Time Series Classifier Recommendation by a Meta-Learning Approach

A. Abanda[a,*], U. Mori[b], Jose A. Lozano[a,b]

[a]*Basque Center for Applied Mathematics (BCAM), Mazarredo Zumarkalea, 14, 48009 Bilbao, Spain*
[b]*Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Manuel de Lardizabal 1, 20018 Donostia-San Sebastian, Spain*

Abstract

This work addresses time series classifier recommendation for the first time in the literature by considering several recommendation forms or meta-targets: classifier accuracies, complete ranking, top-M ranking, best set and best classifier. For this, an ad-hoc set of quick estimators of the accuracies of the candidate classifiers (landmarkers) are designed, which are used as predictors for the recommendation system. The performance of our recommender is compared with the performance of a standard method for non-sequential data and a set of baseline methods, which our method outperforms in 7 of the 9 considered scenarios. Since some meta-targets can be inferred from the predictions of other more fine-grained meta-targets, the last part of the work addresses the hierarchical inference of meta-targets. The experimentation suggests that, in many cases, a single model is sufficient to output many types of meta-targets with competitive results.

Keywords: Time series classification, meta-learning, landmarkers, hierarchical inference, meta-targets

*Corresponding author
Email addresses: `aabanda@bcamath.org` (A. Abanda), `usue.mori@ehu.es` (U. Mori), `ja.lozano@ehu.eus` (Jose A. Lozano)

1. Introduction

Time series data mining is an increasing field of research due to the great amount of time series data that is being collected every day from a wide range of application domains, such as bioinformatics, engineering [1] or in the field of energy [2]. A time series is an ordered sequence of observations and, in contrast to standard data mining techniques, methods for time series mining need to deal with this ordered nature [3]. In this way, the time series community has proposed many specific methods for representing, indexing, clustering or classifying time series, among other tasks [3]. This work focuses on time series classification (TSC) [4].

As with other tasks, the growing interest in time series data mining has given rise to an increase in the available TSC algorithms. According to the "No Free Lunch" theorem [5], no single algorithm can dominate all other algorithms over all existing problems. In [4], Bagnall *et al.* proposed a categorization that groups the time series classifiers by the type of characteristics that each classifier focuses on in order to attempt to discriminate between the different classes of a time series supervised classification problem. The authors stated that the methods in each category *should* be specially suitable to a particular type of problem or dataset. Even if some attempts have been made in this direction [6], this hypothetical bijection between problem types and classifier types has not been proved yet.

In this context, and given the large amount of proposed classifiers, from a pragmatic user's point of view, choosing a suitable classifier for a given problem is a difficult task. Furthermore, many of the state-of-the-art classifiers are computationally expensive, which makes the approach of trying all of them unaffordable. Algorithm recommendation is a challenge that has never been explored in TSC. In non-temporal data mining, however, it is an area with a long record within a specific field: meta-learning [7].

Meta-learning is usually described as the task of learning to learn, and it includes any approach that exploits prior learning knowledge in order to learn new tasks. Meta-learning systems were firstly proposed for model selection or hyperparameter optimization, while nowadays it is also employed for few-shot learning [8] in (deep) neural networks [9] , among other tasks. Our work focuses on the former application, in which the main goal is to provide automatic recommendation on model selection to a non-expert user [10, 11, 12]. It is commonly used for classification [13] or clustering [14], but it has also been employed for hyperparameter optimization [15]. The way meta-learning systems give advice about models is by exploiting meta-knowledge, i.e., knowledge that relates the characteristics of datasets with the performance of the available algorithms [7]. There are three main ways in which the meta-knowledge can be extracted [7]: by employing simple, statistical and information-theoretic meta-features, model-based meta-features, and landmarkers. The first type of meta-features employ descriptive statistics or information-theoretic measures extracted from the datasets, such as class entropy, to describe the datasets, while model-based meta-features exploit parameters or characteristics found when applying a given model to a dataset. For instance, if a decision tree is used to classify the instances of a given dataset, some characteristics of the learned tree, such as the depth, can be used to describe the dataset. Landmarkers, instead, are based on the idea that the performance of fast and simple algorithms can be used to predict the performance of other computationally more expensive algorithms.

Regarding the output of the recommendation system, also known as meta-target, there are several ways in which it can be given. For example, the system can recommend the best classifier, a set of *good* algorithms or a ranking. As such, if the output of the meta-learning system is the best classifier, for instance, the user is "constrained" to use this algorithm and has no information about the rest of the algorithms. If the meta-target is a complete ranking of all the algorithms, contrarily, the user has much more information but, from a learning point of view, the problem becomes more complex.

Despite its potential, the relationship between meta-learning and time series is almost non-existent in the literature. Indeed, the few methods that exploit meta-learning for time series data are focused on forecasting method recommendation [16, 17, 18, 19, 20] or similarity measure selection for time series clustering [21]. The aim of this work is to employ a meta-learning approach for time series classifier recommendation. Since state-of-the-art meta-features are designed for non-ordered data, we propose a novel set of 24 specific landmarkers for TSC, which meet the requirements of being simple, fast and good predictors of the performance of state-of-the-art TSC algorithms. We validate the proposed landmarkers for classifier recommendation considering several types of meta-targets. Finally, we explore the hierarchical inference of meta-targets; some types of meta-targets are more fine-grained than others and, from the prediction of these meta-targets, less fine-grained meta-targets can be inferred. As such, we experimentally compare the two approaches for obtaining a given meta-target: by direct prediction of the meta-target and by inference from the predictions of more fine-grained meta-targets.

The rest of the work is organized as follows. In Section 2, our method for time series classifier recommendation is described in two parts: meta-attributes and meta-targets. The hierarchical inference of meta-targets is presented in Section 3, while the experimentation is exposed in Section 4. Lastly, the main conclusions of our work and future work are presented in Section 5.

2. Time Series Classifier Recommendation

The main objective of this work is to propose a time series classifier recommendation (TSCR) system. Algorithm recommendation, as stated in [7], aims at "saving time by reducing the number of algorithms tried out on a given problem with minimal loss in the quality of the results obtained when compared to the best possible ones".

From a practical point of view, the scheme of a TSCR system is shown in Figure 1. Given a repository of supervised time series datasets and a set of candidate classifiers, firstly, the accuracies of the candidate classifiers in those datasets are calculated by evaluating each classifier in each dataset. Then, the accuracies of the classifiers are employed to construct the meta-target. Secondly, a vector of meta-attributes is extracted from each dataset. Lastly, the meta-attributes and meta-target are used to build the TSCR system employing a meta-learner. Note that this is a supervised learning scenario and each meta-target type requires a specific meta-learner. In this way, for a new time series dataset -in which the accuracies of the candidate classifiers are unknown-, the meta-attributes are extracted and passed to the TSCR, which outputs a classifier recommendation based on the chosen meta-target type. In the following sections, the meta-attributes proposed and meta-targets considered in this work are presented.

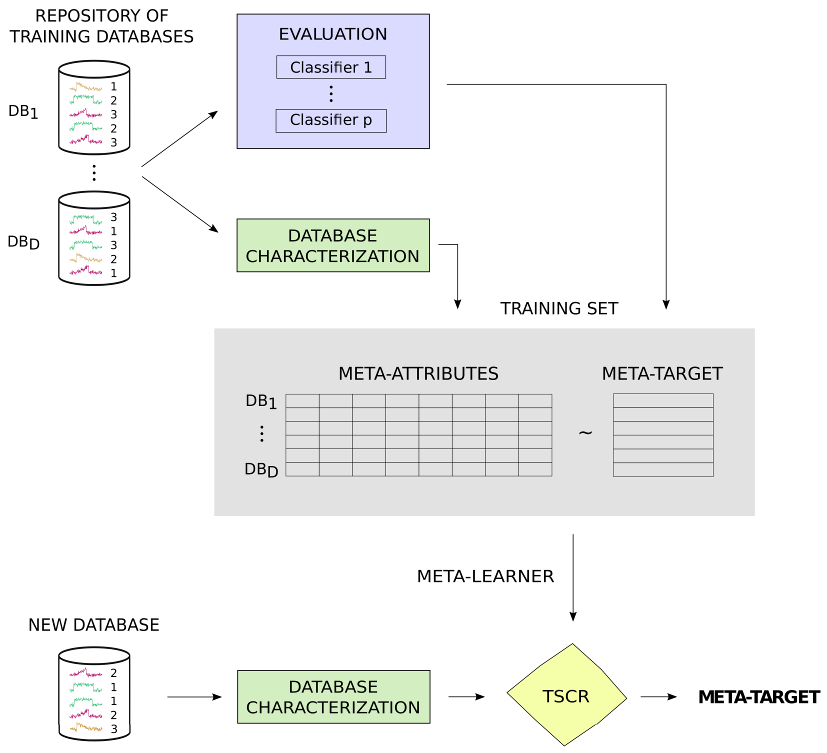


Figure 1: Meta-learning scheme for TSC algorithm recommendation.

