Knowledge-Based Systems 240 (2022) 108101

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Meta-features for meta-learning

Adriano Rivolli [a,c], Luís P.F. Garcia [b,*], Carlos Soares [d], Joaquin Vanschoren [e], André C.P.L.F. de Carvalho [c]

[a] Universidade Tecnológica Federal do Paraná (UTFPR), Câmpus Cornélio Procópio, Av. Alberto Carazzai, 1640, Cornélio Procópio, Paraná 86300-000, Brazil
[b] Department of Computer Science, University of Brasília, Asa Norte, Brasília, Distrito, Federal 70910-900, Brazil
[c] Institute of Mathematical and Computer Sciences, University of São Paulo, Av. Trabalhador São-carlense, 400, São Carlos, São, Paulo 13560-970, Brazil
[d] Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
[e] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, Netherlands

ARTICLE INFO

Article history:
Received 2 August 2021
Received in revised form 15 December 2021
Accepted 30 December 2021
Available online 7 January 2022

Keywords:
Meta-features
Characterization measures
Meta-learning
Classification problems

ABSTRACT

Meta-learning is increasingly used to support the recommendation of machine learning algorithms and their configurations. These recommendations are made based on meta-data, consisting of performance evaluations of algorithms and characterizations on prior datasets. These characterizations, also called meta-features, describe properties of the data which are predictive for the performance of machine learning algorithms trained on them. Unfortunately, despite being used in many studies, meta-features are not uniformly described, organized and computed, making many empirical studies irreproducible and hard to compare. This paper aims to deal with this by systematizing and standardizing data characterization measures for classification datasets used in meta-learning. Moreover, it presents an extensive list of meta-features and characterization tools, which can be used as a guide for new practitioners. By identifying particularities and subtle issues related to the characterization measures, this survey points out possible future directions that the development of meta-features for meta-learning can assume.

1. Introduction

Machine learning (ML) algorithms have an inductive bias: they each make assumptions about the data distribution and choose specific generalization hypotheses over several other possible generalizations, thus restricting the search space [1,2]. As true data distribution is unknown, several techniques are typically tried to achieve a satisfactory solution for a particular task. This trial-and-error approach is laborious and subjective, given the many choices that need to be made. Alternatively, recommendation systems based on meta-learning (MtL) presents a data-driven, automatic selection of techniques, by using knowledge extracted from previous tasks [3].

MtL can be distinguished by the type of information used to learn from previous tasks [4]. E.g., the experience can be learned from: (i) model evaluations by recommending hyper-parameter values, configuration search spaces and optimization approaches for similar tasks [5]; (ii) previously successful models using transfer learning [6] and few-shot learning [7]; and (iii) task properties by recommending algorithms using data characterization and learning performance [8]. Moreover, MtL also supports Automated ML (AutoML) [9] which is a combination of techniques for algorithm selection, hyperparameter optimization and neural architecture searches to recommend ML pipelines. This paper is mainly concerned with investigating MtL recommendation systems based on knowledge extracted from task properties. We believe that a better data characterization can improve MtL recommendations, resulting in better performance.

These recommender systems require a systematic collection of dataset characteristics and an assessment of the corresponding performance of different ML algorithms when applied to these datasets [3]. The result is a dataset in the meta-level, where each meta-example represents one of the datasets in the collection. The predictive attributes of each meta-example are values, called meta-features, extracted from the corresponding dataset. The target attribute of each meta-example is the performance obtained by the ML algorithms when they are applied to the corresponding dataset. Next, an ML algorithm can be applied to the meta-dataset, inducing a meta-model that can be used in a recommendation system to select the most suitable algorithm(s) for a new dataset [10,11]. The information extracted by

* Corresponding author.
E-mail addresses: rivolli@utfpr.edu.br (A. Rivolli), luis.garcia@unb.br (L.P.F. Garcia), csoares@fe.up.pt (C. Soares), j.vanschoren@tue.nl (J. Vanschoren), andre@icmc.usp.br (A.C.P.L.F. de Carvalho).

https://doi.org/10.1016/j.knosys.2021.108101
0950-7051/

the meta-features plays a crucial role in the successful use of MtL [12–14].

In the early 2000s, meta-features were one of the main research topics in MtL, resulting in the proposal of the most significant (or well-known) measures currently used [15–17]. Afterward, empirical studies investigated the effectiveness of existing and new meta-features in different tasks [18–20]. Although the most relevant surveys of MtL [3,4,10,11,21–23] present some aspects of meta-features, they do it briefly. Most of them only list the standard groups and meta-features. As far as the authors know, there is no complete guide of meta-features in the ML literature, nor a comprehensive list of the characterization measures.

It is important to mention that as well as MtL, meta-features can also be used to summarize dataset benchmarks of ML experiments [11,24]; to pick up similar datasets for a specific study [25]; to explain the performance of ML algorithms [15,26,27]; to guide the creation of artificial datasets [11,28]. Furthermore, in AutoML [9], meta-features can be used to learn from past experiences and reduce the search space of possible solutions.

Given the importance of the meta-features for the success of an MtL experiment and their different utilities for the ML community, the main contributions of this survey are:

1. A comprehensive list and description of the existing meta-features, which can be used as a guide for their use in MtL.
2. A taxonomy to characterize meta-features. Thus, when a new meta-feature is proposed, the taxonomy can be used for its correct specification.
3. A detailed discussion about different aspects relative to the meta-features and the dataset characterization process, providing new insights and pointing out future works.
4. A summarized list of tools for extracting meta-features, which can support the work of ML scientists and practitioners.

The rest of the paper is structured as follows. Section 2 briefly summarizes the main aspects of MtL. Section 3 presents a formalization and a taxonomy for the meta-features discussed in this text. Section 4 presents a bibliographical synthesis that covers the state-of-the-art of meta-features. Section 5 discusses the main strengths, weaknesses and open issues of using meta-features in MtL experiments. Section 6 reviews the main tools available for characterizing datasets. Section 7 concludes this work by summarizing its main contributions and pointing out avenues for future research.

2. Meta-learning

ML algorithms learn from previous experience [29]. In MtL, these previous experiences are used to learn from the learning process itself, also known as *learning to learn* [4]. Given that each ML algorithm has a particular inductive bias, and based on the *No Free Lunch* theorem [2,30], which states that no ML algorithm outperforms all other ML algorithms for all problems, an alternative to improve predictive performance for a new dataset is to predict and recommend the most suitable ML algorithm for this dataset. Although it cannot be ensured that the most suitable algorithm will be recommended, previous learning experiences can be used to recommend what is believed to be the most suitable ML algorithm, among a set of ML algorithms. For this recommendation, an ML algorithm can be applied to previous learning experiences, inducing a predictive meta-model, which can be used to predict the most suitable ML algorithm for a new dataset. MtL investigated the induction of predictive meta-models able to provide this recommendation. As a result, MtL can be used to automatize, speed up and improve the design of ML-based solutions [4,21,31].

MtL fits the abstract model for algorithm selection and recommendation systems proposed by Rice [32] and adapted by Smith-Miles [10], which includes the following components: the problem space P; the feature space F; the ML algorithm space A; the performance space Y; and the ML algorithm used for MtL. Following a data-driven process, the performance (Y) of a set of algorithms (A) over several datasets (P) is associated with characteristics of such datasets (F), represented by meta-features. A learning model induced from this meta-data can recommend a suitable algorithm for a new dataset.

The meta-features used in an MtL system depend on the problem domain since they must capture characteristics that may impact the performance of the considered ML algorithms. Thus, meta-features have to be representative to allow the recommendation of the most suitable algorithm [33,34], among a set of ML algorithms, with an acceptable computational cost to be extracted [13,17]. The characterization of the datasets has to demand less computation power than the trial and error approach, otherwise it becomes computationally infeasible. Moreover, the quality of predictive data is critical to the success of an ML and the success of an MtL application depends on the quality of the meta-features [12,13].

Although meta-features can be used for other purposes, MtL is the main reason for their proposal, investigation and improvement. In learning from task properties, some examples found in the literature comprise: hyperparameter tuning [35–37]; prediction of meta-features values [38,39]; induction of abstract meta-features [40]; performance prediction [41,42]; recommendation of classification [15,43], clustering [44–46], data transformation [47], feature selection [18], image segmentation [48] and noise detection [49] algorithms.

