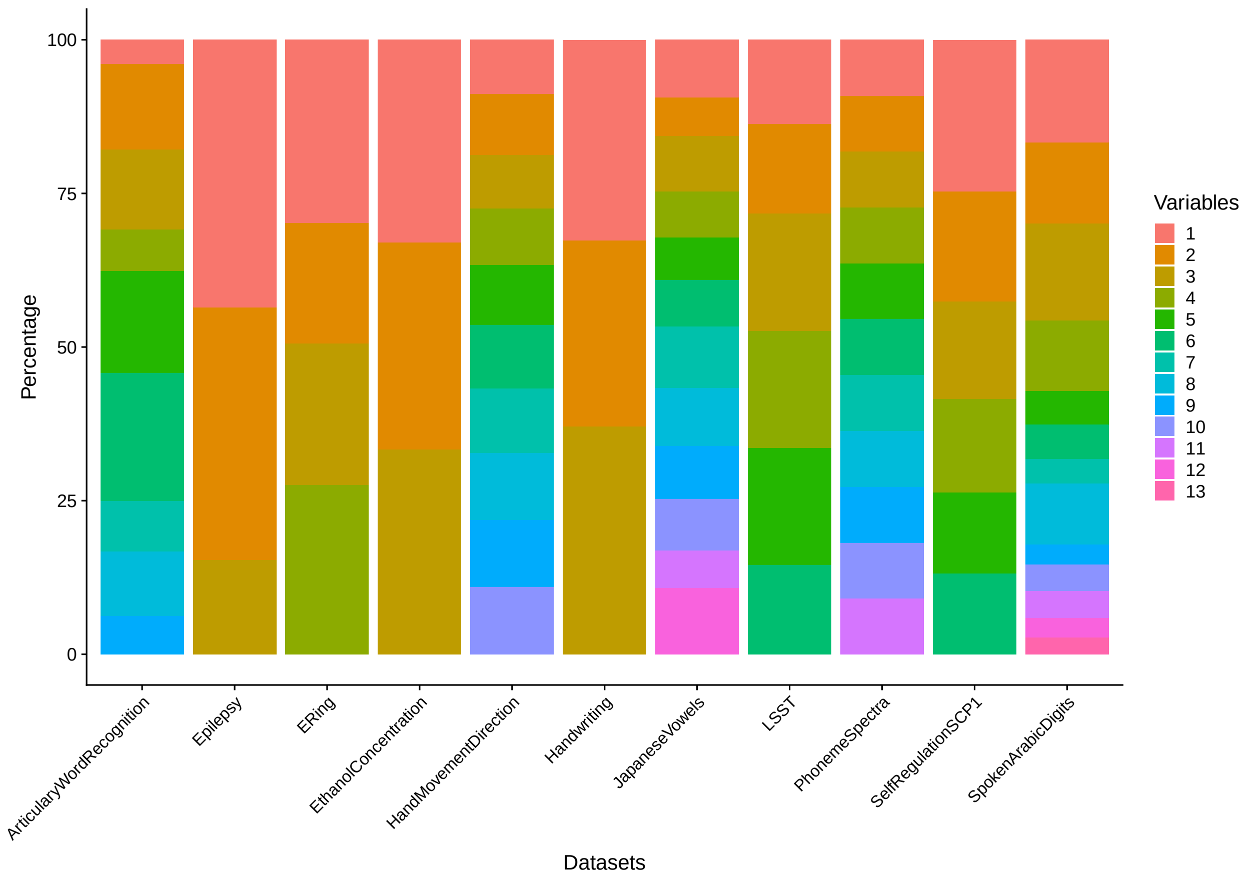


Figure 8: Accumulated importance by variable.

throughout different datasets. However, there are also certain cases where other features that are less important on average offer the best results. These features are usually related to characteristic behaviors of the time series.

The method has been implemented in the R programming language. The code is publicly available.

7 Acknowledgements

This research has been partially funded by the following grants: TIN2016-81113-R from the Spanish Ministry of Economy and Competitiveness, and P12-TIC-2958 from Andalusian Regional Government, Spain. Francisco J. Baldán holds the FPI grant BES-2017-080137 from the Spanish Ministry of Economy and Competitiveness.

References

[1] A. Abdiansah, R. Wardoyo, Time complexity analysis of support vector machines (SVM) in LibSVM, International journal computer and application 128 (3) (2015) 28–34.

[2] A. Antonucci, R. De Rosa, A. Giusti, F. Cuzzolin, Robust classification of multivariate time series by imprecise hidden Markov models, International Journal of Approximate Reasoning 56 (2015) 249–263.

[3] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, E. Keogh, The UEA multivariate time series classification archive, 2018, arXiv preprint arXiv:1811.00075.

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

[5] F. J. Baldán, J. M. Benítez, Complexity Measures and Features for Times Series classification, arXiv preprint arXiv:2002.12036.

[6] M. Baydogan, Multivariate Time Series Classification Datasets, `http://www.mustafabaydogan.com`, accessed: 2020-07-01 (2017).

[7] M. G. Baydogan, G. Runger, Learning a symbolic representation for multivariate time series classification, Data Mining and Knowledge Discovery 29 (2) (2015) 400–422.

[8] M. G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Mining and Knowledge Discovery 30 (2) (2016) 476–509.

[9] A. Bostrom, A. Bagnall, A shapelet transform for multivariate time series classification, arXiv preprint arXiv:1712.06428.

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

[11] W. A. Chaovalitwongse, R. S. Pottenger, S. Wang, Y.-J. Fan, L. D. Iasemidis, Pattern-and network-based classification techniques for multichannel medical data signals to improve brain diagnosis, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 41 (5) (2011) 977–988.

[12] M. Das, M. Pratama, A. Ashfahani, S. Samanta, FERNN: A fast and evolving recurrent neural network model for streaming data classification, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.

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

[14] F. K. Došilović, M. Brčić, N. Hlupić, Explainable artificial intelligence: A survey, in: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018, pp. 0210–0215.

[15] D. Dua, C. Graff, UCI Machine Learning Repository (2017).
URL `http://archive.ics.uci.edu/ml`

[16] K. Fauvel, É. Fromont, V. Masson, P. Faverdin, A. Termier, Local Cascade Ensemble for Multivariate Data Classification, arXiv preprint arXiv:2005.03645.

[17] T. Górecki, M. Łuczak, Multivariate time series classification with parametric derivative dynamic time warping, Expert Systems with Applications 42 (5) (2015) 2305–2312.

[18] J. Grabocka, M. Wistuba, L. Schmidt-Thieme, Fast classification of univariate and multivariate time series through shapelet discovery, Knowledge and Information Systems 49 (2) (2016) 429–454.

[19] F. Karim, S. Majumdar, H. Darabi, S. Harford, Multivariate LSTM-FCNs for time series classification, Neural Networks 116 (2019) 237–245.

[20] S. Karimi-Bidhendi, F. Munshi, A. Munshi, Scalable classification of univariate and multivariate time series, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018, pp. 1598–1605.

[21] I. Karlsson, P. Papapetrou, H. Boström, Generalized random shapelet forests, Data Mining and Knowledge Discovery 30 (5) (2016) 1053–1085.

[22] E. J. Keogh, M. J. Pazzani, Derivative dynamic time warping, in: Proceedings of the 2001 SIAM International Conference on Data Mining, SIAM, 2001, pp. 1–11.

[23] K. Khadiev, I. Mannapov, L. Safina, The Quantum Version Of Classification Decision Tree Constructing Algorithm C5. 0, arXiv preprint arXiv:1907.06840.

[24] M. Khan, N. Javaid, M. N. Iqbal, M. Bilal, S. F. A. Zaidi, R. A. Raza, Load prediction based on multivariate time series forecasting for energy consumption and behavioral analytics, in: Conference on Complex, Intelligent, and Software Intensive Systems, Springer, 2018, pp. 305–316.

[25] B. Klockl, Multivariate time series models applied to the assessment of energy storage in power systems, in: Proceedings of the 10th International Conference on Probablistic Methods Applied to Power Systems, IEEE, 2008, pp. 1–8.

[26] Z. Li, X. Jin, X. Zhao, Drunk driving detection based on classification of multivariate time series, Journal of Safety Research 54 (2015) 61–67.

[27] T. Lin, T. Guo, K. Aberer, Hybrid Neural Networks for Learning the Trend in Time Series, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, AAAI Press, 2017, p. 2273–2279.

[28] E. A. Maharaj, A. M. Alonso, Discriminant analysis of multivariate time series: Application to diagnosis based on ECG signals, Computational Statistics & Data Analysis 70 (2014) 67–87.

[29] G. Manis, Fast computation of approximate entropy, Computer methods and programs in biomedicine 91 (1) (2008) 48–54.

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

[31] A. McGovern, D. H. Rosendahl, R. A. Brown, K. K. Droegemeier, Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction, Data Mining and Knowledge Discovery 22 (1-2) (2011) 232–258.

[32] J. Mei, M. Liu, Y.-F. Wang, H. Gao, Learning a mahalanobis distance-based dynamic time warping measure for multivariate time series classification, IEEE Transactions on Cybernetics 46 (6) (2015) 1363–1374.

[33] C. Orsenigo, C. Vercellis, Combining discrete SVM and fixed cardinality warping distances for multivariate time series classification, Pattern Recognition 43 (11) (2010) 3787–3794.

[34] W. Pei, H. Dibeklioğlu, D. M. Tax, L. van der Maaten, Multivariate time-series classification using the hidden-unit logistic model, IEEE Transactions on Neural Networks and Learning Systems 29 (4) (2017) 920–931.

[35] S. Samanta, M. Pratama, S. Sundaram, N. Srikanth, A Dual Network Solution (DNS) for Lag-Free Time Series Forecasting, in: 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1–8.

[36] P. Schäfer, U. Leser, Multivariate time series classification with WEASEL+ MUSE, arXiv preprint arXiv:1711.11343.

[37] S. Seto, W. Zhang, Y. Zhou, Multivariate time series classification using dynamic time warping template selection for human activity recognition, in: 2015 IEEE Symposium Series on Computational Intelligence, IEEE, 2015, pp. 1399–1406.

[38] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, E. Keogh, Generalizing DTW to the multi-dimensional case requires an adaptive approach, Data Mining and Knowledge Discovery 31 (1) (2017) 1–31.

[39] K. S. Tuncel, M. G. Baydogan, Autoregressive forests for multivariate time series modeling, Pattern Recognition 73 (2018) 202–215.