2.1. Meta-Attributes

The characterization of a dataset is, probably, one of the challenges that has attracted most attention in meta-learning, since the performance of a meta-learning system highly depends on this characterization [22]. Indeed, meta-attributes need to fulfil two crucial requirements: on the one hand, they need to be fast to compute, and, on the other hand, they need to contain useful information for discriminating between the performances of the candidate classifiers. The definition of the meta-attributes highly depends on the specific task for which the meta-learning system is built.

In the case of time series data, as mentioned earlier, there are rather few works that have addressed the development of meta-attributes [16, 17, 18, 19, 20, 21]. In fact, most of these works focus on time series forecasting algorithm recommendation [16, 17, 18, 19, 20]. In contrast to time series classification or clustering, where a set of time series is needed, a time series forecasting problem is composed of a single time series. This is an important distinction from the point of view of meta-learning, since, in the former case, meta-attributes correspond to single time series, while in the latter, meta-attributes correspond to a time series dataset. As far as we know, the only work in which time series dataset characterization has been addressed is [21], where a set of meta-attributes are proposed with the purpose of similarity measure selection for time series clustering. However, this characterization is defined for unsupervised time series datasets and can not be applied in our case, which focuses on supervised time series datasets. This work focuses on the definition of meta-attributes for supervised time series datasets.

In this context, most of the state-of-the-art meta-attributes are designed for non-ordered instances and they become meaningless for describing time series datasets. The most generalizable meta-attributes are landmarkers, since they are quick estimators of the accuracies of the candidate classifiers and, hence, can be defined for any type of task and data. As such, we propose a set of landmarker-based meta-attributes for supervised time series datasets.

The design of landmarkers totally depends on the candidate classifiers considered in the meta-learning problem. In our case, the candidate classifiers have been chosen from those that appear in the work by Bagnall *et al.* [4], since the results obtained in their extensive experimentation are publicly available [23] -as well as a Weka-compatible Java toolbox, *tsml* [24], with the implementation of most of the classifiers included in their work-. In this way, we choose as candidate classifiers those that are included in [4] and implemented in the *tsml* or *sktime* (and the extension for deep learning *sktime-dl*) [25] toolboxes. These classifiers are: C4.5 decision tree (C45) [26], naive Bayes (NB) [27], Bayes Network (BN) [28], support vector machines with linear (SVML) [29] and quadratic kernel (SVMQ) [29], Rotation Forest (RotF) [30], Random Forest (RandF) [31] , Multilayer Perceptron (MLP) [32], 1-NN with Euclidean distance (NN), 1-NN with Dynamic Time Warping distance (DTW) [33], 1-NN with Weighted DTW distance (WDTW) [34], 1-NN with Time Warp Edit distance (TWE) [35], 1-NN with Move–Split–Merge distance (MSM) [36], 1-NN with Complexity Invariant distance (NN_CID) [37], 1-NN with Edit Distance with Weal Penalty (ERP) [38], 1-NN with Derivative DTW (DD_DTW) [39], 1-NN with Derivative Transform distance (DTD_C) [40], Time Series Forest (TSF) [41], Fast Shapelets (FS) [42], Shapelet Transform (ST) [43], Bag of Patterns (BOP) [44], and Bag of SFA Symbols (BOSS) [45]. Additionally, even if they are not included in [4], we have added two new benchmark deep learning classifiers: Resnet [46] and InceptionTime [47] in order to have a more up-to-date set of candidate classifiers. In this way, the list of candidate classifiers includes 24 heterogeneous time series classifiers. The discarded classifiers from [4] are: Longest Common Subsequence (LCSS) [48] (the *tsml* implementation does not reproduce the results), Elastic Ensemble (EE) [49] (implementation with bugs), Time Series Bag-of-Features (TSBF) [50] (not implemented in *tsml*), Learned Pattern Similarity (LPS) [51] (implementation with bugs), Dynamic Time Warping Features (DTW_F) [52] (code not available) and Collective of Transformation-Based Ensembles (COTE) [53] (it is an ensemble of several TSC methods, so from an algorithm recommendation point of view, it does not make sense to include it). Additionally, 3 classifiers have been discarded due to their high requirement of computational resources: Logistic [54] (which requires more than 50GB of RAM memory in the PigCVP dataset from the UCR repository [55]), and Learn Shapelets [56] and SAX Vector Space Model (SAXVSM) [57] (which take more than 5 and 25 days to classify a single train/test split in the Crop dataset from the UCR, correspondingly). Note that some of the considered classifiers (ST and BOSS, for instance) are also ensembles, but they are ensembles of classifiers of the same nature, while COTE is an ensemble of many types of different classifiers.

Given a candidate classifier C_i, its landmarker L_i is a quick estimator of the accuracy obtained by C_i. This quick estimator is generally obtained in two ways: by running simplified versions of the candidate classifier [58] (algorithm reduction), or by running the original algorithm in a subsample of the data, obtaining the so-called *subsampling landmarkers* [59] (dataset reduction). In both types of reductions, the greater the reduction, the higher the loss of relation between the landmarker and the original algorithm [60], so striking a balance between the level of reduction and the relation with the original algorithm is a key aspect. Our approach exploits both types of reductions: first, subsampling landmarkers are applied by evaluating the classifiers on a subsample of the dataset. Given a time series dataset with N instances, we propose a stratified subsample that depends on N. The proportion to which the dataset is reduced, the subsample ratio (r), is shown in Table 1. We are dealing with supervised datasets so, in those datasets in which the subsample ratio does not permit a minimum number of instances of each class, the r specified in Table 1 is increased by steps of 0.1 until $N * r/NC \geq 10$ or $r = 0.8$, where NC is the number of classes.

In the second step, an algorithm reduction is carried out for those classifiers that still take too long in the subsampled datasets. In order to identify the slow classifiers, we conduct the following analysis: we sort the 112 datasets from the UCR repository employed in this work by the dimension (defined by N*L, where N is the number and L the length of the series) and run the 24 classifiers in subsampled versions of the 5 largest datasets (*StarlightCurves*, *UWaveGestureLibraryAll*, *HandOutlines*, *MixedShapesRegu-*

Table 1: The subsample ratio (r) applied to a time series dataset with N instances (before the modification to ensure a minimum number of representatives of each class).

Intervals	Subsample ratio (r)
$N < 100$	0.80
$100 \leq N < 300$	0.60
$300 \leq N < 800$	0.40
$800 \leq N < 1500$	0.20
$1500 \leq N < 5000$	0.10
$5000 \leq N$	0.05

larTrain, and,*NonInvasiveFetalECGThorax1*)[1]. If a classifier takes more than 30 minutes in any of the aforementioned datasets, it is considered a slow classifier. In this way, we identify 7 slow classifiers: BOSS, DD_DTW, DTD_C, MLP, ST, ResNet and InceptionTime, to which algorithm reductions are carried out. Three types of algorithm reductions are proposed (summarized in Table 2) :

- Reducing the parameter range: in those cases in which a parameter is set by a grid search, this grid search is deleted and, by default, the value of the parameter is set to the middle of the search range (DD_DTW and DTD_C). In the R parameter of the LS, however, some values in the search range incur larger costs than others, so we set it to the computationally least expensive value.