2.1. Meta-learning surveys

Previous MtL surveys provide a good overview of existing approaches, datasets, benchmarks [4,10,21,22] and MtL strategies for a specific type of recommendation [50,51]. While Vanschoren [4] presents a more general discussion about the types of MtL systems and their relation with learning to learn tasks, Vilalta and Drissi [21], Lemke et al. [22] and Smith-Miles [10] focus on constructing better MtL systems based on the knowledge extracted from task properties. Moreover, Rossi et al. [50] and Cunha et al. [51] are examples of surveys that discuss MtL for specific tasks such as recommender systems for data streams and collaborative filtering.

Vanschoren [4] defines MtL as learning to learn tasks based on prior experience on other tasks, systematically observing the differences of the ML algorithms in a wide range of learning tasks. The more similar the tasks, the better the meta-data can recommend ML algorithms for new tasks. Moreover, the author categorizes MtL according to three types of meta-data: (i) learn purely from model evaluations; (ii) characterize tasks to express task similarity and build meta-models; and (iii) transfer trained model parameters between tasks. Furthermore, the author claims that MtL can speed up and improve the design of ML pipelines or neural network architectures, avoiding manual approaches, in a data-driven way.

Vilalta and Drissi [21] and Lemke et al. [22] focus on MtL systems based on the knowledge extracted from task properties. While Vilalta and Drissi [21] show how to build self-adaptive learners, algorithms that improve their bias dynamically through experience by accumulating meta-knowledge, Lemke et al. [22] present the boundaries and definitions of MtL and the practical considerations to be taken into account when designing an MtL

system. The authors also discuss the intersection between common themes such as ensembles methods and algorithms with dynamic bias selection.

Smith-Miles [10] surveys Rice's Algorithm Selection Problem [32], the first study on automated algorithm selection models. The author improved and unified Rice's framework considering the Algorithm Selection Problem as a MtL learning problem. The proposed framework included learning phases from the meta-data and refinement and theoretical support. The evaluation of the performance of the refined algorithms closes the loop. The generalization allowed this framework to tie together the cross-disciplinary developments.

Regarding the MtL application scenario, Rossi et al. [50] and Cunha et al. [51] are surveys of data streams and collaborative filtering, respectively. While Rossi et al. [50] provide a set of guidelines to characterize non-stationary data and classify the characteristics regarding their dependence on data morphology, Cunha et al. [51] focus on providing a systematic literature review for MtL for collaborative filtering. These algorithms employ user feedback and their similarity to recommended items.

There are other surveys covering different perspectives, such as prior models using transfer learning, few-shot learning and configuration search spaces between similar tasks using strategies such as optimization approaches for AutoML [52–56]. It is important to observe that this paper is concerned with MtL recommendation systems based on the knowledge extracted from task properties. Thus, data characterization is the main goal to improve MtL recommendations.

3. Taxonomy

Let $\mathcal{D}$ be a dataset with n instances, such that $\mathcal{D} = \{(\vec{x}_i, y_i) \mid 1 \leq i \leq n\}$. Each instance $\vec{x}_i = [v_{i1}, v_{i2}, \ldots, v_{id}]$ is a vector with d predictive attribute values, paired with a target value, y_i. A meta-feature $f \in F$ is a function $f : \mathcal{D} \rightarrow \mathbb{R}^k$ that, when applied to a dataset $\mathcal{D}$, returns a set of k values that characterize the dataset, and that are predictive for the performance of algorithms when they are applied to the dataset. Function f can be detailed as

$$f(\mathcal{D}) = \sigma(m(\mathcal{D}, h_m), h_s), \tag{1}$$

such that $m : \mathcal{D} \rightarrow \mathbb{R}^{k'}$ is a characterization measure; $\sigma : \mathbb{R}^{k'} \rightarrow \mathbb{R}^k$ is a summarization function; h_m and h_s are hyperparameters used for m and σ, respectively. Note that k' can be different than k. The summarization function is required in propositional scenarios when a fixed cardinality k is needed [57], regardless of the value of k'.

Traditionally, no distinction has been made between the concepts of a meta-feature, f, and a characterization measure, m, in the literature. This may be natural when a measure results in a single value ($k' = k = 1$) and σ is the identity function, thus $f = m$. However, when a measure m can extract more than one value from each dataset, i.e. k' can vary according to $\mathcal{D}$, these values still need to be mapped to a vector of fixed length k. For instance, when a characterization can be computed per attribute (e.g., the mutual information between an attribute and the target), many authors use $f \approx mean(m)$ [17,43,58]. Other common summarization functions are histograms [59], minimum and maximum [60], and skewness and kurtosis [61]. They are detailed in Section 4.1.

These definitions allow the categorization of meta-features in a well-defined taxonomy, illustrated in Table 1. In this framework, all characterization measures are themselves described in terms of their required *inputs* and *outputs*. While some of these categories are only descriptive, others define whether or not a meta-feature is suitable for a specific scenario.

Table 1
Categories used to describe a measure or group of measures.

Level	Category Name	Options
Input	Task	Classification Supervised Any
	Extraction	Direct Indirect
	Argument	n Predictive Attributes (nP) All Predictive Attributes (∗P) Target Attribute (T)
	Domain	Numerical Categorical Both
	Hyperparameters	Yes, No
Output	Range	[min, max]
	Cardinality	k
	Deterministic	Yes, No
	Exception	Yes, No

Some measures are restricted to specific tasks, such as *classification*. Others can be more generically applied to *supervised* tasks, which includes regression problems. The measures classified as *any* are the most general and can also be applied to unsupervised tasks such as clustering, and semi-supervised problems. In *supervised* tasks, a target attribute is required to evaluate the meta-features, which is not necessary for meta-features of the type *any*.

The cardinality defines the number of possible values returned by a measure. A distinction between single-valued measures ($k = 1$) and multi-valued measures ($k > 1$) is important for data analysis, mainly to define whether or not a summarization function must be applied. For most of the multi-valued measures, the cardinality is related to aspects such as the number of instances, attributes or classes in the considered datasets.

Some measures are *non-deterministic*, meaning that there is no guarantee that the same result will be obtained for the same input in different runs. When reproducibility is necessary, the same randomization seed must be used for each run or the measures must be executed a number of times and averaged to account for the randomization effect.

Finally, while some measures are *robust*, others can generate *exceptions* for certain datasets, leading them not to emit valid values in all cases. This can occur in particular conditions, such as a division by zero or a logarithm of a negative number. Although it is a technical detail, its occurrence during a characterization process might result in missing values in a meta-base, which eventually need to be treated.

4. Meta-features

A fundamental question in MtL is how to extract suitable information to characterize specific tasks. Researchers have been trying to answer this question by looking for dataset properties that can affect learning algorithm performance, measuring this performance outright [15,62], investigating alternatives [33,63] and adapting/creating new measures based on existing ones [17, 58].

In all cases, the meta-features were organized in groups. These groups are subsets of data characterization measures [3] that share similarities among them. However, they are not always clearly and strictly delimited. Hence, when two different studies mention using a certain group of measures, it does not mean that they use exactly the same measures [10]. Additionally, different names have been used to describe these groups of measures. In this work, we propose to organize the measures in six groups:

Simple: measures that are easily extracted from data [20], commonly known, and do not require significant computational resources [34]. They are also called *general* measures [17].

Statistical: measures that capture the statistical properties of the data [20]. These measures capture data distribution indicators, such as average, standard deviation, correlation and kurtosis. They are computed in numerical attributes only [17].

Information-theoretic: measures from the information theory field [17]. These measures are based on entropy [64], capturing the amount of information in the data and their complexity [10]. They can be used to characterize discrete attributes.

Model-based: measures extracted from a model induced using the training data [20]. Many of these are based on properties of decision tree (DT) models [16,62], referred to as *decision-tree-based* meta-features [62]. Properties extracted from other models are also used [18].

Landmarking: measures that use the performance of simple and fast learning algorithms to characterize datasets [10]. The algorithms must have different inductive biases and should capture relevant information with a low computational cost [33,65].

Others: measures not included in the previous groups, such as standalone measures, time-related measures [19], concept and case-based measures [11,66], clustering and distance based measures [44,46,67], among others.