[40] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E. Keogh, Indexing multi-dimensional time-series with support for multiple distance measures, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 216–225.

[41] M. Wistuba, J. Grabocka, L. Schmidt-Thieme, Ultra-fast shapelets for time series classification, arXiv preprint arXiv:1503.05018.

[42] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 947–956.

[43] C. Yu, L. Luo, L. L.-H. Chan, T. Rakthanmanon, S. Nutanong, A fast LSH-based similarity search method for multivariate time series, Information Sciences 476 (2019) 337–356.

4 SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments

- F. J. Baldán, Daniel Peralta, Yvan Saeys, J. M. Benítez. SCMFTS: Scalable and distributed Complexity Measures and Features for univariate and multivariate Time Series in Big Data environments. International Journal of Computational Intelligence Systems, (2021). Accepted.
  - Status: **Accepted**.
  - Impact Factor (JCR 2020): **1.736**
  - Subject Category: **Computer Science, Artificial Intelligence**
  - Rank: **100/139**
  - Quartile: **Q3**

SCMFTS: SCALABLE AND DISTRIBUTED COMPLEXITY MEASURES AND FEATURES FOR UNIVARIATE AND MULTIVARIATE TIME SERIES IN BIG DATA ENVIRONMENTS

Francisco J. Baldán[*]
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI, University of Granada
Granada, Spain, 18071
`fjbaldan@decsai.ugr.es`

José M. Benítez Sánchez
Department of Computer Science
and Artificial Intelligence
DiCITS, iMUDS, DaSCI, University of Granada
Granada, Spain, 18071
`J.M.Benitez@decsai.ugr.es`

Daniel Peralta
Data Mining and Modelling for Biomedicinee
VIB Center for Inflammation Research
Technologiepark-Zwijnaarde 71,
Ghent, Belgium, 9052
Department of Applied Mathematics,
Computer Science and Statistics
Ghent University, Krijgslaan 281,
S9, Ghent, Belgium, 9000
`daniel.peralta@irc.vib-ugent.be`

Yvan Saeys
Data Mining and Modelling for Biomedicinee
VIB Center for Inflammation Research
Technologiepark-Zwijnaarde 71,
Ghent, Belgium, 9052
Department of Applied Mathematics,
Computer Science and Statistics
Ghent University, Krijgslaan 281,
S9, Ghent, Belgium, 9000
`yvan.saeys@ugent.be`

ABSTRACT

Time series data are becoming increasingly important due to the interconnectedness of the world. Classical problems, which are getting bigger and bigger, require more and more resources for their processing, and Big Data technologies offer many solutions. Although the principal algorithms for traditional vector-based problems are available in Big Data environments, the lack of tools for time series processing in these environments needs to be addressed. In this work, we propose a scalable and distributed time series transformation for Big Data environments based on well-known time series features (SCMFTS), which allows practitioners to apply traditional vector-based algorithms to time series problems. The proposed transformation, along with the algorithms available in Spark, improved the best results in the state-of-the-art on the Wearable Stress and Affect Detection dataset, which is the biggest publicly available multivariate time series dataset in the University of California Irvine (UCI) Machine Learning Repository. In addition, SCMFTS showed a linear relationship between its runtime and the number of processed time series, demonstrating a linear scalable behavior, which is mandatory in Big Data environments. SCMFTS has been implemented in the Scala programming language for the Apache Spark framework, and the code is publicly available [5].

Keywords Time series · Time series features · Feature Based Approach · Big Data · Scalability

1 Introduction

Nowadays, we can find devices generating data anywhere and at any time [22]. With the expansion of new technologies, the volume of data generated is growing by leaps and bounds. Until now, the typical time series data comes from well-known fields, for example, from the stock market [24], from industry with power consumption logs [21], or from

[*]Corresponding author.

medical fields with specific applications, such as electrocardiograms [1]. However, nowadays we have access to a lot of new devices like smartwatches which continuously generate information through their incorporated sensors, such as heart rate, temperature or humidity monitors. All the data sources mentioned contain information of high importance that needs to be extracted and used to improve the service offered.

All the examples mentioned above are related to recording one or multiple magnitudes over time, generating a specific type of data named time series. Any magnitude regularly recorded over time is a time series. On the one hand, forecasting future values of a time series has been a very popular topic [30]. Forecasting stock price trends [41] is the most typical example but also one of the hardest. On the other hand, the search for patterns in time series is a task that is attracting more attention each day. Problems like detection of fraudulent energy consumption [38] or detection of heart malfunctions [15] are gathering increasing attention from the research community. With increasingly cheaper sensors and more complex models, new problems of interest appear every day.

There are growing numbers of cases in which multiple variables are recorded at the same time in the same process [17], generating Multivariate Time Series (MTS). MTS problems have additional complexity due to the relationships between the different variables that compose each MTS. An example of this kind of problem is the forecast of the energy demand when additional meteorological variables are available like temperature, humidity, or wind speed, among others.

The data volume generated by the expansion of IoT scales quickly providing a quantity of data that is not processable by the traditional computation model. The concept of Big Data arises in order to face this kind of problem. The MapReduce paradigm [8] proposes a distributed computational model that can face a high volume of data efficiently. Apache Spark [16] is a popular framework that offers high-speed capabilities and includes the MapReduce workflow. Spark has one of the most extensive libraries for machine learning in Big Data environments, MLlib [29], and an extra repository with some untested proposals named spark packages [31]. Although the MLlib has the most representative algorithms for machine learning, the set of available algorithms is still limited. At the time of writing this paper, there are few tools for time series processing in MLlib or spark packages, and in general in Big Data environments.

In this work, we propose a scalable and distributed time series transformation based on well-known time series features, named SCMFTS, to provide an alternative vector-based representation of time series that enables the use of the traditional machine learning techniques available in Big Data environments. We have implemented it in Apache Spark through Scala, guaranteeing a fully scalable behavior, being the first proposal of this type made for Big Data environments. The code is publicly available [5]. SCMFTS allows practitioners to face problems that would otherwise be impossible and to improve the results obtained through the additional information provided by the new time series features. The proposed transformation is applicable for univariate and multivariate time series. It has been tested for effective accuracy and linear scalability.

The remainder of this paper is organized as follows. In Section 2, we analyze the works related to our proposal. Section 3 explains the transformation proposed, the time series features selected, and the workflow of our proposal. In Section 4, we summarize the obtained results. Finally, we show the conclusions of our work in Section 5.

2 Related works

In this section, we analyze the state-of-the-art of time series processing in Big Data 2.1 and the main Big Data frameworks 2.2.

2.1 Time series in Big Data

For almost a decade, the processing of enormous amounts of time series has been an active research topic. One of the most representative works tries to process trillions of time series subsequences through the dynamic time warping distance measure [33], which has a high complexity. After this first work, we can see a succession of new proposals that try to face the problem of processing time series to a larger scale. For example, the FastShapelet (FS) algorithm [34] which provides a reduction in time complexity of the original proposal at a cost in accuracy, proposals of a generic and scalable framework for automated anomaly detection to deal with large-scale time series data [23], a fast and scalable Gaussian process modeling oriented to astronomical time series [12], or a scalable distance-based classifier for time series named Proximity Forest [28], show us the growing interest in processing larger and larger sets of time series. However, we can see how the limitations of the traditional computation model and computation systems are also there in these works. Limitations in the available resources to deal with a large problem, becoming impossible its storage in memory or obtaining unacceptable running times, among others.

To face the limitations mentioned above the Distributed FastShapelet Transform (DFST) algorithm [2] has been introduced, the first time series classification algorithm developed in a distributed way. DFST joins the low complexity of the FastShapelet (FS) algorithm with the Shapelet Transform (ST) [26] performance. ST proposes to use the distance between the selected shapelets and each time series in the dataset as the input features. For this reason, we can consider it as a feature based method. The performance of ST depends on the machine learning algorithm used on the transformed dataset, but it provides competitive results concerning the best proposals of the state-of-the-art. In addition, the DFST method lets us apply the traditional vector-based algorithms already available in Apache Spark to time series problems, expanding the tools to process this kind of data. But this approach can only be used in supervised problems.

The time series analysis problem has additional characteristics with respect to the traditional vector-based problems. Characteristics like time dependency, trend, seasonality, or stationarity of a time series, among others, must be considered in the algorithm proposals, raising the complexity of the methods or adding some limitations to them and making the application of the proposed methods in distributed environments impossible. For example, on the one hand, FS analyzes the entire dataset sequentially and provides the best decision tree, based on shapelets founded, evaluating each shapelet with the complete dataset. On the other hand, DFST evaluates the shapelets candidates in a distributed way on the data available in each node and saves the most valuated. This is because the shapelets evaluation process has a high complexity, which makes it impossible to apply it to the complete dataset in Big Data environments.

Following the feature based approach used in DFTS and ST, but without depending on shapelets, we can find multiple works based on extracting time series features. For example, the application of multiple types of mathematical operations over a time series to obtain valuable information from it that explains the underlying structure of their behavior [14][13], the selection from a theoretical point of view of the most representative features of a time series that could explain their behavior [20], or the proposal of a set of 22 characteristics [27] through exhaustive experimentation which guarantee these features as the most representative of the original set of features, among others.

Unsupervised feature extraction has been applied in other domains [32]. In the particular case of time series, recently, it has been demonstrated that a set of well-known complexity measures and time series features is able to provide competitive results concerning the state-of-the-art of univariate [3] and multivariate [4] time series classification. To extrapolate this approach to a distributed Big Data environment is necessary to filter and prepare the selected features to be totally independent of each other and do not require relationships between different time series or additional information. These conditions allow their inclusion in a distributed environment, increasing the limited amount of tools for time series processing in Big Data environments.

2.2 Big Data frameworks

Even with the previous scalable —but not distributed— proposals, some problems are impossible to process when storage needs become untractably large for a single computer. The MapReduce paradigm addresses these issues by proposing a distributed computation model that joins the capabilities of multiple computers to obtain the necessary resources transparently for the user. The MapReduce paradigm is based on two types of operations:

- The *map* operation distributes throughout the cluster the computation needed over each instance of the dataset. This operation is applied independently over each instance, and there is no possible interaction between different instances.
- The *reduce* operation brings to the cluster driver the generated results for a previous map operation. In this case, there are interactions between the different instances of the dataset.