  Reducing the iterative process/ensemble: for the classifiers that have an iteration process (MLP, ResNet and InceptionTime), we limited the number of iterations. An analogous reduction is carried out for the ensembles (BOSS).

- Setting a time limit: in the ST classifier, the training time can be directly limited by the user, so we use this characteristic to reduce the computation

[1]The computation times of the 24 classifiers in the 5 subsampled datasets are included as supplementary material (Part I).

time.

Table 2: Algorithm reductions carried out to the slow classifiers.

Classifier	**Parameter**	**Default**	**Reduced**
BOSS	MaxEnsembleSize	500	100
DD_DTW	α	grid search in {0,0.01, ..., 1}	0.5
DTD_C	α	grid search in {0, 0.01, ..., 1}	0.5
MLP	NumEpochs	500	50
ST	time limit	unlimited	5 minutes
ResNet	NumEpochs	1500	200
InceptionTime	NumEpochs	1500	200

To sum up, given a supervised time series dataset, we propose a set of 24 landmarkers to describe the dataset: 17 of them are based on dataset reductions, while the rest are based on dataset and algorithm reductions.

2.2. Meta-target

The meta-target, also known as the recommendation, corresponds to the output of the recommendation system. Five types of meta-targets are considered in this work (summarized in Table 3): classifier accuracies, complete ranking, top-M ranking, best set and best classifier. Each meta-target type gives rise to a different TSCR system, in which both the meta-target in the training set and meta-learner shown in Figure 1 must be specifically designed. We will refer to the meta-learner associated to a given meta-target type as the specific meta-learner (SML). The SMLs employed for each meta-target type are summarized in Table 4. In the following lines, a brief overview of the considered meta-target types is presented.

Classifier accuracies: this meta-target provides the accuracies obtained by each candidate classifier in a given dataset. When building the training set for learning this TSCR system, the values of the meta-targets for the training

Table 3: Summary of the considered meta-target types.

Meta-target	**Output type**
Classifier accuracies	$\begin{array}{cccccc} \mathrm{C}_1 & \mathrm{C}_2 & \dots & \mathrm{C}_8 & \dots & \mathrm{C}_P \\ 0.91 & 0.87 & \dots & 0.95 & \dots & 0.72 \end{array}$
Complete ranking	$\begin{array}{ccccc} \mathrm{C}_8 & \mathrm{C}_1 & \mathrm{C}_2 & \dots & \mathrm{C}_5 \\ 1 & 2 & 3 & \dots & \mathrm{P} \end{array}$
Top-M ranking	$\begin{array}{ccc} \mathrm{C}_8 & \mathrm{C}_1 & \mathrm{C}_2 \\ 1 & 2 & 3 \end{array}$
Best subset	$\{\mathrm{C}_1 , \mathrm{C}_8\}$
Best	C_8

instances are directly defined by the accuracies of the classifiers. This meta-target type gives rise to a multi-output regression problem, and the SML chosen for this problem is the linear multi-output regression. It is the most fine-grained meta-target type, and the user is free to choose among the candidate classifiers with the provided information.

Complete ranking: this meta-target provides an accuracy-based ranking of the candidate classifiers in a given dataset. The values of the meta-targets in the training set are obtained by converting the raw accuracies of the candidate classifiers into a ranking. In case of ties, the best ranking position of the tied classifiers is assigned to all the tied classifiers. This meta-target type requires a ranking learning strategy, and the SML employed in this case is the K-NN for rankings [13] (based on the Euclidean distance), commonly used in meta-learning. From the user point of view, ranking recommendation is less fine-grained than the classifier accuracies but still leaves the user with a great choice since information regarding all the classifiers is provided.

Top-M ranking: in this case, the meta-target corresponds to the partial ranking of the M best-performing classifiers in a given dataset. When building the training set, the values of the meta-targets are obtained by converting the classifier accuracies into a ranking and selecting the classifiers at the first M positions (M is predefined by the user). The ties are handled in the same manner as for complete rankings. Analogously to the case of complete rankings, the SML employed for this meta-target type is the K-NN for rankings based on the Euclidean distance. This meta-target type is less fine-grained than the complete ranking since only the ranking of the M best-performing classifiers is provided.

Best subset: this meta-target provides the set of best-performing classifiers in a given dataset. Establishing which algorithms perform well on a given dataset is not straightforward, and it is usually defined in relative terms. A commonly employed approach to obtain the values of the meta-targets in the training set consists of defining a margin [61], such that the classifiers with accuracies within this margin are considered *applicable* classifiers in the given dataset. We propose a margin based on the proposal in [61], but enhanced to consider the range of the accuracies obtained by the candidate classifiers in the given dataset:

$$\Big[a_{\max}, a_{\max} - W(a_{\max} - a_{\min})\Big) \tag{1}$$

where W is a user-defined parameter, while $a_{\max}$ and $a_{\min}$ are the maximum and minimum accuracies obtained by the candidate classifiers in the given dataset. Note that, the greater W, the wider the interval and, hence, the larger the set of applicable classifiers. The best set meta-target gives rise to a multi-label classification problem, and, analogous to the ranking meta-targets, a K-NN version -based on the Euclidean distance- for multi-label classification [62] is employed as SML. This meta-target type is less fine-grained than the top-M meta-target types since it does not provide a ranking, but an unordered set of the applicable classifiers.

Best: this meta-target provides the best classifier among the candidate classifiers. The values of the meta-targets in the training set are obtained in a

Table 4: The specific meta-learner employed for each meta-target.

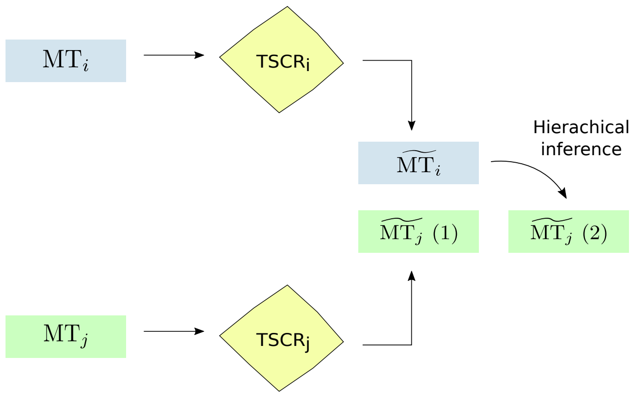
Meta-target	**SML**
Classifier accuracies	Linear multi-output regression
Complete ranking	K-NN for rankings
Top-M rankings	K-NN for rankings
Best set	K-NN for multi-label
Best	K-NN for multi-class

straightforward manner. The best classifier meta-target gives rise to a multi-class standard classification problem, and, for the sake of consistency, a K-NN for multi-class problems is used. This meta-target type is the least fine-grained meta-target, since it only provides the best performing classifier and gives no information about the rest of the classifiers.

3. Hierarchical inference of meta-targets

In this section, we explore the hierarchical relationship between the different meta-target types. Some meta-target types are more fine-grained than others and this fact gives rise to a hierarchy. By using this hierarchy, instead of building a specific TSCR system for each meta-target type, we investigate the approach of inferring meta-targets from the predictions made by more fine-grained TSCR systems.

Figure 2 shows the scheme of the hierarchical inference of the meta-target MT_j from the meta-target MT_i, where MT_i is a more fine-grained meta-target than MT_j. In Section 2.2, we showed how $\widetilde{\mathrm{MT}_i}$ and $\widetilde{\mathrm{MT}_j}(1)$ are predicted by the associated TSCR_i and TSCR_j systems, correspondingly. In the hierarchical inference, $\widetilde{\mathrm{MT}_j}(2)$ is obtained by transforming the prediction $\widetilde{\mathrm{MT}_i}$ made by TSCR_i.



MT_i
TSCRi
Hierachical inference
$\widetilde{\mathrm{MT}_i}$
$\widetilde{\mathrm{MT}_j}$ (1)
$\widetilde{\mathrm{MT}_j}$ (2)
MT_j
TSCRj

Figure 2: Scheme of the two approaches for obtaining $\widetilde{\mathrm{MT}_j}$: (1) by employing the TSCR_j associated to MT_j and (2) by hierarchical inference from the prediction of the MT_i meta-target.