The first three groups represent the most common and traditional approaches for data characterization [3]. They receive different names such as *direct characterization*, *basic* measures [18], *DCT* [16], *standard* [68] and *STATLOG* measures [10]. In earlier work, statistical measures were also called *discriminant* meta-features [69]. The next two groups depend on ML algorithms to extract model complexity or performance measures, while the last group includes characterizations for specific types of data, such as time series. Vanschoren [70] offers a more fine-grained categorization of meta-features, based on intrinsic biases of learning algorithms, such as *data normality*, *feature redundancy*, and *feature-target association*.

The rest of this section is organized as follows. Section 4.1 describes the summarization functions and Sections 4.2 to 4.7 provide a systematic definition and description of these measures using the taxonomy shown in Table 1. The formal definition of each measure is available in Appendix. In the descriptions, $-\infty$ and ∞ are used when it is not possible to define the range of a measure, whereas *inherited* is used when the measure range is defined by the value range of specific dataset attributes. The use of an upper stroke bar in the range and cardinality indicates an approximated value. When the columns *Extract*, *Domain*, *Hyperp.*, *Excep.* and *Det.* describe a constant property, they are suppressed from the tables and identified in the caption. The section starts with a description and an analysis of the main summarization functions, considering that they are part of the meta-feature definition (Eq. (1)).

4.1. Summarization functions

In this study, summarization functions are used to normalize the cardinality of meta-features and to characterize other meta-feature aspects, such as tendency, distribution and variability of the results. Given that many measures are multi-valued and that their cardinalities vary according to the dataset, comparisons between multiple datasets can be infeasible. Consequently, the summarization transforms non-propositional data to propositional [60], making them suitable to be organized in a meta-base, for instance. In the literature, summarization functions have been called meta-level attributes [60], meta^2-features [61] and post-processing functions [71].

It is worth noting that in some studies [17,18,72], to cite a few, the mean function is used as part of the meta-feature definition and it is the only way used to summarize the results. Other studies have used distinct subsets of summarization functions, such as histogram [59]; minimum, mean and maximum [60]; minimum, maximum, mean and standard deviation [49,73,74]; mean, standard deviation and quartiles 1, 2 and 3 [42]; minimum, maximum, mean and standard deviation, kurtosis and skewness [61].

Table 2 presents a non-exhaustive list of the summarization functions, their range, cardinality and a brief description. The *quantiles* and *histogram* result in multiple values. The former summarizes a measure by representative values of the measure distribution, whereas the latter uses the proportion of values in each range of data. A hyperparameter specifying the number of bins in which the results are split [59] defines the cardinality of the *histogram*. Some functions such as *count*, *histogram* and *kurtosis* change the range of the characterized measure, while others inherit the range of the measure that they summarize, such as *max*, *mean* and *min*. The *identity function* is conceptually used when a characterization measure results in a single value ($k' = 1$).

Pinto et al. [71] proposed that the summarization functions should be organized in groups: *descriptive statistical* includes the most common functions and summarizes a set of values in a single result such as *max*, *min*, *mean*, *median*, *sd*, *skewness*, *kurtosis*, *iqRange*, among others; *distribution* characterizes the distribution of the measure using multiple values. For this purpose, the use of histogram with a fixed number of bins [59] and the use of quartiles to summarize the set of values [42] are alternatives observed in the literature; the *hypothesis test* assesses an assumption about a set of values, resulting in one or more values, as the p-values and/or the test results. However, its use has not been observed in the literature.

Conceptually, any function that offers guarantees of a fixed cardinality, regardless of the number of values received by it, can be applied as a summarization function. Thus, even though a *post-processing* function [71] can also generate an indiscriminate number of values, a summarization function cannot. The summarization functions presented in Table 2 can be applied to all multi-valued measures indiscriminately. Some combinations of the measure/summarization-function explore semantic concepts, e.g. the standard deviation of the proportion of classes [69] indicates imbalanced class distribution. Summarization functions can be created for a specific measure or group of measures. An example would be a function that counts the number of values larger than a specific prefixed threshold. However, this approach is not common when the main goal is to summarize all characterization measures and not only a group or few measures. Section 5.5 addresses this matter as an open issue and shows possible insights concerning their use and exploration.

Table 2
Main summarization functions.

Acronym	Range	Cardinality	Brief description
count	$[1, k]$	1	Computes the cardinality of the measure, suitable when the cardinality is a variable.
histogram	$[0, 1]$	*user*	Describes the distribution of the measured values, suitable for measures with high cardinality.
iqRange	$[0, \infty]$	1	Computes the interquartile range of the measured values.
kurtosis	$[-3, \infty]$	1	Describes the shape of the measured value distribution.
max	*inherited*	1	Results in the maximum values of the measure.
mean	*inherited*	1	Computes the averaged values of the measure.
median	*inherited*	1	Results in the central value of the measure.
min	*inherited*	1	Results in the minimum value of the measure.
quartiles	*inherited*	5	Results in the minimum, first quartile, median, third quartile and maximum of the measured values.
range	$[0, \infty]$	1	Computes the range of the measured values.
sd	$[0, \infty]$	1	Computes the standard deviation of the measured values.
skewness	$[-\infty, \infty]$	1	Describes the distribution shape of the measured values in terms of symmetry.

Table 3
Simple measures and their characteristics. They are directly extracted, deterministic and free of hyperparameters.

Acronym	Task	Argument	Domain	Range	Card.	Excep.
attrToInst	Any	∗P	Both	$[0, \overline{d}]$	1	No
catToNum	Any	∗P	Both	$[0, \overline{d}]$	1	Yes
classToAttr	Classif.	∗P+T	Both	$[0, q]$	1	No
freqClass	Classif.	T	Categ.	$[0, 1]$	q	No
instToAttr	Any	∗P	Both	$[0, \overline{n}]$	1	No
instToClass	Any	∗P+T	Both	$[1, \overline{n}]$	1	No
nrAttr	Any	∗P	Both	$[1, +\infty]$	1	No
nrAttrMissing	Any	∗P	Both	$[0, d]$	1	No
nrBin	Any	∗P	Both	$[0, d]$	1	No
nrCat	Any	∗P	Both	$[0, d]$	1	No
nrClass	Classif.	T	Categ.	$[2, \overline{n}]$	1	No
nrInst	Any	∗P	Both	$[q, +\infty]$	1	No
nrInstMissing	Any	∗P	Both	$[0, n]$	1	No
nrMissing	Any	∗P	Both	$[0, \overline{dn}]$	1	No
nrNum	Any	∗P	Both	$[0, d]$	1	No
numToCat	Any	∗P	Both	$[0, \overline{d}]$	1	Yes

4.2. Simple meta-features

The simple measures, listed in Table 3, are directly extracted from the data and represent basic information about the dataset. They are the simplest set of measures in terms of definition and computational cost [17,20,34,75]. They are also deterministic and free of hyperparameters. Semantically, the measures are related to the number of predictive attributes, instances, target classes and missing values.

The measures related to attributes are: number of attributes (*nrAttr*); number of binary attributes (*nrBin*); number of categorical attributes (*nrCat*); number of numeric attributes (*nrNum*); proportion of categorical versus numeric attributes (*catToNum*) and vice-versa (*numToCat*). These measures are relevant to characterize the main aspects of a dataset, providing information that can support the choice of an algorithm for a particular learning task.

The number of instances (*nrInst*) and the number of classes (*nrClass*) indicate the dataset size and its label diversity. When combined with the *nrAttr*, we can define *attrToInst* and *instToAttr*, which represent the dimensionality and sparsity of the data, respectively. The latter is a potential indicator for overfitting when its value is too small [72]. The number of classes per attribute (*classToAttr*) and instances per classes (*instToClass*) measure properties of the target attribute distribution, such as class imbalance. Likewise, the frequency of instances in each class (*freqClass*) allows the extraction of measures such as the proportional frequency of the majority and minority class [43], default accuracy/error [16] and standard deviation of the class distribution [69]. When combined with summarization functions, it can describe imbalanced learning scenarios.