The most popular framework that includes this paradigm was Apache Hadoop [39], written in Java. However, its limitations, such as the necessity of writing to disk every step in the workflow, the impossibility to implement iterative behaviors efficiently, or the necessity to hand code every operation, among others, have led to the emergence of new frameworks, like Apache Spark [16] or Apache Flink [11]. Spark proposes a framework that includes in-memory processing, increasing the processing speed with several orders of magnitude. In addition, Spark introduces the new Resilient Distributed Datasets (RDD) data structure [42] and lazy evaluation. Every transformation and action applied over the RDD is recorded in the RDD lineage. This lineage is a register of each operation applied to the RDD. It allows the RDD to be recovered in any of its previous states, giving a high fault tolerance to the system.

Although Spark provides the framework to process enormous quantities of data and the time series processing is evolving towards processing ever-increasing amounts of data, we cannot find enough tools for processing time series in Spark with official support yet. If we analyze MLlib, we cannot find specific time series algorithms for classification, clustering, or forecasting tasks. In spark-packages [31], we only can find two time series proposals. The first one is the spark-ts[2] package, which provides statistical modeling for time series in a distributed way. However, it is oriented to forecasting tasks, and it has been discontinued for a long time. The second one is the Distributed FastShapelet Transform referred to in the previous section. MLlib includes tools for handling data streaming, but only a streaming linear regression model is available.

[2]`https://spark-packages.org/package/sryza/spark-timeseries`

3 Scalable and distributed time series transformation proposal

In this work, we propose, through a MapReduce framework, a scalable and distributed transformation for univariate and multivariate time series (SCMFTS) based on well-known time series features for Big Data environments. The proposed transformation provides a traditional vector-based representation for time series data. SCMFTS allows us to apply algorithms that are not time series specific to time series problems. Table 6, in Appendix A, contains the selected set of features used in this work.

In sequential and supervised scenarios, the above approach has proven to obtain competitive results concerning the main algorithms of the state-of-the-art [3][4]. In addition, SCMFTS is able to process MTS with multiple frequencies and lengths, allowing practitioners to add new features easily.

Our proposal is based on the extraction of the features of each time series. More formally, it can be stated as follows. Given a time series dataset represented in the temporal domain, a new, vectorial representation of the dataset is obtained as follows:

- Unidimensional case. For each time series in the dataset, all the features are computed. The time series is represented by the vector composed of the values for the computer features. The order of the feature values is the same for all the time series in the dataset.
- Multidimensional case. For each time series in the dataset, for each variable in the multivariate time series, all the features are computed. The individual time series is represented by the vector composed of the values for the computer features. The order of the feature values is the same for all the time series in the dataset. The multivariate time series is represented by a vector formed as the concatenation of the corresponding vectors to each of the individual time series composing the multivariate time series.

This new vector-based representation of time series opens a huge landscape of applicable methods for time series. But it is also rather robust allowing for easy usage of complex methods.

While these transformations follow trivially from our proposal for time series representations in previous papers [3][4], in the current work, we dive deep into its effective application in a Big Data scenario. To effectively study this, we have designed and developed an actual implementation of the representation conversion in a software package able to face real-world problems. This particular implementation is what we term SCMFTS. Its most relevant design issues are detailed in the rest of this section. Its performance and scalability are further analyzed in the remainder of the paper.

Our implementation unifies the distributed computation provided by Spark with the powerful statistical tools available in R to obtain the desired transformation in Big Data environments. We have developed SCMFTS in Scala/Spark, and the communications between Spark and R have been done through rscala [7].

To process the time series correctly, a number of considerations must be made:

- Each multivariate time series has a unique key that identifies it (tsKey).
- Each variable of an MTS has a unique key (varKey). The combination of a time series key with a variable key is unique.
- The input data must have the following format: (tsKey, varKey, tsClass [optional], tsData$_1$, tsData$_2$, ... , tsData$_n$).
- Due to the possible differences between the multiple variables that compose an MTS, we chose to process each variable individually. For example, one variable could have hundreds of data points, but another variable could have thousands of data points. Because the communication between Spark and R must be done through a simple data structure like Array[Array[PrimitiveDataType]], including both variables in the same RDD forces us to fill the shortest time series with 0.0 wasting a vast amount of memory resources.

Figure 1 shows the workflow of SCMFTS for the multivariate case. We have included the class column to illustrate a typical supervised problem. In the case of univariate time series, the process is the same, but we do not need the filter and joins steps. First, the framework reads the data in the correct format or processes it until we obtain this format (step 1). Due to this, we can unequivocally identify each time series and variable. Second, we generate an RDD for each problem variable, filtering the input data by the varKey column (step 2). Next, we process each RDD/Variable in a distributed way, generating a new RDD for each variable with the extracted features (step 3), which are then joined iteratively (step 4), obtaining a final RDD/dataset where each instance contains the ordered extracted features from each variable (step 5).

In Algorithm 1, we show the pseudocode of SCMFTS, utilizing the operations in Scala/Spark and R. In line 1, we initialize a list that will contain the computed RDDs with the time series features. In lines 2 to 5, our proposal obtains

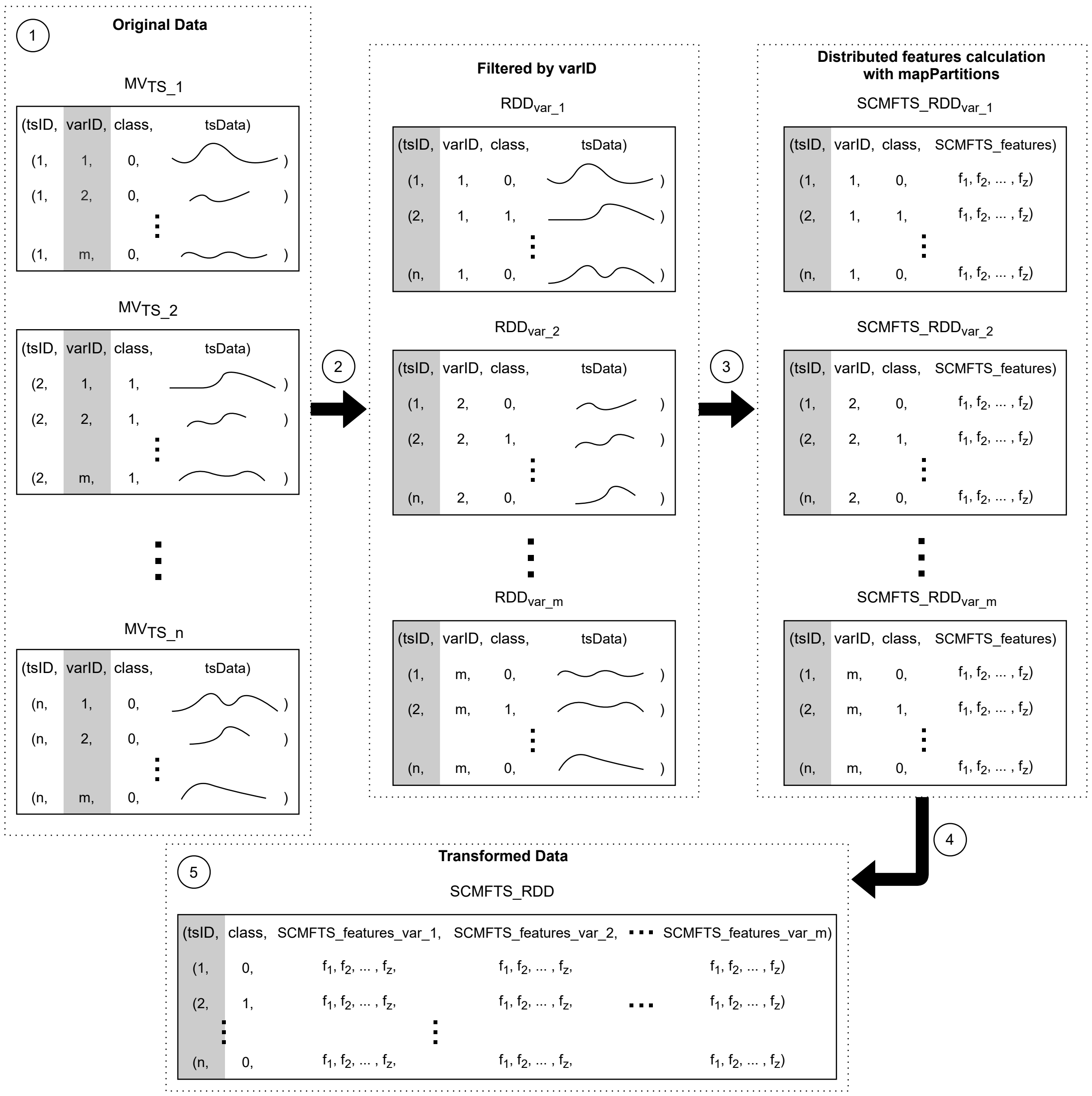

1
Original Data
MVTS_1
(tsID, varID, class, tsData)
(1, 1, 0,)
(1, 2, 0,)
(1, m, 0,)
MVTS_2
(2, 1, 1,)
(2, 2, 1,)
(2, m, 1,)
MVTS_n
(n, 1, 0,)
(n, 2, 0,)
(n, m, 0,)
2
Filtered by varID
RDDvar_1
RDDvar_2
RDDvar_m
3
Distributed features calculation with mapPartitions
SCMFTS_RDDvar_1
SCMFTS_RDDvar_2
SCMFTS_RDDvar_m
(tsID, varID, class, SCMFTS_features)
f1, f2, ... , fz)
4
5
Transformed Data
SCMFTS_RDD
(tsID, class, SCMFTS_features_var_1, SCMFTS_features_var_2, ▪▪▪ SCMFTS_features_var_m)
(1, 0, f1, f2, ... , fz, f1, f2, ... , fz, f1, f2, ... , fz)
(2, 1, f1, f2, ... , fz, f1, f2, ... , fz, ▪▪▪ f1, f2, ... , fz)
(n, 0, f1, f2, ... , fz, f1, f2, ... , fz, f1, f2, ... , fz)

Figure 1: Workflow example of SCMFTS. The shaded column in each case represents the key value of the typical MapReduce paradigm schema (key, value). The remaining unshaded columns belong to the value field.

the input data of each variable that composes the MTS processed. We chose a mapPartition transformation because the initialization of the R environment is a time-consuming process. Using this transformation, we only initialize an R environment for each data partition, processing every time series in that partition in the same R session. In line 6, we initialize the output RDD that will contain, in order, the RDDs with the calculated features for each variable that composes the input MTS. In this first step, we include the feature-RDD of the first variable. In lines 7 to 9, we join, iteratively and in order, the output RDD with each feature-RDD of the rest of the variables. We use the tsID as the unique key for the join, and we remove the varID and class columns because we included them in the first RDD for this output. Finally, in line 10, we return the final RDD that contains the features calculated for every variable that composes each MTS. The output RDD can be used as input of any traditional vector-based machine learning algorithm available in Spark. Naturally, it can also be exported to a CSV-like format so that it can be published or further processed on a different platform.