More fine-grained information is learnt by TSCR_i than by TSCR_j, so $\widetilde{\mathrm{MT}_j}(2)$ could be expected to be more accurate than $\widetilde{\mathrm{MT}_j}(1)$. However, from a learning point of view, the prediction of MT_i is a harder task, and, therefore, more errors could be committed, so the comparison of these two approaches can shed some light on which learning strategy should be adopted when building a recommendation system for TSC algorithms.

As far as we know, despite its interest, this point of view is still almost unexplored in the meta-learning community. In [63], the author inferred the expected ranking and the expected best classifier from the predicted classifier accuracies, but reported that the inferred versions seem to perform slightly worse than the specific strategies. Bensusan *et al.* carried out a similar experiment in [64], where a ranking is inferred from the predicted classifier accuracies. In the same manner, the authors reported that even if the regression models obtained low errors, the results of the inferred rankings were not very promising. Moreover, not all the possible hierarchical dependencies are studied, but only a few pairs. In this work, we explore all the feasible hierarchical inferences for the considered meta-target types.

The feasible inferences for each meta-target type are shown in Table 5. From classifier accuracies, all the meta-targets can be inferred: complete ranking, top-M ranking, best set and best classifier. Complete ranking, top-M ranking and best classifier transformations are straightforward, while, in the best set transformation, the same procedure described in Section 2.2 is followed. A complete ranking can be directly transformed into a top-M ranking and best classifier. The best set, however, can not be inferred since the criterion employed for the transformation is not applicable in rankings. In the same manner, a top-M ranking can not be transformed into a best set, and from this meta-target, only the best classifier can be inferred. From the best set, the only meta-target that could be inferred (best classifier) is not feasible, due to the unordered nature of the best set. The best classifier meta-target is the least fine-grained meta-target, and hence, no meta-target can be inferred from it.

Table 5: Feasible hierarchical inferences.

Meta-target	**Feasible inferences**
Classifier accuracies	Complete ranking, Top-M ranking Best set, Best
Complete ranking	Top-M ranking, Best
Top-M ranking	Best

4. Experimentation

The experimentation is divided into four parts: in the first part, the experimental set up is introduced, as well as the evaluation procedure employed for the proposed methods. In the second part, the results obtained by the landmarkers are analysed and compared to those obtained by the original classifiers. The experimental evaluation of the proposed TSCR systems is presented in the third part. Lastly, the hierarchical inference of the meta-targets is experimentally studied[2].

4.1. Experimental set-up

4.1.1. Datasets

The experimentation has been carried out employing datasets from the UCR repository [55], a frequently used benchmark archive for evaluating time series classification methods. This repository includes datasets with a great variety of characteristics, such as different application domains, varying time series lengths, number of classes, etcetera. In this work, we have decided to use datasets of univariate equal-length time series; most of time series classifiers can only be applied in datasets of this type and the need for meta-learning becomes more evident in a context with a large number of candidate classifiers. As such, the 112 datasets from the UCR with these characteristics have been used.

4.1.2. Classifiers

All the experiments have been carried out in python with the help of the *sklearn* library. For the classifier accuracies meta-target, the linear multi-output regression included in the *sklearn* library is used. We set the number of neighbours in the K-NN to 5 after preliminary experiments. For the top-M rankings, we explore the results for M=3, M=5 and M=10. In the best subset prediction, we need to establish the width of the margin for the best classifier, W, beforehand. Figure 3 shows the number of labels by instance for the different values of W we have considered. It can be seen that the number of labels increases as W grows. We employed the K-NN version of the *sklearn* for multi-label classification with K=5 as it obtains competitive results compared to other classifiers and ensures consistency with the previous approaches. Lastly, regarding the

[2]The TSCR systems, as well as the code for reproducing all the experiments presented in this work are available at `https://gitlab.bcamath.org/aabanda/tscr`.

best classifier meta-target, Figure 4 shows the distribution of labels among the 112 datasets. It can be seen that InceptionTime is best performing classifier in 37 datasets, followed by ResNet, ST and BOSS (which are the best performing classifiers in 15, 15 and 11 datasets, respectively). In fact, only 16 of the 24 algorithms win at least once, so it is a 16-class classification problem, for which we employ the K-NN classifier for multi-class of the *sklearn* library with K=1, set by preliminary experiments.

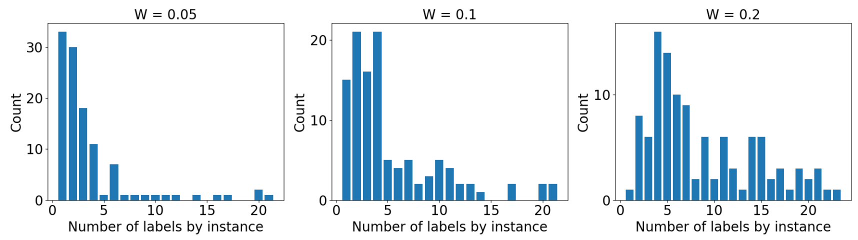


Figure 3: Distribution of labels for the best set meta-target.

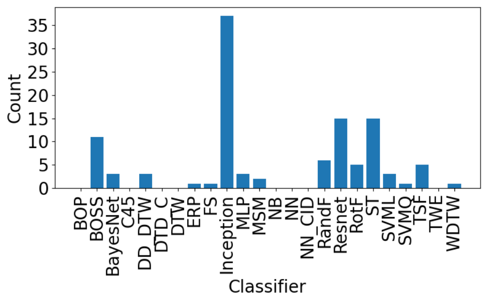


Figure 4: Distribution of labels for the best classifier meta-target.

4.1.3. Evaluation

Each meta-target type requires a suitable evaluation measure for the corresponding output type. In the following, the metrics employed for each meta-target type are presented (summarized in Table 6). For the classifier accuracies meta-target, we employ the Mean Absolute Error (MAE), where A_c and $\widetilde{A_c}$ are the true and predicted accuracies of the candidate classifier c, and p is the number of classifiers. In order to evaluate a complete ranking meta-target, we employ a similarity measure based on the normalized Kendall's τ distance for rankings -which measures the pairwise disagreements between two rankings-. Specifically, given a Kendall's τ distance between two rankings r_1 and r_2 of length n, $\tau(r_1, r_2)$, we propose the similarity measure $\tilde{\tau}_c$, which takes values from 0 (for reverse rankings) to 1 (for identical rankings). There are several ways for generalizing the Kendall's τ distance to top-M rankings [65]; a basic generalization consists of, given two complete rankings r_1 and r_2, considering all the elements at positions higher than M are tied in both rankings, and computing the Kendall's τ distance between those partial rankings. In this way, given two top-M rankings r'_1 and r'_2, if $\tau(r'_1, r'_2)$ is the Kendall's τ distance between these partial rankings, we propose the similarity measure between top-M rankings $\tilde{\tau}_p$. This similarity measure takes values analogously to $\tilde{\tau}_c$. For the best set meta-target, the evaluation measure used is the mean label-based accuracy, L_a, where Y_{true} and Y_{pred} are the set of true and predicted labels, correspondingly. This metric takes values from 0 (when the intersection is zero) to 1 (when the predicted set is equal to the true set). Finally, in the case of best classifier meta-target, a weighted version of the F1 score (F1_w) is employed as the evaluation measure: the F1 score is calculated for each label and then a weighted average that takes into account the label frequency is computed. The F1_w score takes values from 0 (worst prediction) to 1 (perfect prediction).

4.2. Analysis of the landmarkers

The estimations of the landmarkers have been obtained by executing the classifiers in a stratified random train/test partition with the same proportions as in [23]. Note that the accuracies obtained by the landmarkers may vary depending on the run, due to the dataset reduction of the landmarkers, the train/test split, and the randomness of classifier itself. Hence, with the aim of avoiding noisy results, we have run 10 executions of the landmarkers, and we have used the results obtained in each execution as input for the recommendation system. In this way, we have trained and evaluated 10 recommendation

Table 6: The meta-learner and evaluation metric employed for each meta-target type.