Finally, some measures assess dataset quality, such as the number of attributes (*nrAttrMissing*) and instances (*nrInstMissing*) with missing values, as well as the total number (*nrMissing*). As some ML algorithms can deal with missing values better than others, these measures can provide important information for algorithm selection, or indicate when data treatment is necessary, as discussed in Section 5.1.

Some authors have proposed modified versions of these measures. For instance, Todorovski et al. [60] use the log of the number of instances, Brazdil et al. [31] use the proportion of categorical attributes, and Kalousis and Hilario [76] use the proportion of numerical attributes. These modifications can be defined with summarization functions.

When using these measures for MtL, it may be necessary to normalize their values. With the exception of *nrAttr* and *nrInst*, they can be normalized using their theoretical maximum value, as shown in the column named Range in Table 3. When the dataset has only numerical attributes, *numToCat* cannot obtain a valid value. The same occurs with *catToNum* when the dataset has only categorical attributes.

4.3. Statistical meta-features

Statistical measures can extract information about the performance of statistical algorithms [75] or about data distribution, for instance, central tendency and dispersion [17]. They are the largest and the most diversified group of meta-features, as shown in Table 4. Statistical measures are deterministic and support only numerical attributes. Some measures require the definition of hyperparameter values, while others can generate exceptions, e.g. caused by division by zero. Some of them are indirectly extracted, and are closely related to the discriminant group reported in Lindner and Studer [69]. The others can be widely applied as they only use predictive attributes as input.

Correlation (*cor*) and covariance (*cov*) capture the interdependence of the predictive attributes [75]. They are computed for each pair of attributes in the dataset, resulting in $(d-1)/2$ values.

Table 4
Statistical measures and their characteristics. They are deterministic and only accept numerical attributes.

Acronym	Task	Extract	Argument	Hyperp.	Range	Card.	Excep.
canCor	Classif.	Indirect	∗P+T	No	[0, 1]	$\overline{d}$	No
cor	Any	Direct	2P	Yes	[0, 1]	$\overline{d^2}$	Yes
cov	Any	Direct	2P	No	$[0, \infty]$	$\overline{d^2}$	No
nrDisc	Classif.	Indirect	∗P+T	No	$[0, d]$	1	No
eigenvalues	Any	Indirect	∗P	No	$[0, \infty]$	$\overline{d}$	No
gMean	Any	Direct	1P	No	$[0, \infty]$	*d*	Yes
hMean	Any	Direct	1P	No	*inherited*	*d*	No
iqRange	Any	Direct	1P	No	$[0, \infty]$	*d*	No
kurtosis	Any	Direct	1P	No	$[-3, \infty]$	*d*	Yes
mad	Any	Direct	1P	No	$[0, \infty]$,	*d*	No
max	Any	Direct	1P	No	*inherited*	*d*	No
mean	Any	Direct	1P	No	*inherited*	*d*	No
median	Any	Direct	1P	No	*inherited*	*d*	No
min	Any	Direct	1P	No	*inherited*	*d*	No
nrCorAttr	Any	Direct	∗P	Yes	[0, 1]	1	Yes
nrNorm	Any	Direct	∗P	Yes	$[0, d]$	1	No
nrOutliers	Any	Direct	∗P	Yes	$[0, d]$	1	No
range	Any	Direct	1P	No	$[0, \infty]$	*d*	No
sd	Any	Direct	1P	No	$[0, \infty]$	*d*	No
sdRatio	Classif.	Indirect	∗P+T	No	$[1, \infty]$	1	Yes
skewness	Any	Direct	1P	No	$[-\infty, \infty]$	*d*	Yes
tMean	Any	Direct	1P	Yes	*inherited*	*d*	No
var	Any	Direct	1P	No	$[0, \infty]$	*d*	No
wLambda	Classif.	Indirect	∗P+T	No	[0, 1]	1	No

The former is a normalized version of the latter, and the absolute value of both measures are frequently used, which changes the range from $[-1, 1]$ and $[-\infty, \infty]$, respectively, to the values reported in Table 4. To summarize the number of attributes with a high correlation, *nrCorAttr* computes the proportion of highly correlated attribute pairs (e.g., $cor > 0.5$). High values indicate a strong correlation between the attributes, which can be interpreted as a level of redundancy in the data [76]. Thus, low values can be better than high ones as the data is more discriminative. Even though they ignore the target attribute, the summarization of these values describes potentially useful information regarding the learning problem.

Most statistical measures are extracted for each attribute separately. Measures of central tendency consist of the *mean* and its variations such as the geometric mean (*gMean*), harmonic mean (*hMean*) and trimmed mean (*tMean*); and the *median*. Measures of dispersion consist of the interquartile range (*iqRange*), *kurtosis*, maximum (*max*), median absolute deviation (*mad*), minimum (*min*), *range*, standard deviation (*sd*), *skewness* and variance (*var*). While one points to the center of a distribution, the other shows how much the values are spread from the center, complementing themselves. Their range depends directly on the attributes' range, with few exceptions such as kurtosis and skewness. These two, are suitable to capture the normality of the data attributes [70].

A specific measure to capture the normality of the attributes is the *nrNorm*, which computes the number of attributes normally distributed. Similarly, *nrOutliers* counts the number of attributes that contain outliers. Normality and outliers may impact the behavior of learning algorithms, which make these measures useful in an MtL scenario.

The values of the measures of central tendency and dispersion varied according to each attribute. Given that they ignore the target value, it is plausible to assume that the information captured by these measures is possibly irrelevant for a learning algorithm; e.g. what is the difference between a dataset with attributes whose *mean* is high or low? However, when the measures are relativized among themselves, more useful information could be captured. At least, the comparison between different datasets will make more sense. As an example, the relative discrepancy between the values of *mean* and *median* is an indicative of outliers in the attribute; when two datasets are compared, one can observe in which dataset this variation is larger. However, such relativization has not been observed in the literature yet.

The discriminant statistical measures present some specificities such as being exclusively used for classification tasks. By considering the target value and using the whole dataset as input, they result in a single value. Canonical correlations (*canCor*), the number of discriminant values (*nrDisc*), the homogeneity of covariances (*sdRatio*) and the Wilks' lambda (*wLambda*) represent the discriminant measures. Finally, the *eigenvalues* from the covariance matrix only use the predictive data to be computed.

The canonical correlation meta-features (*canCor* and *nrDisc*) capture the amount of useful information contained in the attributes [77], therefore many and few values demonstrate that the predictive attributes can represent the target well. On the other hand, despite ignoring the target, the eigenvalues indicate how spread out the data is over the principal components. High values indicate a good separability, even though it does not indicate how discriminatory the data is with regard to the target. In turn, when the *sdRatio* is close to 1, there is not much difference between the covariance matrices within the different groups of instances (segmented by the classes) [75]. Values strictly larger than 1 indicate the opposite, possibly revealing a better division between the classes. Finally, a 0 value for *wLambda* indicates that there is a total discrimination from the variables, and 1 means no discrimination at all.

Concerning the hyperparameters, different correlation methods such as Pearson's correlation, Kendall's τ and Spearman's ρ coefficient [78], can be used to compute the *cor* measure. This is also applied to the *nrCorAttr* measure, which additionally requires a threshold value to define high correlations. The *tMean* requires defining how much data should be discarded to compute the mean. Finally, the *nrNorm* and *nrOutliers* are dependent on the algorithm to compute whether or not a distribution is normal and has outliers. Even though *skewness* and *kurtosis* could be seen as algorithm dependent, their variations do not produce observable differences for large samples of data [79].

Some measures can throw exceptions and due to this are not calculated correctly. The *cor*, *kurtosis*, *nrCorAttr* and *skewness* could generate an error with a constant attribute caused by division by zero. The *sdRatio* uses *log* in this formulation, and the possibility of obtaining a negative value makes the measure error-prone. The *gMean* can be computed in 2 different ways and both

Table 5
Information-theoretic meta-features and their characteristics. They are directly extracted, free of hyperparameters, robust, deterministic and support only categorical attributes.

Acronym	Task	Argument	Range	Card.
attrEnt	Any	1P	$[0, \log_2(n)]$	*d*
classEnt	Classif.	T	$[0, \log_2(q)]$	1
eqNumAttr	Classif.	∗P+T	$[0, \infty]$	1
jointEnt	Classif.	1P+T	$[0, \log_2(n)]$	*d*
mutInf	Classif.	1P+T	$[0, \log_2(n)]$	*d*
nsRatio	Classif.	∗P+T	$[0, \infty]$	1

can generate errors, one using product and another using *log*. The former can obtain arithmetic overflow/underflow while the latter cannot support negative values.