Algorithm 1 SCMFTS procedure

```
Input:
    $inputData$: list of input RDDs with (tsID, varID, class, tsData) structure
Output:
    $outputData$: output RDD that contains the time series features calculated for each input time series
    with (tsID, class, scmftsFeaturesVars) structure
1: scmftsVars ← [ ]
2: for each variable in inputData do
3:     varData ← inputData.filter(variable==varID)
4:     scmftsVars[variable] ← varData.mapPartitions{rscala::scmfts(varDataPartition)}
5: end for
6: outputRDD ← scmftsVars[first]
7: for each scmftsRDD in scmftsVars[-first] do
8:     outputRDD ← outputRDD.join(scmftsRDD[-varID,-class], by=tsID)
9: end for
10: return (outputRDD)
```

Algorithm 2 Imputation procedure

```
Input:
    $SCMFTS\_RDD$: input RDD that contains the time series features calculated for each input time
    series (tsID, class, scmftsFeaturesVars) structure
Output:
    $imputedSCMFTSDataset$: output RDD with (tsID, class, scmftsFeaturesVars) structure without
    the non-desirable values
1: for each column in SCMFTS_RDD[-tsID, -class] do
2:     for each value in SCMFTS_RDD[ , column.index] do
3:         if is.infinite(value) then
4:             if value $\geq 0$ then
5:                 value ← max(SCMFTS_RDD[ , column.index], ignore.inf)
6:             else
7:                 value ← min(SCMFTS_RDD[ , column.index], ignore.inf)
8:             end if
9:         end if
10:     end for
11:     column ← imputeMean(SCMFTS_RDD[ , column.index])
12: end for
13: imputedSCMFTSDataset ← SCMFTS_RDD
14: return (imputedSCMFTSDataset)
```

Sometimes, the time series features used are not present in the processed time series. For example, a time series without a seasonal pattern or too short to calculate an autocorrelation coefficient cannot offer values for these features. To face this problem, we have included an imputation process after the application of Algorithm 1, explained in Algorithm 2. This procedure grants our proposal a robust behavior to a wide variety of scenarios such as time series that are too short, non-seasonal, trendless, among others.

In lines 1-12, we focus our imputation process on the time series features columns. For each value in each column, we identify if this value is infinite or -infinite, replacing this value by the maximum or minimum finite value of the column, respectively (lines 2-10). After this imputation, the rest of the non-desirable values are imputed using the mean of the column, ignoring the non-desirable values in the mean calculation of each column (line 11). Finally, we obtain the final RDD without non-desirable values (lines 13-14).

In summary, the proposal computes multiple features out of MTS in a robust and scalable way thanks to its MapReduce workflow, whose design, combined with an imputation strategy, enables the tackling of MTS with different lengths and frequencies for each variable with no memory nor runtime overhead, resulting in a fixed length vector for each MTS. The combination of R, Scala, and Spark ensures the efficiency, failure tolerance, and extensibility of the software.

4 Empirical study

In this section, we conduct an empirical evaluation of the performance and scalability of SCMFTS. Section 4.1 details the empirical evaluation design. In Section 4.2, we show the results obtained by SCMFTS and the selected proposals for comparison.

4.1 Experimental design

With the aim to evaluate the scalability and performance of SCMFTS, we thoroughly designed and rigorously conducted an empirical analysis. Section 4.1.1 includes the motivation of the selected dataset along with their description. Section 4.1.2 specifies the measures used to evaluate our proposal and the methodology performed. In Section 4.1.3, we describe the models to which SCMFTS is compared. Finally, Section 4.1.4 includes the hardware and software used for the experiments.

4.1.1 Datasets

To evaluate the performance of SCMFTS in a real world, we have selected the multivariate time series dataset with the highest number of instances in the UCI Repository [9], the Wearable Stress and Affect Detection (WESAD) dataset [37]. In this dataset, we try to identify the patient's state through 14 different sensors that measure biological parameters of the subject, such as blood volume pulse or respiration, among others, and its movement through acceleration sensors in the x, y, and z axes. We have information from 15 different patients. WESAD dataset is used to address two different problems:

- The first problem tries to differentiate three states: baseline vs. stress vs. amusement.
- The second problem differentiates stress vs. non-stress states, joining baseline and amusement states under the same label non-stress.

For data extraction, we followed the segmentation process with a sliding window explained by the original authors: window shift of 0.25 seconds, a window size of 5 seconds for the ACC signals, and 60 seconds for the physiological signals. For subject 14, we also removed the first 136 time series because they contain multiple missing values. In this way, we have generated 132,477 MTS composed of 14 variables each. Table 1 shows the names of each variable and its most representative characteristics.

Table 1: MTS characteristics of the WESAD dataset

Variable	Time Series length	Frequency (Hz)
c_ACCx	3,500	700
c_ACCy	3,500	700
c_ACCz	3,500	700
c_ECG	42,000	700
c_EDA	42,000	700
c_EMG	42,000	700
c_RESP	42,000	700
c_TEMP	42,000	700
w_ACCx	160	32
w_ACCy	160	32
w_ACCz	160	32
w_BVP	3,840	64
w_EDA	240	4
w_TEMP	240	4

To better evaluate the scalability, a larger dataset is necessary. Since there is no one publicly available, we have created a synthetic dataset composed of MTS with three variables, with 100 data points per variable. We have generated two different sets of time series:

- The first set of ten datasets containing from 100,000 to 1,000,000 MTS with increments of 100,000 MTS between each dataset.

- A second set of 10 datasets contained from 1,000,000 to 10,000,000 MTS with increments of 1,000,000 MTS between each dataset

The problem generated has four classes obtained from combining random walk processes with AutoRegressive Integrated Moving Average (ARIMA) models in different ways. In Table 2, we show the setup for the variables of these four classes. For variable one, we use an ARIMA(0,1,0) model to simulate a random walk process. Variable two is composed of different ARIMA models, and variable three contains combinations of variables one and two through multiple functions.

Table 2: Setup for four class problem

Class	Variable 1	Variable 2	Variable 3
0	ARIMA(0,1,0)	ARIMA(1,1,1): AR(0.5), MA(0.5)	sin(var1+var2)
1	ARIMA(0,1,0)	ARIMA(1,1,1): AR(0.25), MA(0.5)	cos(var1+var2)
2	ARIMA(0,1,0)	ARIMA(2,1,2): AR(0.2,0.1), MA(0.1,0.1)	sin(var1)+cos(var2)
3	ARIMA(0,1,0)	ARIMA(2,1,2): AR(0.5,0.25), MA(0.1,0.1)	cos(var1)+sin(var2)

4.1.2 Measures and methodology

To evaluate the performance of our proposal, we compare SCMFTS in two and three class sub-problems from the WESAD dataset against the original work using all the available variables, using as much data as possible due to the Big Data approach of our proposal. Although a number of papers with algorithmic proposals working on varied versions of the dataset have been published [6][19][25][35][36], they do not follow the original segmentation process nor the Leave-One-Subject-Out (LOSO) Cross-Validation (CV) approach. Due to this, their results are not comparable with those reported in the original work. We have opted to follow the specifications provided in [37], using the same measures and the same validation criteria. To compare the different models, we use the accuracy and F1-score [10]. The accuracy is obtained by dividing the number of correctly classified instances by the total number of instances in the test set. The F1-score is defined as the harmonic mean of model precision and recall, and it is obtained by combining model precision and recall. F1-score represents a measure of thoroughness. We have evaluated all models using the LOSO CV procedure.

The scalability of SCMFTS is evaluated through two different approaches. First, we measure runtimes by increasing the number of processed time series. In this case, we seek to obtain a linear relationship between the runtimes and the number of processed time series. Secondly, we will measure runtimes by varying the number of workers in the cluster and the number of cores per worker independently. In both cases, we will compare the evolution of runtimes with the ideal and unachievable case of scalability expressed by Amdahl's law [18].

4.1.3 Models

Since the main target of our proposal is Big Data scenarios, we have focused our experimentation on all the available variables to process as much data as possible. The study in [37] used the following models: Decision Tree (DT), Random Forest (RF), AdaBoost (AB), Linear Discriminant Analysis (LDA)[3], and K-Nearest Neighbors (KNN) with k=9. For the (DT, RF, AB) models, the referred to work set the minimum number of samples required to split a node to 20, and they set the number of base estimators to 100 for (RF and AB). To ensure a fair comparison and reproducibility, we have chosen the models available in the MLlib of Apache Spark (DT, RF, KNN). In addition, we have used the same hyperparameters specified in [37]. Some of these models have additional parameters that were not specified in the mentioned work. All hyperparameters used in our experimentation are listed in Table 3.

Table 3: Hyperparameters for used models

Model	Setup
DT	MaxDepth=10, MinInstancesPerNode=20, Seed=1
RF	MaxDepth=10, MinInstancesPerNode=20, numTrees=100, maxBins=32, Seed=1
KNN	K=9, Euclidean distance, Normalized data with mean 0 and standard deviation 1

[3]The LDA method available in Spark corresponds to Latent Dirichlet Allocation and is not related to the Linear Discriminant Analysis method used in WESAD work.

To maintain the number of data partitions over the entire process, we have set this number in all experiments performed to three times the number of total available cores, which is the typical recommendation for Spark.

Considering the LOSO approach used in [37], we have applied the imputation process explained in Algorithm 2 to each subject data independently. Due to the data source of the WESAD dataset, we have an additional column that contains the identification number of the subject that provides each time series. Using this information, we can filter the data of each subject and process it independently. Obviously, this value is not used within the training datasets.

4.1.4 Hardware and software

We have performed the experimentation in a Big Data cluster composed of one driver/master node and 17 slave/computing nodes. The computing nodes hold the following characteristics: 2 × Intel(R) Xeon(R) CPU E5-2620 processors, 6 cores per processor with HyperThreading, 2.00 GHz, 64 GB RAM, 2 TB HDD (1 TB HDFS). We have used the following software configuration: Ubuntu 18.04.5 LTS, Apache Hadoop 2.7, Apache Spark 3.0.1, 19 threads/node, 833 RAM GB (48 GB/node).