Meta-target	**Metric**
Classifier accuracies	$MAE = \frac{1}{p}\sum_{c=1}^{p} \lvert A_c - \widetilde{A_c}\rvert$
Complete ranking	$\tilde{\tau}_c(r_1, r_2) = 1 - \frac{\tau(r_1,r_2)}{n(n-1)/2}$
Top-M rankings	$\tilde{\tau}_p(r'_1, r'_2) = 1 - \frac{\tau(r'_1,r'_2)}{M^2}$
Best set	$L_a = \frac{\lvert Y_{\text{true}} \cap Y_{\text{pred}}\rvert}{\lvert Y_{\text{true}} \cup Y_{\text{pred}}\rvert}$
Best	$\text{F1}_w = \frac{1}{L}\sum_{l=1}^{L} w_l F1_l$

systems, one for each execution of the landmarkers. As a preliminary study, in order to assess the quality of the proposed landmarkers, their accuracies and computational times are analysed, with the objective of comparing them to those obtained by the original classifiers.

Regarding the accuracies of the original classifiers, we employ the publicly available results reported in [23]. In this experimentation, with the aim of estimating the performances of the classifiers as accurately as possible, the authors execute each classifier in 100 train/test stratified re-samples of each dataset and report the mean accuracy values. Note that the accuracies of the original classifiers are publicly available, but the computational time of the full process (learning, parameter optimization and test phase) is not. In order to give some insights into the computational cost of executing all the classifiers, we run the 24 original classifiers in the *StarlightCurves* and *Crop* datasets. The former is one of largest datasets in the UCR repository (9236 series of length 1024, with 3 classes), while the latter is a large dataset with many classes (24000 series of length 46, with 24 classes). The total time spent for executing the 24 classifiers is more than 12 days for the *StarlightCurves* dataset and almost 4 days for the *Crop* dataset in a high performance computing cluster.

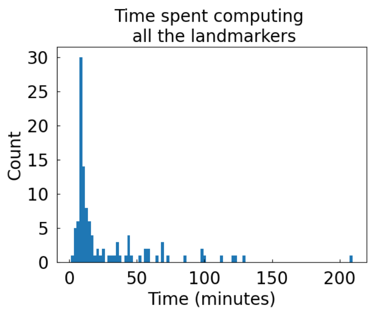
Regarding the computational cost of the landmarkers, Figure (5a) shows the time spent (in minutes) in applying all the landmarkers (in one execution) to the 112 datasets. It can be seen that, in 98 of the 112 datasets, the computation of all the landmarkers takes less than an hour. Those datasets in which the computation takes more than an hour are large datasets that contain many classes, for which, in order to ensure a minimum number of representatives of each class, the subsample ratio is not as small as in other datasets with similar dimensions. In this way, the time reduction is especially significant for those dataset with few classes. This is the case of *StarlightCurves*, for example, for which the computation of all the landmarkers takes 44 minutes, compared to the 12 days spent by the original classifiers. In the case of datasets with many classes, the time reduction is not so great, even if it is still significant. In the *Crop* dataset, for example, the computation of all the landmarkers takes almost an hour (58 minutes), compared to the 4 days spent by the original classifiers. Concerning the individual computation times of the landmarkers[3], in 80 of the 112 datasets, the slowest classifier is the ST classifier (which has a time limitation), while BOSS, ResNet and RotF are slowest in 28, 3, and 1 dataset, respectively.

In order to explore the relation between the accuracies obtained by a landmarker and those obtained by the corresponding original classifiers, we employed the *Pearson correlation coefficient*. For this, given a classifier, we have computed the mean correlation between its accuracy and the accuracy obtained by the corresponding landmarker in the 10 executions. The mean correlations between the landmarkers and the original classifiers[4] are shown in Figure 5b.

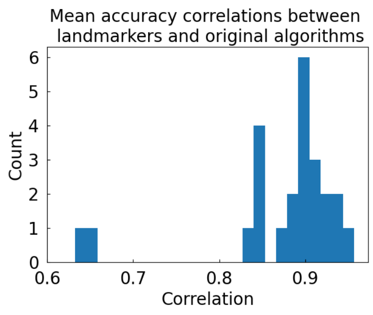
[3]The cost of computing the 24 landmarkers in each dataset, including the slowest landmarker and its corresponding time is included as supplementary material (Part II).

[4]The mean correlation of each landmarker and the corresponding classifier is included as supplementary material (Part III).

It can be seen that most of the landmarkers have a high correlation with the corresponding original classifiers; the mean correlation is 0.87. The landmarker with the lowest correlation is the InceptionTime, which is one of the slowest classifiers, so finding a suitable trade-off between computation time reduction and relation with the original classifier is a hard task. Indeed, small changes in the reduction grade result in a considerable increase in the time needed to compute the landmarker. Taking into account that a correlation of 0.63, even if not high, still means that there is a positive correlation between the landmarker and the original classifier, we chose to prioritize the time over the correlation.



(a)



(b)

Figure 5: (a) Histogram of the time spent computing the landmarkers in the 112 dataset. (b) Histogram of the mean correlations of the 24 landmarkers and the corresponding original algorithms.

Another way of testing the correlation of the landmarkers and the original classifiers is dataset-wise. Given a dataset, the results obtained by the landmarkers (in one execution) and the original classifiers in this dataset can be compared visually to see whether or not they are related. Due to the lack of space, only two examples[5] are shown in Figure 6. It can be seen that there is a clear relation between the accuracies obtained by the original algorithms and the corresponding landmarkers in the both datasets shown in the figure, since they share a similar pattern. The landmarkers of both datasets are good

[5]The figures of the 112 datasets are included as supplementary material (Part IV).

potential predictors of classifier recommendations.

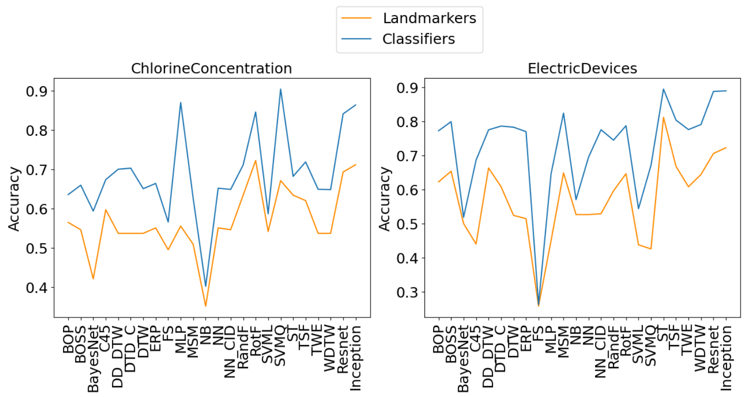


Figure 6: Accuracies obtained by the original 24 classifiers and the landmarkers in *ChlorineConcentration* (left) and *ElectricDevices* (right) datasets.

4.3. TSCR

In this section, the proposed TSCR systems are experimentally evaluated. Apart from the SMLs presented in Section 2.2, we present three additional methods for each meta-target type: on the one hand, since classifier recommendation has never been addressed before in TSC, in order to determine if the results obtained by our methods are accurate or not, a baseline (BS) for each meta-target type is presented. Moreover, in order to highlight the contribution of the proposed landmarkers, the recommendation performance of the landmarkers is compared with the recommendation performance of standard meta-features for non-sequential data (MF). On the other hand, since not all landmarkers necessarily have a great prediction power, a method that includes a forward landmarker selection (SML_FLS) is considered. In the following, a brief introduction to those methods is presented.

BS: since there are no state-of-the-art proposals, we consider a set of agnostic methods that, for each meta-target type, outputs a constant prediction without going through any learning process. This prediction is made based on the mean values of the accuracies of the classifiers in the training set: given a training set composed of n instances (datasets), let $\mathbf{A} = \{a_{i,j}\}$ be the matrix of accuracies of the candidate classifiers in the n datasets, where $i = 1, \ldots, n$ refers to the index of the dataset, and $j = 1, \ldots, 24$ refers to the index of the candidate classifier. Hence, the constant output provided by the BS method for the classifier accuracies meta-target is the vector $\mathbf{a} = (\overline{a_{*1}}, \ldots, \overline{a_{*24}})$, where $\overline{a_{*j}} = \frac{1}{n}\sum_{i=1}^{n} a_{ij}$ refers to the mean of the column j. For the rest of the meta-targets, the output is obtained by applying procedure to build each meta-target (described in Section 2.2) to the vector $\mathbf{a}$.