As the majority of the statistical measures do not consider the class information, Castiello et al. [17] proposed an indirect way to explore it. This approach splits the dataset according to the class labels and computes the measures for each subset. However, the authors are not aware of any empirical evaluation of this approach. Besides, many statistical measures need to be summarized as several possible values can be obtained.

Instances from each class may present differences in relation to one or more measures. However, by summarizing the values of the same measure for different groups of instances together, either piece of information is possibly lost or the differences observed are mitigated. Regardless of this, empirical studies concerning this approach and new alternatives to summarize these values are unexplored in the literature.

Finally, it is important to observe that the statistical measures only support numerical attributes. Datasets that contain categorical data must be either partially ignored or converted to numerical values.

4.4. Information-theoretic meta-features

Information-theoretic meta-features capture the amount of information in the data. Table 5 shows the information-theoretic measures, which require categorical attributes and most of them are restricted to representing classification problems. Moreover, they are directly computed, free of hyperparameters, deterministic and robust. Semantically, they describe the variability and redundancy of the predictive attributes to represent the classes.

The entropy of the predictive attributes (*attrEnt*) and the target values (*classEnt*) capture the average uncertainty in the predictive and class attributes [64], respectively. In the former, all predictive attributes are assessed, thus its summarization can provide an overview of the attributes' capacity for class discrimination. In the latter, it represents how much information, on average, is necessary to specify one class [17]. In a learning perspective, a predictive attribute with a low entropy contains a low discriminatory power [75], whereas a target attribute with low entropy contains a high level of purity. These measures are usually normalized.

The joint entropy (*jointEnt*) and the mutual information (*mutInf*) compute the relationship of each attribute with the target values. While the former captures the relative importance of the predictive attributes to represent the target [68], the latter represents the common information shared between them, indicating their degree of dependency [75].

Finally, the equivalent number of attributes (*eqNumAttr*) and the noise signal ratio (*nsRatio*) capture information that is related to the minimum number of attributes necessary to represent the target attribute and the proportion of data that are irrelevant to describe the problem [80], respectively.

Table 6
Model-based meta-features and their characteristics. These meta-features are indirectly extracted, robust, deterministic, require the definition of hyperparameters and support both attribute types.

Acronym	Task	Argument	Range	Card.
leaves	Sup.	∗P+T	$[q, \overline{n}]$	1
leavesBranch	Sup.	∗P+T	$[1, \overline{n}]$	$\overline{n}$
leavesCorrob	Sup.	∗P+T	$[0, 1]$	$\overline{n}$
leavesHomo	Sup.	∗P+T	$[q, +\infty]$	$\overline{n}$
leavesPerClass	Classif.	∗P+T	$[0, 1]$	*q*
nodes	Sup.	∗P+T	$[q, \overline{n}]$	1
nodesPerAttr	Sup.	∗P+T	$[0, \overline{n}]$	1
nodesPerInst	Sup.	∗P+T	$[0, 1]$	1
nodesPerLevel	Sup.	∗P+T	$[1, \overline{n}]$	$\overline{n}$
nodesRepeated	Sup.	∗P+T	$[0, \overline{n}]$	$\overline{d}$
treeDepth	Sup.	∗P+T	$[1, \overline{n}]$	$\overline{n}$
treeImbalance	Sup.	∗P+T	$[0, 1]$	$\overline{n}$
treeShape	Sup.	∗P+T	$[0.0, 0.5]$	$\overline{n}$
varImportance	Sup.	∗P+T	$[0, 1]$	$\overline{d}$

It is important to add that in the experimental analyses carried out, when a dataset only has numerical attributes, they need to be discretized. Another alternative would be to know their data distribution and to apply a modified formulation of entropy [17]. How sensitive the measures are concerning this issue is not known. A detailed discussion about the support for different data types is presented in Section 5.1.

4.5. Model-based meta-features

The meta-features from this group are information extracted from a predictive learning model, in particular, a DT model. They characterize a dataset by how complex is the model induced, which, for DT, can be the number of leaves, the number of nodes and the shape of the tree. Table 6 shows the DT model meta-features. They are designed to characterize supervised problems, all measures are deterministic, robust and require the definition of hyperparameters: the DT induction algorithm (together with its hyperparameter values) used to induce the DT model.

The measures based on leaves are identified with the prefix *leaves*, which describe, to some extent, the complexity of the orthogonal decision surface. Some measures result in a value for each leaf, and those measures are the number of distinct paths (*leavesBranch*), the support described in the proportion of training instances to the leaf (*leavesCorrob*) and the distribution of the leaves in the tree (*leavesHomo*).

The proportion of leaves to the classes (*leavesPerClass*) represents the classes complexity and the result is summarized per class. While *leavesCorrob* and *leavesPerClass* have a fixed range independent of the dataset, *leaves* and *leavesBranch* have a maximum value limited by the number of instances. In practice, the most observed limit is associated with the number of attributes, which also determines the cardinality of them. Only *leavesHomo* does not have a defined limit of values.

The measures based on nodes, which extract information about the balance of the tree to describe the discriminatory power of attributes, are identified with the prefix *nodes*. Together with *nodes*, the proportion of nodes per attribute (*nodesPerAttr*) and the proportion of nodes per instance (*nodesPerInst*) result in a single value. The number of nodes per level (*nodesPerLevel*) and the number of repeated nodes (*nodesRepeated*) have the number of attributes at their maximum value. While *nodesPerLevel* describes how many nodes are present in each level, *nodesRepeated* represents the number of nodes associated with each attribute used for the model.

The measures based on the tree size, which extract information about the leaves and nodes to describe the data complexity,

Table 7
Common landmarking meta-features and their characteristics. They are indirectly extracted, non-deterministic and require defining the hyperparameters.

Acronym	Task	Argument	Domain	Range	Card.	Excep.
bestNode	Sup.	∗P+T	Both	[0, 1]	*user*	No
eliteNN	Sup.	∗P+T	Both	[0, 1]	*user*	No
linearDiscr	Sup.	∗P+T	Num.	[0, 1]	*user*	Yes
naiveBayes	Sup.	∗P+T	Both	[0, 1]	*user*	No
oneNN	Sup.	∗P+T	Both	[0, 1]	*user*	No
randomNode	Sup.	∗P+T	Both	[0, 1]	*user*	No
worstNode	Sup.	∗P+T	Both	[0, 1]	*user*	No

are identified with the prefix *tree*. The tree depth (*treeDepth*) represents the depth of each node and leaf, the tree imbalance (*treeImbalance*) describes the degree of imbalance in the tree and the shape of the tree (*treeShape*) represents the entropy of the probabilities to randomly reach a specific leaf in a tree from each one of the nodes.

Finally, the importance of each attribute (*varImportance*) represents the amount of information present in the attributes before a node split operation. The amount of information is defined by the randomization of incorrect labeling. This measure varies according to the DT algorithm. As an example, the C4.5 algorithm uses the information gain from the information-theoretic group to compute the importance of the attributes [62] and the CART algorithm uses the Gini index [81].

Other model-based measures, using different learners, such as *k*-Nearest Neighbors (kNN) and Perceptron neural networks were presented in Filchenkov and Pendryak [18]. However, some of these measures have a very high computational cost. Some others have the concept already described by well-known groups. Similarly, in Nguyen et al. [82], the weights learned by distinct feature selection algorithms were defined as model-based meta-features.

4.6. Landmarking meta-features

Landmarking is an approach that characterizes datasets using the performance of a set of fast and simple learners, different from the model-based meta-features, which extract information from the learning models. Although the performance of any algorithm can be used as landmarking, including sophisticated algorithms, some of them have been specifically used as meta-features. Table 7 lists the most common landmarking measures. They characterize supervised problems and are indirectly extracted, thus the whole dataset is used as an argument. They require defining hyperparameters: the learning algorithm; the evaluation measure to assess the model performance; and, the procedure used to compute them (e.g., cross-validation). While the range is dependent on the evaluation measure (usually between 0 and 1), the cardinality is from the procedure, thereby it is user-defined. Since their training and test data samples are randomly chosen, all landmarking are non-deterministic.