The source code of SCMFTS is publicly available [5].

4.2 Results

In this section, we evaluate the two main aspects of SCMFTS: performance and scalability. Section 4.2.1 includes the performance results of SCMFTS on the WESAD dataset. In Section 4.2.2, we analyze the scalability of SCMFTS on the synthetic dataset.

4.2.1 Performance results on WESAD

To assess the performance of the SCMFTS proposal, we have applied it to solve the two- and three-classes problems of WESAD and compared it to the ML procedures analyzed in the original work. The empirical results, expressed in terms of accuracy and F1-score, are displayed in Table 4.

The results show that SCMFTS provides consistently better results than the WESAD work, with the available Spark machine learning models, in both cases. In addition, we can see in the WESAD work [37], Table 3, that the best results for the three classes problem are provided by the AB model but only using the chest physiological modalities (c_ECG, c_EDA c_EMG, c_RESP, and c_TEMP): 80.34 of accuracy and 72.51 of F1-score, and still SCMFTS+RF provides better results. In the two classes problem, the WESAD work provides the LDA model with the chest physiological modalities as the best results with 93.12 of accuracy and 91.47 of F1-score, Table 4 in WESAD work [37], outperforming in accuracy our best model but not in F1-score. In this case, the WESAD work uses fewer variables and a model that is not available in Spark, so we cannot make a direct comparison. It is important to note that the LDA model provides the best results in 20 out of 32 cases in the WESAD work, so this particular model clearly offers better results than the others. In addition, the AB model in the three-class problem and the LDA model in the two-class problem provide results significantly better than DT, RF, and KNN models in the WESAD work.

Table 4: Accuracy and F1-score results for WESAD dataset

	Three classes		Two classes	
	Accuracy	F1-Score	Accuracy	F1-Score
WESAD proposal:				
DT	63.56	58.05	83.60	80.83
RF	74.97	64.08	87.74	85.71
AB	79.57	68.85	87.00	83.88
LDA	75.80	71.56	92.28	90.74
KNN (k=9)	56.14	48.70	74.20	69.14
SCMFTS:				
DT	64.08	62.69	85.38	84.71
RF	**81.62**	**77.16**	**92.67**	**91.96**
KNN (k=9)	77.78	75.79	89.89	89.90

There are relevant differences between the multiple variables that compose the MTS of this problem. Due to the segmentation applied to the original time series proposed in the WESAD work, we have variables that contain from 160 to 42,000 data points in the same problem. These differences between variables generate significant variations in runtime for feature calculation between the different variables, as we can see in Table 5. As usual, a high number of data points generates high runtimes, but if we compare runtimes for variables c_ACCx, c_ACCy, or c_ACCz with w_BVP, this does not happen. It is so because of the differences in the frequency value of these variables, which is included in the time series features calculation affecting the runtime. These phenomena are not related to the Spark implementation performed, but it depends on the structure of the input time series.

Table 5: Run times for WESAD dataset

Variable	Time SCMFTS (secs)	Time Series length	Frequency (Hz)
c_ACCx	990.51	3,500	700
c_ACCy	1008.76	3,500	700
c_ACCz	994.42	3,500	700
c_ECG	23,007.39	42,000	700
c_EDA	22,727.54	42,000	700
c_EMG	22,263.16	42,000	700
c_RESP	23,639.61	42,000	700
c_TEMP	22,396.20	42,000	700
w_ACCx	151.24	160	32
w_ACCy	150.72	160	32
w_ACCz	145.75	160	32
w_BVP	813.51	3,840	64
w_EDA	242.28	240	4
w_TEMP	244.43	240	4
All variable join:	183.93		
Total Time:	118,959.47		

4.2.2 Scalability results on synthetic dataset

In this section, we analyze the scalability performance of SCMFTS. Particularly, we focus on the three most frequently considered dimensions: number of instances to process, number of machines available, and number of cores per machine. In Figure 2, we show the relationship between the runtime of SCMFTS and the number of MTS to process. For this experimentation, we have used the 17 available workers and 19 cores/threads per worker. Figure 2 allows us to graphically identify a linear relationship between the runtime and the number of time series to be processed through SCMFTS. This feature is a mandatory requirement for the scalability considerations in Big Data environments.

In Figure 3, we can compare the runtime of SCMFTS with different numbers of workers. In this case, we perform the experimentation with a dataset composed of 1,000,000 MTS. As usual, an increase in the number of workers entails a reduction in the runtime. For example, if we compare the one and three workers cases, we can appreciate that the reduction obtained is close to three times. This behavior is present in the rest of the comparison in the Figure 3. In this kind of process, the ideal case is to obtain a time reduction equal to the number of the added workers as Amdahl's law [18] specifies, but it is a theoretical limit and in general impossible to achieve in practice. SCMFTS is near to the optimal case. Furthermore, we have to note the existence of additional procedures related to adding workers to the cluster, like extra workers communications, data transmission, among others, that do not let us reach the performance of the ideal case.

Each worker has 12 real/physical cores, 24 with hyper-threading technology. To evaluate the core's performance of SCMFTS, we process 1,000,000 MTS using the 17 workers available in our cluster, but we vary the number of used cores in each worker. Figure 4 shows the relationship between the runtime of SCMFTS and the number of cores per worker. We can appreciate that the reduction in runtime has similar behavior to the one observed previously, Figure 3, but with a greater gap regarding the optimal case. If we compare the case of one core per worker with the cases of three or five cores per worker, among others, we can see that the amount of runtime reduction is directly related to the number of cores used: one core case has a runtime close to 25,000 seconds, three cores case has a runtime close to 7,500 seconds, five cores case has a runtime slightly higher to 5,000 seconds, etc. But in this case, we can appreciate that the runtime stops decreasing with the number of cores with 11 cores per worker. This issue is due to the number of physical cores by machine, which is 12. Although hyper-threading technology allows us to increase the efficiency of a core to provide an additional virtual core, we cannot reach the maximum desired performance in computationally intensive tasks.

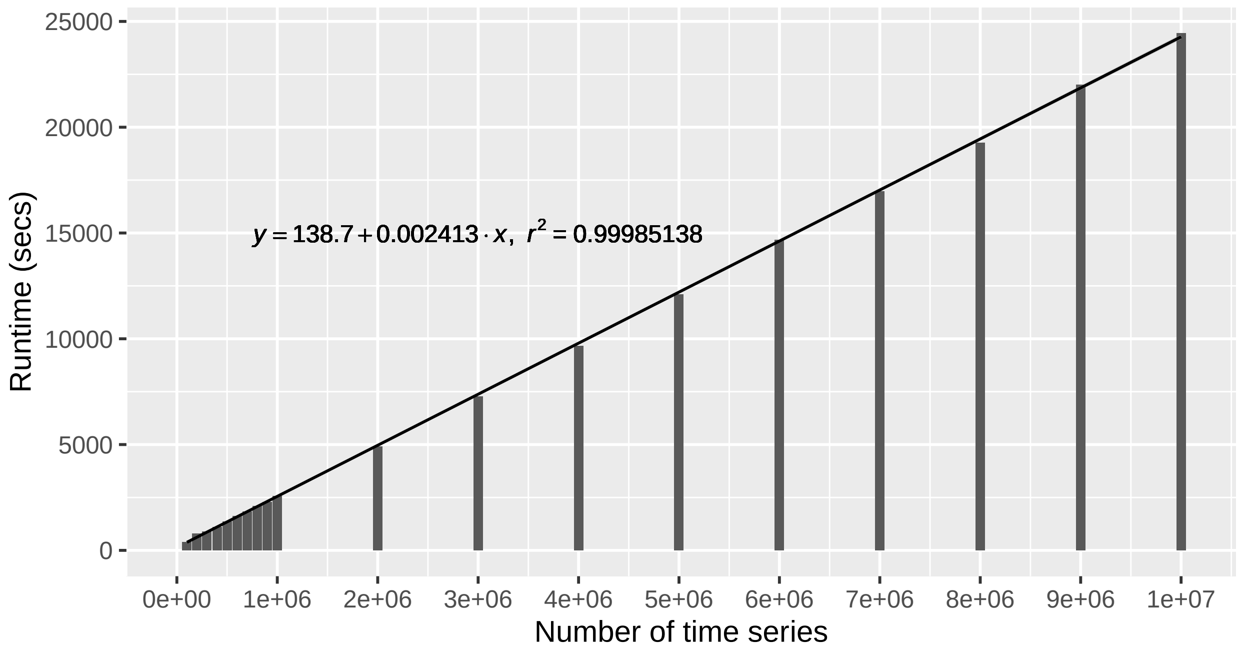


Figure 2: Scalability experiment: Runtime vs Number of time series

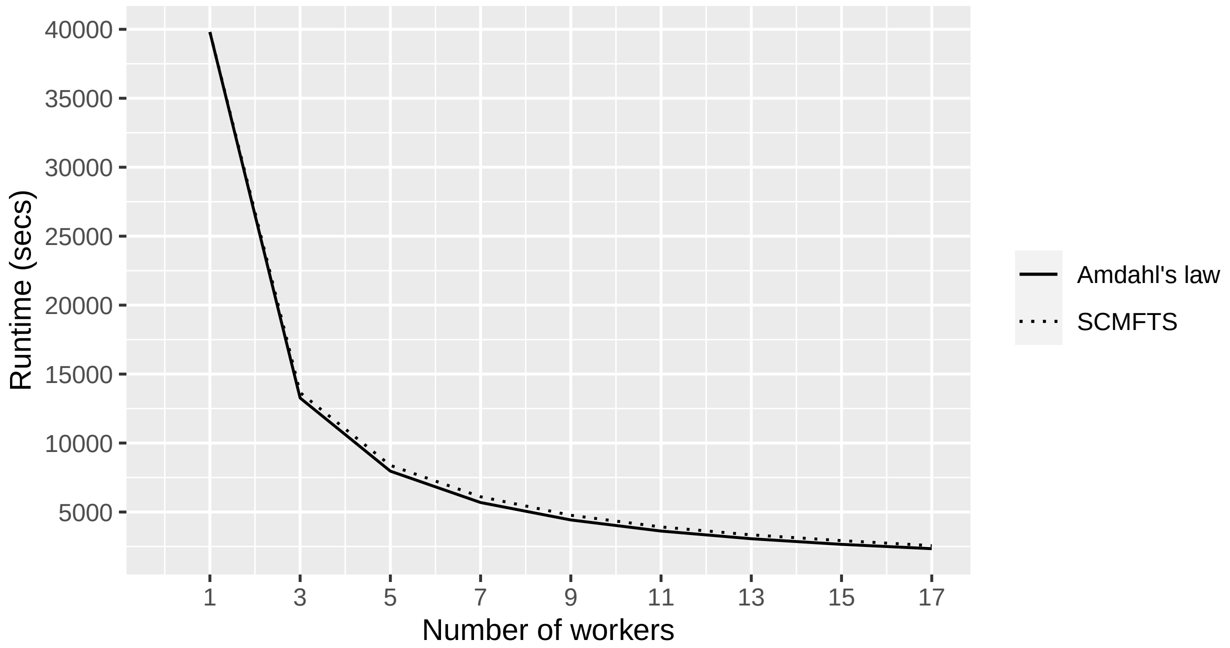


Figure 3: Scalability experimentation: Runtime vs Number of workers

Based on the results obtained in this section, in which our proposal shows behaviors close to the ideal, we can conclude that SCMFTS has a high scalability performance.