MF: In this method, instead of the landmarkers, standard meta-features for non-sequential data are used as meta-attributes for the recommendation system. Specifically, 73 standard meta-features (general, statistical and info-theoretical) are extracted from each dataset employing the *pymfe* package [66] for Python. These meta-features are used as input for the recommendation system, employing the corresponding meta-learner in each case.

SML_FLS: the followed forward landmarker selection (FLS) procedure is described in Algorithm 1. The FLS starts with a random landmarker and, at each iteration, adds the landmarker that improves the performance of the meta-learner the most to the set of selected landmarkers, until there is no improvement. The operator δ depends on the meta-target type and is the result of evaluating the predictions of the SML with the corresponding evaluation metric. For complete ranking, top-M ranking, best set and best classifier meta-targets the aim is to maximize δ (as in the pseudo-code), while for classifier accuracies meta-target is evaluated in terms of error, so the aim is to minimize δ.

From a general point of view, the evaluation procedure of the methods is the following: for the BS and SML, a 10 times repeated 5-fold nested cross validation [67] is carried out. For the forward selection, instead, a two-level nested cross validation is performed: in the first level, a 10 times repeated 5-fold nested cross validation is performed for evaluating the set of landmarkers, which have been selected in an internal 3-fold nested cross validation of training sets of the first level. The method starts with all the landmarkers once and selects the set of

Algorithm 1 Forward landmarker selection (FLS)

```
Input:
    LM = set of all the landmarkers
    $\delta$ = operator that computes the performance of a set of landmarkers
Output:
    S = set of selected landmarkers
Algorithm:
    Initialize S = one.random(LM)
    candidate = $\underset{\mathrm{L}_i \in \mathrm{LM} \setminus \mathrm{S}}{\operatorname{argmax}} \delta (\mathrm{S} \cup \mathrm{L}_i)$
    while $\delta$ (S ∪ candidate) > $\delta$ (S) do
        S = S ∪ candidate
        if |S| = |LM| then
            break
        else
            candidate = $\underset{\mathrm{L}_i \in \mathrm{LM} \setminus \mathrm{S}}{\operatorname{argmax}} \delta (\mathrm{S} \cup \mathrm{L}_i)$
```

landmarkers that obtains the best performance in the 3-folds. In the case of the methods depending on the landmarkers (SML and SML_FLS), this procedure is repeated for each execution of the landmarkers.

The results obtained by the BS, MF, SML and SML_FLS approaches for the considered meta-target types are shown in Table 7. The BS and MF methods do not depend on the landmarkers, so we show the results for a single execution. In the columns corresponding to SML and SML_FLS, the mean and standard deviation (between parenthesis) of the results of 10 recommendation systems (one for execution of the landmarkers) are shown. Recall that each meta-target type is validated employing a different evaluation measure and, hence, the results of different rows can not be compared with each other.

First of all, it can be seen that the standard deviations are very low, which means that the results are rather stable with respect to the execution of the landmarkers. In the case of the classifier accuracies meta-target, it can be

Table 7: Results of the baselines (BS), specific meta-learners with meta-features as input (MF), specific meta-learners with landmarkers (SML) as input, and specific meta-learners with forward landmarker selection (SML_FLS) for the considered meta-target types.

Meta-target		**Metric**	**BS**	**MF**	**SML**	**SML_FLS**
Classifier accuracies		MAE	0.15	0.14	**0.07 (0.01)**	**0.07 (0.01)**
Complete ranking		$\tilde{\tau}_c$	0.73	0.73	**0.75 (0.01)**	0.74 (0.01)
Top-M ranking	M=3	$\tilde{\tau}_p$	0.37	0.31	0.48 (0.02)	**0.49 (0.01)**
	M=5		0.50	0.50	**0.61 (0.01)**	0.60 (0.01)
	M=10		0.63	0.64	0.69 (0.01)	**0.70 (0.01)**
Best set	W=0.05	L_a	**0.35**	0.28	0.30 (0.01)	0.29 (0.01)
	W=0.1		0.25	0.34	0.38 (0.01)	**0.40 (0.01)**
	W=0.2		0.41	0.43	0.48 (0.02)	**0.49 (0.01)**
Best		$F1_w$	0.17	**0.27**	0.20 (0.02)	0.19 (0.01)

seen that, both the BS and the MF obtain worse results than the SML, while the forward selection does not improve the performance obtained by the SML. Additionally, the best performance obtained, 0.07, is the MAE of the prediction of 24 classifier accuracies, which, taking into account the range of the accuracies, is fairly accurate.

In the complete ranking prediction, the SML is the best performing approach -closely followed by the BS and MF-, while the landmarker selection does not improve the results. The good performance of the BS deserves attention; the BS always predicts the mean ranking in the training set, so it can be deduced that most of the rankings in the testing set are close to that mean ranking. In order to explore this issue, we conduct the same experiment but, instead of computing the overall performance of all the predicted rankings, the performance is evaluated separately for instances that have different similarities ($\tilde{\tau}_c$) with respect to the mean ranking in the training set. For each fold in the cross validation, the percentiles P_{25}, P_{50} and P_{75} are extracted from the similarities between the rankings in the training set and the mean ranking in the training set, and the instances in the testing set are divided into 4 sets: S_1 is the set of test instances with similarity within the range [P_0, P_{25}), and S_2, S_3 and S_4 within [P_{25} P_{50}), [P_{50} P_{75}) and [P_{75} P_{100}], respectively. Figure 7 shows the mean performance and standard deviation of the BS and SML for the different sets. It can be seen that, as expected, the SML outperforms the BS in those instances with a true ranking that is very dissimilar to the mean ranking in the training set. Even if the average performances are similar (0.73 and 0.74), the SML distributes the error more uniformly among the instances.

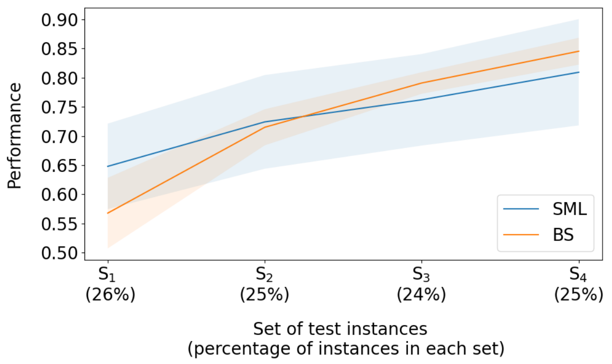


Figure 7: Performance depending on the similarity of an instance respect to the mean ranking in the training set.

In the case of the top-M rankings, two main conclusions can be drawn:the SML_FLS is the best approach in two of the three considered cases, and the performances of all the meta-learners increases with M. These results are quite reasonable since it is expected that predicting a top-10 ranking is an easier task than predicting a top-3 ranking.

A similar pattern can be seen in the best set prediction: the performance of the MF, SML and SML_FLS increases with W and, hence, with the number of labels per instance, while the performance of the BS does not seem to follow this trend. The forward selection improves the results of the SML for two of the three considered W's. The main difference with the results obtained for the top-M rankings is that, in the best set prediction, for W=0.05 the BS outperforms the rest of the approaches. A possible explanation for this fact is that, as Figure 3 shows, for W=0.05, most of the instances have very few labels. As shown in

Figure 4, InceptionTime, BOSS and ST are the best performing classifiers in most of the datasets, so many of the instances contain these labels. Since BS predicts the most frequent classifiers, it obtains good results due to the label imbalance.

Lastly, regarding the best classifier meta-target, we handle the label imbalance by the metric specified in Section 4.1.3. It can be seen that the MF is the best performing approach, followed by the SML and SML_FLS methods.