The measures *bestNode*, *randomNode* and *worstNode* are the performance of a DT-model induced using different single attributes. Respectively, they use the following attributes: the most informative, a random one, and the least informative attribute. The aim is to capture information about the boundary of the classes and combine this information with the linearity of the DT-models induced with the worst and random attributes. The DT algorithm is a hyperparameter defined by the user since different algorithms could be used.

The elite-Nearest Neighbor (*eliteNN*) is the result of the 1NN model using a subset of the most informative attributes in the dataset, whereas the one-Nearest Neighbor (*oneNN*) is the result of a similar learning model induced with all attributes. The distance measure used by the *k*NN algorithm is a hyperparameter.

The Linear Discriminant (*linearDiscr*) and the Naive Bayes (*naiveBayes*) algorithms use all attributes to induce the learning models. The first technique finds the best linear combination of predictive attributes able to maximize the separability between the classes. For such, it uses a covariance matrix and assumes that the data follows a Gaussian distribution. This technique can generate exceptions if the data has redundant attributes. The second technique is based on the Bayes' theorem and calculates, for each feature, the probability of an instance to belong to each class. The combination of all features and related probabilities for one instance that returns the class with the highest probability.

Concerning the hyperparameters, an evaluation measure such as accuracy, balanced accuracy and Kappa is necessary to evaluate the models. Other measures such as precision, recall and F1 could also be used, however, for them, it is necessary to identify the class of interest in binary datasets. The procedures used to induce the model are (i) using the whole instances to train and test; (ii) holdout; and, (iii) cross validation. This information is rarely mentioned in MtL studies and their impact in the characterization measures are not yet known. In practice, it represents a trade-off between stable measures and computational costs.

There are also relative and subsampling landmarkings [33,65]. Instead of using the absolute performance of the landmarkers as meta-features, a relative approach adopts the landmarkers' ranking, which is obtained using pairwise comparisons. Thus, the meta-feature can be a binary value indicating the winner, the difference between them or the ratio of the two performances. Besides, a meta-feature for each ranking position containing the name of the respective landmarker is of a categorical type. On the other hand, subsampling landmarking works by applying traditional algorithms to a reduced subset of the original dataset.

The performance of the landmarkers can be described as a learning curve, representing their use with different sampling sizes of a dataset [83]. Furthermore, in an algorithm recommendation scenario, their relative performance can be learned by meta-models and the prediction from these meta-models can be used as meta-features, analogous to a stacking-based approach [84]. It is an alternative to learning meta-features [4].

4.7. Other meta-features

Many other non-traditional characterization measures have been reported in the literature. Despite the fact they are not broadly used in MtL studies, e.g. due to a high computational complexity or domain bias, they can be useful for a particular learning scenario and MtL problem. Besides, some works show good results when using those characterization measures [44,85,86]. Here, they are arbitrarily presented in the following subgroups: Section 4.7.1 presents *clustering*-based meta-features; Section 4.7.2 describes meta-features based on *complexity* measures; and Section 4.7.3 has a *miscellaneous* collection of meta-features that do not fit in the previous groups.

Table 8
Clustering-based characteristics. They are robust and require defining the hyperparameters.

Acronym	Task	Extract	Argument	Domain	Range	Card.	Determ.
compactness	Any	Indirect	∗P	Both	$[0, \infty]$	k	Yes/No
connectivity	Any	Indirect	∗P	Both	$[0, n]$	1	Yes/No
nrClusters	Any	Indirect	∗P	Both	$[1, \overline{n}]$	1	No
nre	Any	Indirect	∗P	Both	$[0, \infty]$	1	Yes/No
purityRatio	Classif.	Indirect	∗P+T	Both	[0, 1]	q	No
sizeDist	Any	Indirect	∗P	Both	[0, 1]	k	No

4.7.1. Clustering

Clustering measures characterize the instance space using the premise that the instances are distributed into clusters. The clustering partitions can be defined by the distance between the instances, their density or a particular data distribution. Most clustering meta-features have been used to characterize unsupervised problems [44–46]. Thus, they use only the predictive attributes and are indirectly extracted, demanding a set of hyperparameter values, such as the clustering algorithm, its hyperparameters and, possibly, a distance function. By ignoring the classes to create partitions of instances, this group of measures can capture an intrinsic complexity present in the data [87].

In a classification scenario, under the assumption of some learning algorithms that the instances with the same classes are distributed in a mostly compact manner in their predictive attribute space, the partitions can be defined by the class. This approach reduces the complexity of extracting the meta-features as the clustering step is eliminated. However, we did not find a proper exploration of this approach in the literature.

Table 8 presents a list of clustering measures found in MtL studies [44–46,87]. Here, k denotes the number of clusters, such that $k \ll n$ and $k \approx q$ is a reasonable value to consider. If the partitions are defined by the classes, some measures will be deterministic. On the other hand, when they are obtained by a clustering algorithm, they will be non-deterministic. For this reason, these measures are indicated with a 'Yes/No' value in the deterministic column. According to the distance measure used, the meta-features can handle numerical and/or categorical attributes.

Given the data partition produced by a clustering algorithm, *nrCluster* represents the number of clusters, a simple informative measure, which is useful when this number is dynamically defined by the cluster algorithm used. Otherwise, when the number of clusters is a hyperparameter defined by the user, an intuitive option is to use the number of classes. The distribution of the clusters based on the instances' frequency is captured by the measure *sizeDist*. A distribution skewed to the right indicates a complex dataset [87]. If the classes are used to define the clusters, *sizeDist* becomes similar to the *freqClass*. Additionally, *purityRatio* looks at the instances' classes to evaluate the partitions. It is calculated for each class and captures the ratio of clusters that contain instances related to the respective class. Datasets with high values are more complex than those with low values since the classes are distributed across all partitions.

Different validation measures are used to represent the quality of the partitions obtained, such as how compact each group is and how separated the groups are from each other [46]. In a classification context, this information may indicate the separability of the instances, and possibly the classes. *Compactness* [46] measures how compact the clusters are. Lower values indicate compact groups. *Connectivity* captures local densities by counting the violations of the nearest neighbor relationship of instances in different partitions [88]. When normalized by the number of instances, high values indicate that the clusters are not well separated. It could be an informative measure to characterize the suitability of the bias related to instance-based learning algorithms. Normalized relative entropy (*NRE*) indicates how uniform the instances are distributed among the clusters. Values close to zero reveal well distributed clusters.

Although they were not included in Table 8, validation measures found in the clustering literature can be also used to characterize datasets in MtL studies [37]. There are many clustering internal and external validation indices [89] that could be used. The internal indices only consider the clusters to be computed, whereas the external indices require the class values to measure the quality of the partitions. For MtL, internal indices can be used without a clustering algorithm by considering the classes as partitions.

With few exceptions, all these measures (including validation indices) have a high asymptotic computational complexity, which restricts their use. Additionally, they allow a wide range of choices, with different impacts in the value returned. In spite of being able to provide a good characterization, clustering measures are underexplored in the MtL literature.

4.7.2. Complexity

Complexity measures were proposed in [90] to capture the underlying difficulty of classification tasks, considering aspects such as class overlapping, the density of manifolds and the shape of decision boundaries. They were used to support data preprocessing, ML and recommender systems [49,85,91,92]. While the complete survey of the complexity measures can be found in Lorena et al. [93], Table 9 summarizes the main characteristics of these measures.

Ho and Basu [90] divide the complexity measures into three groups: (i) feature overlapping measures; (ii) measures of the separability of classes; and (iii) geometry, topology and density of manifolds measures. Following Lorena et al. [93], we adopted a more granular organization: (i) feature-based measures; (ii) linearity measures; (iii) neighborhood measures; (iv) network measures; and (v) dimensionality measures.

Feature overlapping measures characterize how informative the predictive attributes are to separate the classes. They are: maximum Fisher's discriminant ratio (*F1*); directional-vector maximum Fisher's discriminant ratio (*F1v*); volume of overlapping region (*F2*); maximum individual feature efficiency (*F3*); collective feature efficiency (*F4*). The complexity is low if at least one predictive attribute can separate the classes.