5 Conclusions

In this paper, we have presented a scalable and distributed method, named SCMFTS, for transforming univariate and multivariate time series into a vector of well-known features. This method lets us apply the traditional vector-based

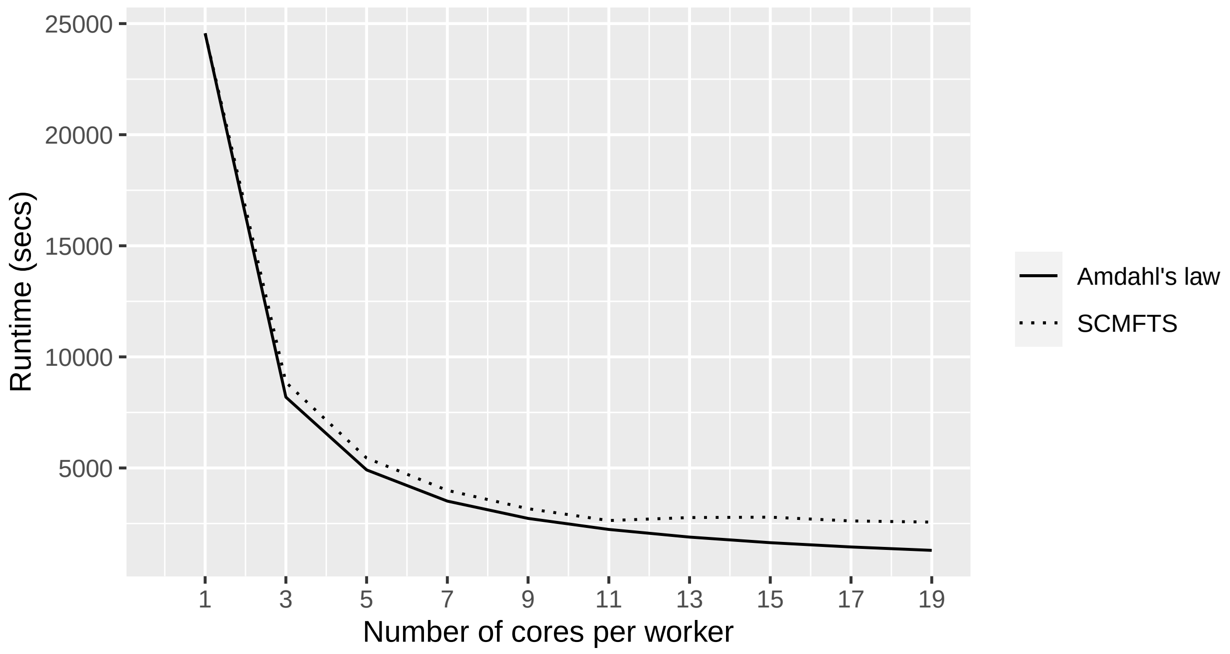


Figure 4: Scalability experiment: Runtime vs Number of cores per worker

algorithms already available in Big Data to time series problems, allowing us to address problems that would otherwise be impossible. SCMFTS extends considerably the limited number of algorithms available to process time series in Big Data environments. Our proposal is able to process MTS with multiple frequencies and lengths and allows practitioners to add new features easily.

SCMFTS has improved the results obtained, under the same conditions, by the state-of-the-art on the biggest multivariate time series dataset available in the UCI Machine Learning Repository, Wearable Stress and Affect Detection (WESAD). The results obtained by SCMFTS on a general problem improved those obtained by the WESAD work solution, applying the proposed transformation without additional considerations and allowing it to be a tool of interest to a large number of researchers in multiple areas. In addition, SCMFTS has shown a totally scalable behavior through exhaustive experimentation, with a linearly scalable relationship in runtime concerning the number of time-series processed.

Our proposal has been implemented in the Scala programming language for the Apache Spark framework, and the code is publicly available. The implementation of SCMFTS has followed the principles of FAIR [40] (Findability, Accessibility, Interoperability, and Reuse) and Open Science.

This proposal opens promising research lines in this topic, as exploring the semi-supervised approach based on the proposed set of features. In Big Data environments, the volume of processed data is high, and the labeling is limited. In those environments, the semi-supervised approach offers very interesting solutions. Another research line is the study of the improvement in the expressivity and performance of the selected set of features in Big Data environments.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

The authors consent to this work for publication.

Availability of data and material

The code of our proposal and synthetic time series generation is publicly available (`https://github.com/fjbaldan/SCMFTS/`). The Wearable Stress and Affect Detection dataset is publicly available already (`https://ubicomp.eti.uni-siegen.de/home/datasets/icmi18/`).

Competing interests

The authors declare that they have no conflict of interest.

Funding

This research has been partially funded by the following grants: TIN2016-81113-R from the Spanish Ministry of Economy and Competitiveness, P12-TIC-2985 and P18-TP-5168 from Andalusian Regional Government, Spain, and EU Commission with FEDER funds. Francisco J. Baldán holds the FPI grant BES-2017-080137 from the Spanish Ministry of Economy and Competitiveness. D. Peralta is a Postdoctoral Fellow of the Research Foundation of Flanders (170303/12X1619N). Y. Saeys is an ISAC Marylou Ingram Scholar.

Authors' contributions

Paper writing and experiment implementation, FJB; Methodology, FJB and DP; Revising and reviewing, DP, JMB, YS; Supervision, DP, JMB, YS; Funding acquisition, JMB and YS.

Acknowledgements

This work has been partially carried out while Francisco J. Baldán was on a research stay at the University of Ghent. Francisco J. Baldán is grateful for the valuable feedback and knowledge received from the members of the Data Mining and Modeling for Biomedicine (DAMBI) group.

References

[1] S. Aarthy, J. M. Iqbal, Time series real time naive bayes electrocardiogram signal classification for efficient disease prediction using fuzzy rules, Journal of Ambient Intelligence and Humanized Computing 12 (5) (2021) 5257–5267.

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

[3] F. J. Baldán, J. M. Benítez, Complexity Measures and Features for Times Series classification, arXiv preprint arXiv:2002.12036.

[4] F. J. Baldán, J. M. Benítez, Multivariate times series classification through an interpretable representation, Information Sciences 569 (2021) 596–614.

[5] F. J. Baldán, D. Peralta, Y. Saeys, J. M. Benítez, Scalable Complexity Measures and Features for Times Series classification package repository, `https://github.com/fjbaldan/SCMFTS/` (2021).

[6] P. Bobade, M. Vani, Stress detection with machine learning and deep learning using multimodal physiological data, in: 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 51–57.

[7] D. B. Dahl, Integration of R and Scala Using rscala, Journal of Statistical Software 92 (1) (2020) 1–18.

[8] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, in: Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6, 2004, p. 10.

[9] D. Dua, C. Graff, UCI machine learning repository (2017).
URL `http://archive.ics.uci.edu/ml`

[10] R. P. Espíndola, N. F. Ebecken, On extending f-measure and g-mean metrics to multi-class problems, WIT Transactions on Information and Communication Technologies 35.

[11] A. Flink, Apache Flink, `http://flink.apache.org/` (2019).

[12] D. Foreman-Mackey, E. Agol, S. Ambikasaran, R. Angus, Fast and scalable Gaussian process modeling with applications to astronomical time series, The Astronomical Journal 154 (6) (2017) 220.

[13] B. D. Fulcher, Feature-based time-series analysis, arXiv preprint arXiv:1709.08055.

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

[15] Z. Haddi, B. Ananou, Y. Trardi, J. F. Pons, S. Delliaux, J. C. Deharo, M. Ouladsine, Advanced machine learning coupled with heart-inter-beat derivatives for cardiac arrhythmia detection, in: 2020 American Control Conference (ACC), 2020, pp. 5433–5438.

[16] M. Hamstra, H. Karau, M. Zaharia, A. Konwinski, P. Wendell, Learning Spark: Lightning-Fast Big Data Analytics (2015).

[17] T. Handhika, Murni, D. P. Lestari, I. Sari, Multivariate time series classification analysis: State-of-the-art and future challenges, in: IOP Conference Series: Materials Science and Engineering, vol. 536, 2019, p. 012003.

[18] M. D. Hill, M. R. Marty, Amdahl's law in the multicore era, Computer 41 (7) (2008) 33–38.

[19] F. I. Indikawati, S. Winiarti, Stress detection from multimodal wearable sensor data, in: IOP Conference Series: Materials Science and Engineering, vol. 771, 2020, p. 012028.

[20] Y. Kang, R. J. Hyndman, F. Li, et al., Efficient generation of time series with diverse and controllable characteristics, Tech. rep., Monash University, Department of Econometrics and Business Statistics (2018).

[21] T.-Y. Kim, S.-B. Cho, Predicting the household power consumption using cnn-lstm hybrid networks, in: Intelligent Data Engineering and Automated Learning – IDEAL 2018, 2018, pp. 481–490.

[22] A. Kobusińska, C. Leung, C.-H. Hsu, R. S., V. Chang, Emerging trends, issues and challenges in Internet of Things, Big Data and cloud computing, Future Generation Computer Systems 87 (2018) 416–419.

[23] N. Laptev, S. Amizadeh, I. Flint, Generic and scalable framework for automated time-series anomaly detection, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 1939–1947.

[24] S. W. Lee, H. Y. Kim, Stock market forecasting with super-high dimensional time-series data using ConvLSTM, trend sampling, and specialized data augmentation, Expert Systems with Applications 161 (2020) 113704.

[25] J. Lin, S. Pan, C. S. Lee, S. Oviatt, An explainable deep fusion network for affect recognition using physiological signals, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 2069–2072.