Summarizing, the experimentation carried out shows that the SMLs are, in 7 out of 9 of the considered scenarios, the best performing TSCR approaches. In some cases, the corresponding BS obtains results that are similar -in a single case better- to the SML, which we think is because there are some classifiers that, in general, obtain better results than others, a fact that benefits the BS. The approach that employs the standard meta-features as input outperforms the landmarker-based approaches in a single case (best meta-target), and the SML_FLS improves the results of the SML in several cases, which indicates that all the landmarkers are informative in some cases, while in orders they are not.

4.4. Hierarchical inference of meta-targets

In this section, we experimentally compare the hierarchical inference of meta-targets with the corresponding TSCR system in order to explore whether or not a specific TSCR is needed. For each meta-target type, we chose the best performing approach between the SML and the SML_FS (Table 7) for the hierarchical inference. The experimental set up is the same as that specified in Section 4.3.

Table 8 displays the means and standard deviations of the results obtained for the 10 executions of the landmarkers in the hierarchical inference. Following the scheme in Figure 2, the columns indicate the MT_i meta-target, while the rows refer to the inferred MT_j meta-targets; the first column, for example, displays the performances obtained for the different meta-target types inferred from the predicted classifier accuracies. The results obtained by the SML of each meta-target type are shown in the diagonal, while the $\times$ symbol reflects that the inference is not feasible for this couple of meta-target types. Analogously to Table 7, each meta-target type and, hence, each row, is evaluated employing a different metric, so the results of different rows can not be compared with each other.

Table 8: Results obtained for the hierarchical inference approach of the considered meta-target types. Columns indicate the departing meta-target, while rows indicate the inferred meta-target.

		Classifier accuracies	Complete ranking	Top-M ranking M = 3, 5, 10	Best set W = 0.05, 0.1, 0.2	Best
Classifier accuracies		**0.07 (0.01)**				
Complete ranking		**0.77 (0.01)**	0.75 (0.01)			
Top-M ranking	M=3	**0.61 (0.06)**	0.49 (0.02)	0.49 (0.02)		
	M=5	**0.61 (0.06)**	**0.61 (0.01)**	**0.61 (0.01)**		
	M=10	0.63 (0.05)	**0.70 (0.01)**	**0.70 (0.01)**		
Best set	W=0.05	**0.32 (0.01)**			0.30 (0.01)	
	W=0.1	0.36 (0.01)	×	×	**0.40 (0.01)**	
	W=0.2	0.46 (0.01)			**0.49 (0.01)**	
Best		**0.27 (0.01)**	0.20 (0.01)	0.20 (0.01)	×	0.20 (0.02)

It can be seen that, in general, the inference obtains very competitive performances, with low variability in the results. In fact, in most of the cases, the results of our experimentation show that there is no need to employ a specific TSCR system for each meta-target type, since almost equally or even better results can be inferred from more fine-grained meta-targets. That means that, in this problem, a single model is sufficient to obtain competitive results for many meta-target types. For the complete ranking meta-target, for instance, our experimentation suggests that a ranking learning method is not necessary since even a better performance is obtained by inferring the rankings from the predictions of the linear multi-output regression. An interesting pattern that is observed for the top-M rankings meta-target is that, the smaller the parameter, the better the results of the hierarchical inference (from the classifier accuracies) are compared to the SML. A possible explanation for this fact is that, smaller parameters give rise to more fine-grained meta-targets (shorter partial rankings), and these meta-targets seem to benefit from more fine-grained TSCR systems.

To sum up, in contrast to what Kalousis *et al.* [63] and Bensusan *et al.* [64] reported, the results of our experimentation suggest that, given a meta-target type, the hierarchical inference obtains results that are competitive with the specific TSCR. The main conclusion of this finding is that, at least in our scenario, a single linear multi-output regression is enough to infer almost all the meta-target types with results that are competitive with the corresponding SML.

5. Conclusion and future work

In this work, time series classifier recommendation has been addressed by a meta-learning approach for the first time in the literature. The proposed method consists of three main parts: landmarker-based time series supervised dataset characterization, classifier recommendation and hierarchical inference of the meta-targets.

In the first part, the temporal supervised dataset characterization is tackled by the proposal of a set of 24 TSC landmarkers, which are obtained by dataset subsampling and algorithm reductions. The experimental analysis of the landmarkers show that they are fast to compute (in the *StarlightCurves* dataset, for instance, the time is reduced from more than 12 days to 44 minutes), while the accuracies obtained by the landmarkers are highly correlated to the results of the original classifiers (mean correlation of 0.87).

Five standard meta-target types are considered: classifier accuracies, complete ranking, top-M ranking, best set and best classifier. For each meta-target type, four TSCR approaches are considered: two specific meta-learners (the SML and the SML_FLS), as well as a baseline BS and an approach with standard meta-features MF, for comparative purposes. The experimentation validates the landmarkers we proposed since, in 7 of the considered 9 recommendations, the SML outperforms the BS and MF, while the forward selection improves or equals the results in 5 of 9 scenarios. Moreover, in those cases in which the BS obtains competitive results, we prove that the proposed methods are more stable than the BS -in the sense that they distribute the error more uniformly among the datasets-.

In the last part of the work, the hierarchical inference of meta-targets is addressed. This issue is an almost unexplored point of view in the meta-learning literature, and the few works that have addressed it [63, 64] did not report competitive results. In our work, by contrast, it is proved to be a promising approach. Most of the meta-target types can be inferred from a single linear multi-output regression, obtaining even better results than those obtained with the corresponding TSCR system.

Regarding future work, the proposed TSCR systems could be extended to other time series classification scenarios, such as multi-variate or unequal-length time series classification. In addition, it would be interesting to find out which characteristic of the time series datasets makes a classifier obtain better results than others. This is a pending aspect in the TSC community and could be addressed by the definition of meta-attributes that describe supervised time series datasets, based on the characteristics that they contain.

Acknowledgements

This research is supported by the Basque Government through the BERC 2022-2025 program and by Spanish Ministry of Economy and Competitiveness MINECO through BCAM Severo Ochoa excellence accreditation SEV-2017-0718, as well as through project TIN2017-82626-R funded by (AEI/FEDER, UE) and acronym GECECPAST. In addition, by the Research Group IT1244-19 programs (Basque Government), PID2019-104966GB-I00 (Spanish Ministry of Economy, Industry and Competitiveness) and Elkartek projects 3KIA (KK2020/00049), SIGZE (KK-2021/00095) and ALUSMART (KK-2021/00065). A. Abanda is also supported by the Grant BES-2016-076890. The authors would like to thank the people who contributed to the UCR time series repository.

References

[1] E. Keogh, S. Kasetty, On the need for time series data mining benchmarks, Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002) 102.

[2] Y. Pang, X. Zhou, J. Zhang, Q. Sun, J. Zheng, Hierarchical electricity time series prediction with cluster analysis and sparse penalty, Pattern Recognition 126 (2022) 108555.

[3] P. Esling, C. Agon, Time-series data mining, ACM Computing Surveys 45 (2012) 1–34.

[4] 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 (2017) 606–660.

[5] D. H. Wolpert, W. G. Macready, F. Analysis, No free lunch theorems for search, Technical Report SFI-TR-95-02-010, The Santa Fe Institute (1996).

[6] A. Bagnall, A. Bostrom, J. Large, J. Lines, Simulated data experiments for time series classification part 1: accuracy comparison with default Settings, arXiv preprint arXiv:1703.09480 (2017).

[7] P. Brazdil, Metalearning: Applications to Data Mining, 2009.

[8] H. Xu, J. Wang, H. Li, D. Ouyang, J. Shao, Unsupervised meta-learning for few-shot learning, Pattern Recognition 116 (2021) 107951.

[9] M. Huisman, J. N. V. Rijn, A. Plaat, A survey of deep meta-learning, volume 54, Springer Netherlands, 2021. URL: https://doi.org/10.1007/s10462-021-10004-4. doi:10.1007/s10462-021-10004-4.

[10] S. Y. Sohn, Meta Analysis of Classification Algorithms for Pattern Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (1999) 1137–1144.