Linearity measures quantify whether the classes are linearly separated. They include the sum of the error distance by linear programming (*L1*); error rate of linear classifier (*L2*); non-linearity of a linear classifier (*L3*). To obtain the linear classifier, a linear Support Vector Machine (SVM) is often used.

Neighborhood measures analyze the neighborhoods of individual examples and try to capture class overlap and the shape of the decision boundary. They include fractions of Borderline Points (*N1*); ratio of intra/extra class nearest neighbor distance (*N2*); error rate of the nearest neighbor classifier (*N3*); non-Linearity of the nearest neighbor classifier (N4); fraction of hyperspheres covering data (*T1*); local set average cardinality (*LSC*). All of them use a distance matrix between all pairs of points in the dataset to define the instances' neighborhoods according to their classes.

The network measures transform a dataset into a graph and extract structural and statistical information from the graph. In

Table 9
Complexity measures and their characteristics. They are robust measures.

Acronym	Task	Extract	Argument	Domain	Hyperp.	Range	Card.	Determ.
clsCoef	Classif.	Indirect	∗P+T	Num.	Yes	[0, 1]	1	Yes
graphDensity	Classif.	Indirect	∗P+T	Num.	Yes	[0, 1]	1	Yes
F1	Classif.	Direct	1P+T	Both	No	[0, 1]	*d*	Yes
F1v	Classif.	Indirect	∗P+T	Both	No	[0, 1]	*q*	Yes
F2	Classif.	Direct	1P+T	Num.	No	[0, 1]	*q*	Yes
F3	Classif.	Direct	1P+T	Num.	No	[0, 1]	*q*	Yes
F4	Classif.	Direct	∗P+T	Num.	No	[0, 1]	*q*	Yes
Hubs	Classif.	Indirect	∗P+T	Num.	Yes	[0, 1]	*q*	Yes
LSC	Classif.	Direct	∗P+T	Num.	No	$[0, 1 - \frac{1}{n}]$	1	Yes
L1	Classif.	Indirect	∗P+T	Num.	No	[0, 1]	*q*	Yes
L2	Classif.	Indirect	∗P+T	Num.	No	[0, 1]	*q*	Yes
L3	Classif.	Indirect	∗P+T	Num.	No	[0, 1]	*q*	No
N1	Classif.	Indirect	∗P+T	Num.	No	[0, 1]	1	Yes
N2	Classif.	Direct	∗P+T	Both	No	[0, 1]	*n*	Yes
N3	Classif.	Direct	∗P+T	Both	No	[0, 1]	*q*	Yes
N4	Classif.	Direct	∗P+T	Both	No	[0, 1]	*q*	Yes
T1	Classif.	Direct	∗P+T	Num.	No	[0, 1]	1	Yes
T2	Any	Direct	∗P	Both	No	$[0, \overline{n}]$	1	Yes
T3	Any	Indirect	∗P	Num.	No	$[0, \overline{n}]$	1	Yes
T4	Any	Indirect	∗P	Num.	No	[0, 1]	1	Yes

this new representation, each example from the dataset corresponds to a node, whilst undirected edges connect pairs of examples and are weighted by the distances between them. These measures include the average density of the network (*graphDensity*) and Hub score (*hubs*). Other complex network measures are presented by Morais and Prati [86], however they are not detailed and we did not find other works using them.

Finally, the dimensionality measures evaluate data sparsity according to the number of instances relative to the predictive attributes of the dataset. The measures include the average number of points per dimension (*T2*); the average number of points per PCA dimension (*T3*); the ratio of the PCA dimension to the original dimension (*T4*). While *T2* is the *instToAttr* meta-features, the *T3* and *T4* differ from *T2* by using a transformed dataset instead of the original.

These complexity measures look at different complexity aspects in a dataset. Thus, they can be related to other groups of measures presented in this study. A variation of them to characterize the classes individually instead of the whole dataset can be found in Barella et al. [94]. They are appropriate to represent the complexity of imbalanced datasets. These complexity measures are free of hyperparameters and some of them can be summarized when their formula is slightly modified to not use a specific summarization procedure (e.g., *F1* which uses the maximum value by default). Their extraction usually has a high computational cost, which can be a restrictive factor for their use in MtL.

4.7.3. Miscellaneous

In this section, we included other characterization measures found in our review, which did not fit in the previous groups and were used in a small number of MtL studies. These measures are summarized in Table 10.

Data distribution measures assess how the data is distributed in the predictive attribute space. One of these measures is the concentration coefficient, also known as *Goodman and Kruskal's* τ [76], which is applied to each pair of attributes (*attrConc*) and to each attribute and the class (*classConc*). In the former $d(d-1)$, values are obtained as it is not symmetric, whereas in the latter, d values are obtained, given that each attribute is associated with the class. Semantically, they represent the association strength between the attributes in each pair of attributes and between each predictive attribute and the target attribute.

Other related measures are the proportion of principal components that explain a specific (e.g., 95%) variance of the dataset (*propPCA*) and the *sparsity*, which extracts the degree of discreetness in each attribute. The former is another measure for capturing the redundancy of predictive attributes, whereas the latter indicates the variance in the values of the attributes.

Case base measures compare the instances with each other to identify properties that might make the learning process more difficult [95]. Most of them are originally proposed as logical measures, however, instead of only capturing the occurrence (or not) of each property, we propose small changes to quantify each occurrence. The *consistencyRatio* quantifies the proportion of repeated instances with different targets, where zero is an ideal value. The *uniquenessRatio* is a generalization of *consistencyRatio*, as it uses only the predictive attributes. To measure how dissimilar the instances are in their attribute space, *incoherenceRatio* computes the proportion of instances that do not overlap with any other instances in a predefined number of attributes. Values close to 1 are preferred in a dataset as it shows that the instances are scattered through the input space.

The concept-based measures characterize the sparsity and the irregularity of the input–output distribution [96]. An irregular distribution is observed when neighboring instances have distinct target values [11]. The weighted distance (*wgDist*) captures how dense or sparse the distribution of the instances is [97]. It could be defined as a distance-based measure. *Cohesiveness* measures the density of the example distribution [70]. Another measure of this subgroup, the concept variation [96] is defined by the cohesiveness average of all possible instances in the input space, therefore unfeasible. Its version using the existing instances is captured by the summarization function *mean*.

Distance-based measures [44] are obtained computing the distance between all pairs of instances (*distInst*) and their correlations combined with the distances (*distCorrInst*). They indicate how close and related pairs of instances are, which may influence the decision boundaries produced by learning algorithms. Finally, the center of gravity (*gravity*) computes the dispersion among the groups of instances according to their class label. In this case, the groups are defined by the classes.

Structural information works well in identifying similar datasets [98], by characterizing binary item sets to capture the distribution of values of both single attributes (*oneItemset*) and pairs of attributes (*twoItemset*) [99]. They capture different and complementary aspects of the dataset. *oneItemset* captures information of each individual's attributes, whereas, *twoItemset* captures possible correlations concerning pairs of attributes. Association rules can also be applied to the transformed dataset to characterize other relations between attributes [11,100].

Table 10
Other miscellaneous measures and their characteristics. They are robust measures.