[26] J. Lines, L. M. Davis, J. Hills, A. Bagnall, A shapelet transform for time series classification, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 289–297.

[27] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, N. S. Jones, catch22: CAnonical Time-series CHaracteristics, Data Mining and Knowledge Discovery 33 (6) (2019) 1821–1852.

[28] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, G. I. Webb, Proximity forest: an effective and scalable distance-based classifier for time series, Data Mining and Knowledge Discovery 33 (3) (2019) 607–635.

[29] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., Mllib: Machine learning in apache spark, The Journal of Machine Learning Research 17 (1) (2016) 1235–1241.

[30] T. Nguyen, T. Nguyen, B. M. Nguyen, G. Nguyen, Efficient time-series forecasting using neural network and opposition-based coral reefs optimization, International Journal of Computational Intelligence Systems 12 (2) (2019) 1144–1161.

[31] S. Packages, 3rd Party Spark Packages, `https://spark-packages.org/` (2019).

[32] D. Peralta, Y. Saeys, Robust unsupervised dimensionality reduction based on feature clustering for single-cell imaging data, Applied Soft Computing 93 (2020) 106421.

[33] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh, Searching and mining trillions of time series subsequences under dynamic time warping, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 262–270.

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

[35] A. Saeed, F. D. Salim, T. Ozcelebi, J. Lukkien, Federated Self-Supervised Learning of Multisensor Representations for Embedded Intelligence, IEEE Internet of Things Journal 8 (2) (2020) 1030–1040.

[36] S. Samyoun, A. Sayeed Mondol, J. A. Stankovic, Stress detection via sensor translation, in: 2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2020, pp. 19–26.

[37] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, K. Van Laerhoven, Introducing wesad, a multimodal dataset for wearable stress and affect detection, in: Proceedings of the 20th ACM international conference on multimodal interaction, 2018, pp. 400–408.

[38] J. L. Viegas, N. M. Cepeda, S. M. Vieira, Electricity fraud detection using committee semi-supervised learning, in: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–6.

[39] T. White, Hadoop: The definitive guide (2012).

[40] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The fair guiding principles for scientific data management and stewardship, Scientific data 3 (1) (2016) 1–9.

[41] B. Wu, T. Duan, A performance comparison of neural networks in forecasting stock price trend, International Journal of Computational Intelligence Systems 10 (1) (2017) 336–346.

[42] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, I. Stoica, Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, in: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), 2012, pp. 15–28.

Appendix A Time series complexity measures and features selected

Table 6: Time series complexity measures and features selected

Fea.	Name	Description
f_1	lempel_ziv	LempelZiv (LZA)
f_2	aproximation_entropy	Aproximation Entropy
f_3	sample_entropy	Sample Entropy (DK Lake in Matlab)
f_4	permutation_entropy	Permutation Entropy (tsExpKit)
f_5	shannon_entropy_CS	Chao-Shen Entropy Estimator
f_6	shannon_entropy_SG	Schurmann-Grassberger Entropy Estimator
f_7	spectral_entropy	Spectral Entropy
f_8	nforbiden	Number of forbiden patterns
f_9	kurtosis	Kurtosis, the “tailedness" of the probability distribution
f_{10}	skewness	Skewness, asymmetry of the probability distribution
f_{11}	x_acf1	First autocorrelation coefficient
f_{12}	x_acf10	Sum of squares of the first 10 autocorrelation coefficients
f_{13}	diff1_acf1	Differenced series first autocorrelation coefficients
f_{14}	seas_acf1	First autocorrelation coefficient of the seasonal component
f_{15}	diff1_acf10	Differenced series sum of squares of the first 10 autocorrelation coefficients
f_{16}	diff2_acf1	Twice differenced series first autocorrelation coefficients
f_{17}	diff2_acf10	Twice differenced series sum of squares of the first 10 autocorrelation coefficients
f_{18}	max_kl_shift	Maximum shift in Kullback-Leibler divergence between two consecutive windows
f_{19}	time_kl_shift	Instant of time in which the Maximum shift in Kullback-Leibler divergence between two consecutive windows is located
f_{20}	outlierinclude_mdrmd	Calculates the median of the medians of the values, while adding more outliers
f_{21}	max_level_shift	Maximum mean shift between two consecutive windows
f_{22}	time_level_shift	Instant of time in which the maximum mean shift between two consecutive windows is located
f_{23}	ac_9	Autocorrelation at lag 9
f_{24}	crossing_points	The number of times a time series crosses the median line
f_{25}	max_var_shift	Maximum variance shift between two consecutive windows
f_{26}	time_var_shift	Instant of time in which the maximum variance shift between two consecutive windows is located
f_{27}	nonlinearity	Modified statistic from Teräsvirta’s test
f_{28}	embed2_incircle	Proportion of points inside a given circular boundary in a 2-d embedding space
f_{29}	spreadrandomlocal_meantaul	Mean of the first zero-crossings of the autocorrelation function in each segment of the 100 time-series segments of length l selected at random from the original time series
f_{30}	flat_spots	Maximum run length within any single interval obtained from the ten equal-sized intervals of the sample space of a time series
f_{31}	x_pacf5	Sum of squares of the first 5 partial autocorrelation coefficients
f_{32}	seas_pacf	Sum of squares of the first 5 partial autocorrelation of the seasonal component
f_{33}	diff1x_pacf5	Differenced series sum of squares of the first 5 partial autocorrelation coefficients
f_{34}	diff2x_pacf5	Twice differenced series sum of squares of the first 5 partial autocorrelation coefficients
f_{35}	firstmin_ac	Time of first minimum in the autocorrelation function
f_{36}	std1st_der	Standard deviation of the first derivative of the time series
f_{37}	stability	Stability variance of the means
f_{38}	firstzero_ac	First zero crossing of the autocorrelation function
f_{39}	trev_num	The numerator of the trev function, a normalized nonlinear autocorrelation, with the time lag set to 1
f_{40}	alpha	Smoothing parameter for the level-alpha of Holt’s linear trend method
f_{41}	beta	Smoothing parameter for the trend-beta of Holt’s linear trend method
f_{42}	nperiods	Number of seasonal periods (1 for no seasonal data)
f_{43}	seasonal_period	Seasonal periods (1 for no seasonal data)
f_{44}	trend	Strength of trend
f_{45}	spike	Spikiness variance of the leave-one-out variances of the remainder component
f_{46}	linearity	Linearity calculated based on the coefficients of an orthogonal quadratic regression
f_{47}	curvature	Curvature calculated based on the coefficients of an orthogonal quadratic regression
f_{48}	e_acf1	First autocorrelation coefficient of the remainder component
f_{49}	seasonal_strength	Strength of seasonal
f_{50}	peak	Strength of peaks
f_{51}	trough	Strength of trough
f_{52}	e_acf10	Sum of the first then squared autocorrelation coefficients
f_{53}	walker_propcross	Fraction of time series length that walker crosses time series
f_{54}	hurst	Long-memory coefficient

f_{55}	unitroot_kpss	Statistic for the KPSS unit root test with linear trend and lag one
f_{56}	histogram_mode	Calculates the mode of the data vector using histograms with 10 bins (It is possible to select a different number of bins)
f_{57}	unitroot_pp	Statistic for the PP unit root test with constant trend and lag one
f_{58}	localsimple_taures	First zero crossing of the autocorrelation function of the residuals from a predictor that uses the past trainLength values of the time series to predict its next value
f_{59}	lumpiness	Lumpiness variance of the variance
f_{60}	motiftwo_entro3	Entropy of words in the binary alphabet of length 3. The binary alphabet is obtained as follows: Time-series values above its mean are given 1, and those below the mean are 0

Bibliography

[ANGAB$^+$21] Al Nahian M. J., Ghosh T., Al Banna M. H., Aseeri M. A., Uddin M. N., Ahmed M. R., Mahmud M., and Kaiser M. S. (2021) Towards an accelerometer-based elderly fall detection system using cross-disciplinary time series features. *IEEE Access* 9: 39413–39431.

[Bay17] Baydogan M. (2017) Multivariate Time Series Classification Datasets. `http://www.mustafabaydogan.com`. Accessed: 2020-07-01.

[BB19] Baldán F. J. and Benítez J. M. (2019) Distributed FastShapelet Transform: a Big Data time series classification algorithm. *Information Sciences* 496: 451–463.

[BC94] Berndt D. J. and Clifford J. (1994) Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press.

[BDHL12] Bagnall A., Davis L., Hills J., and Lines J. (2012) Transformation based ensembles for time series classification. In *Proceedings of the 2012 SIAM international conference on data mining*, pp. 307–318. SIAM.

[BDL$^+$18] Bagnall A., Dau H. A., Lines J., Flynn M., Large J., Bostrom A., Southam P., and Keogh E. (2018) The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075* .

[BGL$^+$19] Bondu A., Gay D., Lemaire V., Boullé M., and Cervenka E. (2019) FEARS: a Feature and Representation Selection approach for Time Series Classification. In *Asian Conference on Machine Learning*, pp. 379–394.

[BJ14] Bagnall A. and Janacek G. (2014) A run length transformation for discriminating between auto regressive time series. *Journal of classification* 31(2): 154–178.

[BLB$^+$17] Bagnall A., Lines J., Bostrom A., Large J., and Keogh E. (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery* 31(3): 606–660.

[BM15] Barak S. and Modarres M. (2015) Developing an approach to evaluate stocks by forecasting effective features with data mining methods. *Expert Systems with Applications* 42(3): 1325–1339.

[BR15] Baydogan M. G. and Runger G. (2015) Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery* 29(2): 400–422.

[BR16] Baydogan M. G. and Runger G. (2016) Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery* 30(2): 476–509.

[BYAV13] Bengio Y., Yao L., Alain G., and Vincent P. (2013) Generalized denoising auto-encoders as generative models. In *Advances in neural information processing systems*, pp. 899–907.

[CBNKL18] Christ M., Braun N., Neuffer J., and Kempa-Liehr A. W. (2018) Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing* 307: 72–77.

[CKE$^+$15] Carbone P., Katsifodimos A., Ewen S., Markl V., Haridi S., and Tzoumas K. (2015) Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 36(4).

[Cry86] Cryer J. D. (1986) *Time series analysis*, volumen 286. Springer.

[CTTY13] Chen H., Tang F., Tino P., and Yao X. (2013) Model-based kernel for efficient time series analysis. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 392–400. ACM.