[11] M. Reif, F. Shafait, M. Goldstein, T. Breuel, A. Dengel, Automatic classifier selection for non-experts, Pattern Analysis and Applications (2014) 83–96.

[12] I. Khan, X. Zhang, M. Rehman, R. Ali, A Literature Survey and Empirical Study of Meta-Learning for Classifier Selection, IEEE Access 8 (2020).

[13] P. B. Brazdil, C. Soares, J. P. Da Costa, Ranking learning algorithms: using IBL and meta-Learning on accuracy and time results, Machine Learning (2003) 251–277.

[14] R. G. F. Soares, D. S. A. D. Araujo, I. G. Costa, T. B. Ludermir, A. Schliep, Ranking and selecting clustering algorithms using a meta-learning approach, IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence) (2008) 3729–3735.

[15] M. Feurer, J. T. Springenberg, F. Hutter, Initializing bayesian hyperparameter optimization via meta-learning, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (2014) 1128–1135.

[16] M. Matijaš, J. A. K. Suykens, S. Krajcar, Load forecasting using a multivariate meta-learning system, Expert Systems with Applications 40 (2013) 4427–4437.

[17] C. Lemke, G. Bogdan, Meta-learning for time series forecasting and forecast combination, Neurocomputing 73 (2016) 2006–2016.

[18] X. Wang, K. Smith-miles, R. Hyndman, Rule induction for forecasting method selection : meta-learning the characteristics of univariate time series, Neurocomputing 72 (2009) 2581–2594.

[19] R. Prudêncio, T. Ludermir, Using machine learning techniques to combine forecasting methods (2004) 1122–1127.

[20] T. B. Ludermir, R. B. Cavalcante, Selection of time series forecasting models based on performance information (2004) 2–7.

[21] U. Mori, A. Mendiburu, J. A. Lozano, Similarity measure selection for clustering time series databases, IEEE Transactions on Knowledge and Data Engineering 28 (2016) 181–195.

[22] J. Kanda, A. D. Carvalho, E. Hruschka, C. Soares, P. Brazdil, Neurocomputing Meta-learning to select the best meta-heuristic for the Traveling Salesman Problem : A comparison of meta-features, Neurocomputing 205 (2016) 393–406.

[23] A. Bagnall, J. Lines, W. Vickers, E. Keogh, The UEA and UCR Time Series Classification Repository, http://www.timeseriesclassification.com.

[24] J. Large, J. Lines, G. Oastler, M. Middlehurst, M. Flynn, A. Bostrom, P. Schäfer, C. Wei Tan, A. Bagnall, UEA time series classification weka-compatible Java toolbox, https://github.com/uea-machine-learning/tsml (2017).

[25] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, F. J. Király, sktime: A Unified Interface for Machine Learning with Time Series, 2019.

[26] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers (1993).

[27] G. H. John, P. Langley, Estimating continuous distributions in bayesian classifiers, in: Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, 1995, pp. 338–345.

[28] I. BenGal, Bayesian networks, in: Bayesian networks, Encyclopedia of Statistics in Quality and Reliability, 2008.

[29] C. Cortes, V. Vapnik, Support-vector networks, Machine learning (1995) 273–297.

[30] J. J. Rodriguez, L. I. Kuncheva, C. J. Alonso, Rotation forest: A new classifier ensemble method, IEEE transactions on pattern analysis and machine intelligence (2006) 1619–1630.

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

[32] H. Taud, J. F. Mas, Multilayer perceptron (mlp), Geomatic Approaches for Modeling Land Change Scenarios (2018) 451–455.

[33] D. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, Workshop on Knowledge Knowledge Discovery in Databases 398 (1994) 359–370.

[34] Y. S. Jeong, M. K. Jeong, O. A. Omitaomu, Weighted dynamic time warping for time series classification, Pattern Recognition 44 (2011) 2231–2240.

[35] P. F. Marteau, Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 306–318.

[36] A. Stefan, D. G. Athitsos V, The move-split-merge metric for time series, IEEE Transactions on Knowledge and Data Engineering (2013) 1425–1438.

[37] G. E. Batista, E. J. Keogh, O. M. Tataw, V. M. De Souza, Cid: an efficient complexity-invariant distance for time series, Data Mining and Knowledge Discovery (2014) 634–669.

[38] L. Chen, R. Ng, On The Marriage of Lp-norms and Edit Distance, Proceedings of the Thirtieth international conference on Very large data bases (2004) 792–803.

[39] T. Górecki, M. Łuczak, Using Derivatives in Time Series Classification, Data Mining and Knowledge Discovery (2013) 310–331.

[40] T. Górecki, Using derivatives in a longest common subsequence dissimilarity measure for time series classificatio, Pattern Recognition Letters 45 (2014) 99–105.

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

[42] T. Rakthanmanon, E. Keogh, Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets, Proceedings of the 13th ICDM International Conference on Data Mining (2013) 668–676.

[43] J. Hills, J. Lines, E. Baranauskas, J. Mapp, A. Bagnall, Classification of time series by shapelet transformation, Data Mining and Knowledge Discovery 28 (2014) 851–881.

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

[45] P. Schäfer, The BOSS is concerned with time series classification, Data Mining and Knowledge Discovery (2015) 1505–1530.

[46] 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) (2017) 1578–1585.

[47] B. Fawaz, H. I.and Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, F. Petitjean, Inceptiontime: Finding alexnet for time series classification, Data Mining and Knowledge Discovery (2020) 1936–1962.

[48] D. S. Hirschberg, Algorithms for the longest common subsequence problem, Journal of the ACM (JACM) (1977) 664–675.

[49] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Mining and Knowledge Discovery (2015) 29(3), 565–592.

[50] R. G. Baydogan, M. G., E. Tuv, A bag-of-features framework to classify time series, IEEE transactions on pattern analysis and machine intelligence (2013) 35(11), 2796–2802.

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

[52] R. J. Kate, Using dynamic time warping distances as features for improved time series classification, Data Mining and Knowledge Discovery 30 (2015) 283–312.

[53] 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 (2016) 1548–1549.

[54] S. le Cessie, J. van Houwelingen, Ridge estimators in logistic regression, Applied Statistics 41 (1992) 191–201.

[55] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. E. Batista, The UCR time series classification archive (2015).

[56] J. Grabocka, N. Schilling, M. Wistuba, L. Schmidt-Thieme, Learning time-series shapelets (2014) 392–401.

[57] P. Senin, S. Malinchik, SAX-VSM : Interpretable Time Series Classification Using SAX and Vector Space Model (2013) 1175–1180.

[58] H. Bensusan, C. Giraud-carrier, Discovering Task Neighbourhoods through Landmark Learning Performances A Set of Landmarkers (2000).

[59] C. Soares, J. Petrak, P. Brazdil, Sampling-based relative landmarks : systematically test-driving algorithms before choosing (2001) 88–95.

[60] R. Leite, P. Brazdil, Improving progressive sampling via meta-learning on learning curves (2004) 250–261.

[61] L. Todorovski, S. Džeroski, Experiments in meta-level learning with ILP (1999) 98–106.

[62] M.-l. Zhang, Z.-h. Zhou, M L-KNN : A lazy learning approach to multi-label learning, Pattern Recognition 40 (2007) 2038–2048.

[63] A. Kalousis, Algorithm Selection via Meta-Learning, Ph.D. thesis, University of Geneva, 2002.

[64] H. Bensusan, A. Kalousis, Estimating the predictive accuracy of a classifier (2001) 25–36.

[65] R. Fagin, R. Kumar, D. Sivakumar, Comparing top k lists, SIAM Journal on discrete mathematics 17 (2003) 134–160.

[66] E. Alcobaca, F. Siqueira, A. Rivolli, L. P. F. Garcia, J. T. Oliva, A. C. P. L. F. de Carvalho, Mfe: Towards reproducible meta-feature extraction, Journal of Machine Learning Research 21 (2020) 1–5.

[67] M. Stone, Cross-validatory choice and assessment of statistical predictions, Journal of the Royal Statistical Society, Series B (Statistical Methodology) (1974) 36(2):111–147.