[CVA19] Chauhan S., Vig L., and Ahmad S. (2019) ECG anomaly class identification using LSTM and error profile modeling. *Computers in biology and medicine* 109: 14–21.

[DBH18] Došilović F. K., Brčić M., and Hlupić N. (2018) Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 0210–0215.

[DG04] Dean J. and Ghemawat S. (2004) Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI'04, page 10. USENIX Association, USA.

[DKK$^+$18] Dau H. A., Keogh E., Kamgar K., Yeh C.-C. M., Zhu Y., Gharghabi S., Ratanamahatana C. A., Yanping, Hu B., Begum N., Bagnall A., Mueen A., Batista G., and Hexagon-ML (October 2018) The ucr time series classification archive. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[DPW20] Dempster A., Petitjean F., and Webb G. I. (2020) ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* 34(5): 1454–1495.

[DRTV13] Deng H., Runger G., Tuv E., and Vladimir M. (2013) A time series forest for classification and feature extraction. *Information Sciences* 239: 142–153.

[DZY$^+$17] Deb C., Zhang F., Yang J., Lee S. E., and Shah K. W. (2017) A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews* 74: 902–924.

[FFW$^+$19] Fawaz H. I., Forestier G., Weber J., Idoumghar L., and Muller P.-A. (2019) Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33(4): 917–963.

[FJ14] Fulcher B. D. and Jones N. S. (2014) Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* 26(12): 3026–3037.

[FLB19] Flynn M., Large J., and Bagnall T. (2019) The contract random interval spectral ensemble (c-RISE): the effect of contracting a classifier on accuracy. In *International Conference on Hybrid Artificial Intelligence Systems*, pp. 381–392. Springer.

[FLF$^+$20] Fawaz H. I., Lucas B., Forestier G., Pelletier C., Schmidt D. F., Weber J., Webb G. I., Idoumghar L., Muller P.-A., and Petitjean F. (2020) InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery* 34(6): 1936–1962.

[Fli19] Flink A. (2019) Apache Flink. `http://flink.apache.org/`.

[FLJ13] Fulcher B. D., Little M. A., and Jones N. S. (2013) Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of the Royal Society Interface* 10(83): 20130048.

[FMAAA17] Foreman-Mackey D., Agol E., Ambikasaran S., and Angus R. (2017) Fast and scalable Gaussian process modeling with applications to astronomical time series. *The Astronomical Journal* 154(6): 220.

[FPSM92] Frawley W. J., Piatetsky-Shapiro G., and Matheus C. J. (1992) Knowledge discovery in databases: An overview. *AI magazine* 13(3): 57–57.

[Fu11] Fu T.-c. (2011) A review on time series data mining. *Engineering Applications of Artificial Intelligence* 24(1): 164–181.

[Ful17] Fulcher B. D. (2017) Feature-based time-series analysis. *arXiv preprint arXiv:1709.08055* .

[GR08] Grenander U. and Rosenblatt M. (2008) *Statistical analysis of stationary time series*, volumen 320. American Mathematical Soc.

[HGD18] Hatami N., Gavet Y., and Debayle J. (2018) Classification of time-series images using deep convolutional neural networks. In *Tenth international conference on machine vision (ICMV 2017)*, volumen 10696, page 106960Y. International Society for Optics and Photonics.

[HM08] Hill M. D. and Marty M. R. (2008) Amdahl's law in the multicore era. *Computer* 41(7): 33–38.

[IJF$^+$17] Isensee F., Jaeger P. F., Full P. M., Wolf I., Engelhardt S., and Maier-Hein K. H. (2017) Automatic cardiac disease assessment on cine-mri via time-series segmentation and domain specific features. In *International workshop on statistical atlases and computational models of the heart*, pp. 120–129. Springer.

[KHL$^+$18] Kang Y., Hyndman R. J., Li F., *et al.* (2018) Efficient generation of time series with diverse and controllable characteristics. Technical report, Monash University, Department of Econometrics and Business Statistics.

[KMDH19] Karim F., Majumdar S., Darabi H., and Harford S. (2019) Multivariate lstm-fcns for time series classification. *Neural Networks* 116: 237–245.

[KPB16] Karlsson I., Papapetrou P., and Boström H. (2016) Generalized random shapelet forests. *Data mining and knowledge discovery* 30(5): 1053–1085.

[KWH12] Kirchgässner G., Wolters J., and Hassler U. (2012) *Introduction to modern time series analysis*. Springer Science & Business Media.

[LDHB12] Lines J., Davis L. M., Hills J., and Bagnall A. (2012) A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 289–297. ACM.

[LJZ15] Li Z., Jin X., and Zhao X. (2015) Drunk driving detection based on classification of multivariate time series. *Journal of safety research* 54: 61–e29.

[LKRH15] Landset S., Khoshgoftaar T. M., Richter A. N., and Hasanin T. (2015) A survey of open source tools for machine learning with big data in the hadoop ecosystem. *Journal of Big Data* 2(1): 1–36.

[LKWL07] Lin J., Keogh E., Wei L., and Lonardi S. (2007) Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15(2): 107–144.

[LL09] Lin J. and Li Y. (2009) Finding structural similarity in time series data using bag-of-patterns representation. In *International conference on scientific and statistical database management*, pp. 461–477. Springer.

[LSK$^+$19a] Lubba C. H., Sethi S. S., Knaute P., Schultz S. R., Fulcher B. D., and Jones N. S. (2019) catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery* 33(6): 1821–1852.

[LSK$^+$19b] Lubba C. H., Sethi S. S., Knaute P., Schultz S. R., Fulcher B. D., and Jones N. S. (2019) catch22: CAnonical Time-series CHaracteristics. *CoRR* abs/1901.10200.

[LSP$^+$19] Lucas B., Shifaz A., Pelletier C., O'Neill L., Zaidi N., Goethals B., Petitjean F., and Webb G. I. (2019) Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery* 33(3): 607–635.

[LTB18] Lines J., Taylor S., and Bagnall A. (2018) Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data* 12(5).

[MA14] Maharaj E. A. and Alonso A. M. (2014) Discriminant analysis of multivariate time series: Application to diagnosis based on ecg signals. *Computational Statistics & Data Analysis* 70: 67–87.

[MBY$^+$16] Meng X., Bradley J., Yavuz B., Sparks E., Venkataraman S., Liu D., Freeman J., Tsai D., Amde M., Owen S., *et al.* (2016) Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research* 17(1): 1235–1241.

[MKY11] Mueen A., Keogh E., and Young N. (2011) Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1154–1162. ACM.

[MLB20] Middlehurst M., Large J., and Bagnall A. (2020) The canonical interval forest (cif) classifier for time series classification. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 188–195. IEEE.

[MLWG16] Mei J., Liu M., Wang Y.-F., and Gao H. (2016) Learning a mahalanobis distance-based dynamic time warping measure for multivariate time series classification. *IEEE Transactions on Cybernetics* 46(6): 1363–1374.

[MWH08] Makridakis S., Wheelwright S. C., and Hyndman R. J. (2008) *Forecasting methods and applications*. John wiley & sons.

[NAM01] Nanopoulos A., Alcock R., and Manolopoulos Y. (2001) Feature-based classification of time-series data. *International Journal of Computer Research* 10(3): 49–61.

[NTAGA18] Nweke H. F., Teh Y. W., Al-Garadi M. A., and Alo U. R. (2018) Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications* 105: 233–261.

[OBPR21] Oprea S.-V., Bâra A., Puican F. C., and Radu I. C. (2021) Anomaly detection with machine learning algorithms and big data in electricity consumption. *Sustainability* 13(19): 10963.

[Pac19] Packages S. (2019) 3rd Party Spark Packages. `https://spark-packages.org/`.

[PR15] Padmavathi S. and Ramanujam E. (2015) Naïve Bayes classifier for ECG abnormalities using multivariate maximal time series motif. *Procedia Computer Science* 47: 222–228.

[RCM$^+$13] Rakthanmanon T., Campana B., Mueen A., Batista G., Westover B., Zhu Q., Zakaria J., and Keogh E. (2013) Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7(3): 1–31.

[RFL$^+$20] Ruiz A. P., Flynn M., Large J., Middlehurst M., and Bagnall A. (2020) 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.

[RGFG$^+$18] Ramírez-Gallego S., Fernández A., García S., Chen M., and Herrera F. (2018) Big Data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce. *Information Fusion* 42: 51–61.

[RK13] Rakthanmanon T. and Keogh E. (2013) Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*, pp. 668–676. SIAM.

[RT18] Rajan D. and Thiagarajan J. J. (2018) A generative modeling approach to limited channel ECG classification. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2571–2574. IEEE.

[Sch15] Schäfer P. (2015) The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery* 29(6): 1505–1530.

[SL17a] Schäfer P. and Leser U. (2017) Fast and Accurate Time Series Classification with WEASEL. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 637–646.

[SL17b] Schäfer P. and Leser U. (2017) Multivariate time series classification with weasel+ muse. *arXiv preprint arXiv:1711.11343* .

[SLJ$^+$15] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., and Rabinovich A. (2015) Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

[Spa16] Spark A. (2016) Apache spark: Lightning-fast cluster computing. *URL http://spark. apache. org* pp. 2168–7161.

[SSS00] Shumway R. H., Stoffer D. S., and Stoffer D. S. (2000) *Time series analysis and its applications*, volumen 3. Springer.

[SYHJ$^+$17] Shokoohi-Yekta M., Hu B., Jin H., Wang J., and Keogh E. (2017) Generalizing dtw to the multidimensional case requires an adaptive approach. *Data mining and knowledge discovery* 31(1): 1–31.

[TB18] Tuncel K. S. and Baydogan M. G. (2018) Autoregressive forests for multivariate time series modeling. *Pattern recognition* 73: 202–215.

[Whi12] White T. (2012) *Hadoop: The definitive guide.* " O'Reilly Media, Inc.".

[WO15] Wang Z. and Oates T. (2015) Spatially encoding temporal correlations to classify temporal data using convolutional neural networks. *arXiv preprint arXiv:1509.07481* .

[WYO17] Wang Z., Yan W., and Oates T. (2017) Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585. IEEE.

[YK09] Ye L. and Keogh E. (2009) Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 947–956.

[ZCD$^+$12] Zaharia M., Chowdhury M., Das T., Dave A., Ma J., McCauly M., Franklin M. J., Shenker S., and Stoica I. (2012) Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedins of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pp. 15–28. USENIX, San Jose, CA.

[ZGLL20] Zhang X., Gao Y., Lin J., and Lu C.-T. (2020) Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 04*, pp. 6845–6852.