Acronym	Task	Extract	Argument	Domain	Hyperp.	Range	Card.	Determ.
Data distribution measures								
attrConc	Any	Direct	2P	Categ.	No	[0, 1]	$\overline{d^2}$	Yes
classConc	Classif.	Direct	1P+T	Categ.	No	[0, 1]	d	Yes
propPCA	Any	Indirect	∗P	Num.	Yes	[0, 1]	1	Yes
sparsity	Any	Direct	1P	Both	No	[0, 1]	d	Yes
Case base measures								
consistencyRatio	Supervised	Direct	∗P+T	Both	No	[0, 1]	1	Yes
incoherenceRatio	Any	Direct	∗P	Both	Yes	[0, 1]	1	Yes
uniquenessRatio	Any	Direct	∗P	Both	No	[0, 1]	1	Yes
Concept based measures (distance based)								
cohesiveness	Classif.	Direct	∗P+T	Both	Yes	$[0, \overline{n}]$	n	Yes
distInst	Any	Direct	∗P	Both	$[0, \infty]$	$\overline{n^2}$	Yes	
distCorrInst	Any	Direct	∗P	Num.	[0, 1]	$\overline{n^2}$	Yes	
gravity	Classif.	Indirect	∗P+T	Both	$[0, \infty]$	1	Yes	
wgDist	Any	Direct	∗P	Both	Yes	$[0, \infty]$	n	Yes
Structural information								
oneItemset	Any	Indirect	∗P	Both	No	[0, 1]	d	Yes
twoItemset	Any	Indirect	∗P	Both	No	[0, 1]	$\overline{d^2}$	Yes
Time based measures								
infotheoTime	Any	Indirect	∗P	Categ.	No	$[0, \infty]$	1	No
landTime	Supervised	Indirect	∗P+T	Both	No	$[0, \infty]$	$\overline{7}$	No
modelTime	Supervised	Indirect	∗P+T	Both	No	$[0, \infty]$	1	No
statTime	Any	Indirect	∗P	Num.	No	$[0, \infty]$	1	No

Time-based measures comprise the elapsed time to compute the previous groups of measures [19], such as statistical, information-theoretic, model-based and landmarking. In this case, the same hardware should be used to compute the meta-features from different datasets, which can be very restrictive. Another option is to use the number of float point operations, but it is not always possible.

5. Discussion

This section discusses the different aspects of the characterization process, most of them strictly related to the taxonomy proposed in Section 3. Frequently ignored details, the unspoken decisions taken by researchers, are reviewed, along with the enumeration of gaps that demand further analysis whether theoretical, empirical or both.

5.1. Input domain

The input domain defines the data type supported by a meta-feature. As an example, statistical meta-features support only numerical data while information-theoretic meta-features support only categorical data. The alternatives adopted to handle non-supported data types, such as transformations, have rarely been reported in the literature, as observed in Smith et al. [80] and Ali and Smith [43] and Reif et al. [20] and Garcia et al. [49]. The impact on the characterization is unknown, mainly for the transformations that increase the number of features or data sparsity.

Fig. 1 summarizes the options adopted in the literature to deal with the data type, which consist of ignoring [59] or transforming the data [17]. By ignoring the attributes, two problems are faced: (i) if a dataset contains only attributes with the ignored data type, all respective measures will have missing values; (ii) in an MtL context, the algorithms/techniques recommended may support the ignored data. In favor of this choice, it can be argued that using only the meta-features that are able to characterize such data is a natural choice as they can properly represent the data [75]. Besides, their inability to process some types of data may be aligned with the limitations of some algorithms, therefore representing useful information. Alternatively, the datasets can be segmented by type (only numerical, only categorical and mixed) where only the suitable measures for each group are used [13,95].

By transforming the attributes, the meta-features can support any data types using a *binarization* or *discretization* approaches. It leads to new decisions since there are different alternatives used to transform the data, including the possibility of combining them together.

The most common transformation of categorical attributes into numerical ones is the *binarization* [101]. Its use to transform categorical attributes with a high number of distinct values is not recommended as it generates a large number of new attributes. Alternatively, each category can be mapped to an integer and then represented in a binary hash, where new attributes are used to represent the bit values of the represented information [102]. The unintended relationships among the new attributes can be a deficiency of this approach, considering the meaninglessness of these relations.

Similarly, some meta-features support only categorical attributes, and the transformation from numeric to categorical attributes may be necessary. For such, *discretization* techniques can be used. These techniques distribute numeric values in distinct intervals, which correspond to the new categories [101]. As a result, order relations in the original values and variations within the same interval are lost. In an unsupervised approach, the intervals can be defined using *equal-width* or *equal-frequency*, where they have the same interval width or the number of values, respectively. Other techniques such as clustering, correlation analysis and decision tree analysis can also be used for value discretization [103,104]. The last two, which are supervised approaches, use the target attribute to define the categories.

5.2. Missing values

The presence of missing values in the original datasets also requires attention, considering that many meta-features do not support the defective records. The alternatives to address this issue are: (i) imputation of values provided by a preprocessing

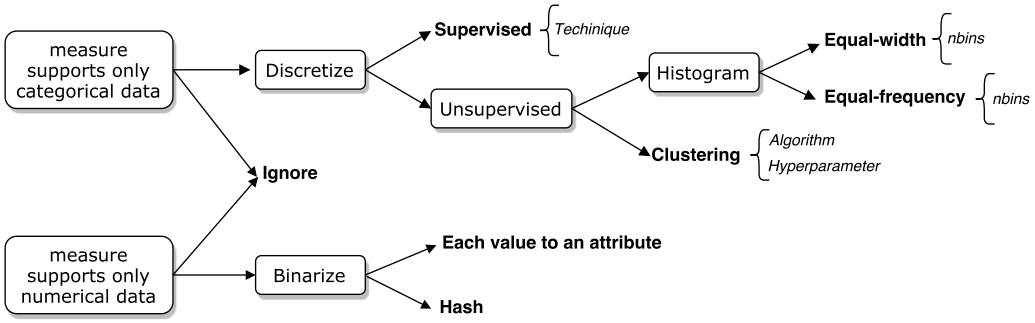

Fig. 1. Options to handle the input data type that are not supported by the meta-features.

step, (ii) removal of attributes and/or records with missing values and (iii) ignoring them during the computation of the measure.

Despite the fact that most datasets available in online public repositories are already preprocessed and do not present any problems, there are no guarantees in this regard. When faced with a dataset with missing values, the most straightforward decision is to remove the defective instances. When the presence of missing values concentrates in a specific feature, and occurs for several instances, another possibility is to remove only the respective attribute instead of the instances. However, when the dataset is small, the alternative to keep information is the imputation of values by a preprocessing technique [101]. However, this alternative may introduce noise into the dataset.

Preprocessing modifies the original data, which can also affect the MtL pipeline. A different alternative would be to adapt the characterization measures or use algorithms that intrinsically support missing values. In that case, the presence of missing values in the datasets is solved internally by each characterization measure. Another solution is to remove the missing data only during the computation of each measure. For measures that use a single predictive attribute (1P), this is a very simple process, with possibly no impact in the characterization process.

Missing values can occur in many real-world datasets and how they are dealt with affect the values of the meta-features extracted from the dataset. However, we did not find any mention to this matter in the MtL literature.

5.3. Hyperparameter values

Some techniques used for meta-feature extraction require the definition of hyperparameter values. Most MtL studies do not discuss this issue, probably because default values are commonly used.

Tables 4, 6, 7 and 10 identify the measures that require the definition of hyperparameter values. Some statistical measures have specific hyperparameter values. All model-based and landmarking meta-features, on the other hand, have hyperparameter values that affect the whole group. For the model-based one, different DT algorithms can be used to induce the model and each algorithm requires additional configurations. For the landmarking, the validation strategy, the evaluation measure and also the algorithms hyperparameters can be modified. In these cases, the same set of configurations is usually adopted for all measures of the group, but not necessarily by more than one author.

Other decisions concerning the use of meta-features and summarization functions can also be seen as hyperparameters. For instance, how to handle the unsupported data type, as described in Section 5.1, and the transformation by class [17] proposed to explore the target information, affect the statistical and information-theoretic groups and can also be defined as hyperparameters. Additionally, the *histogram* summarization function also has a hyperparameter that defines the number of bins to represent the measures.

In summary, the effects of such choices in the data characterization process are unknown. Alternatives, such as tuning the different parameters of the measures, using distinct instances of the same measure and evaluating the amount of information captured by them, have not been explored.

5.4. Range of the measures

The data range has been frequently ignored in MtL studies, which suggests that meta-features have been used directly without transformation or it has not been properly reported. Although meta-features have a different range of values, they are used together in a meta-base. Considering that some algorithms are influenced by attributes with different ranges [104,105], the meta-data can be transformed by min–max scaling or z-score normalization., as illustrated by the vertical axis in Fig. 2.

The transformation can occur at three distinct moments: (i) in the *dataset*, before any computation; (ii) in the result of the characterization measure, before the summarization function; and (iii) in the meta-base, after computing the meta-feature. The application of these transformations in different moments possibly affect the obtained results, regardless of how the change takes place.

The dataset transformation is an alternative for the measures whose scale is determined by the values present in the dataset (range is *inherit*). Changes in the original data range will reflect on the outcome of these meta-features. The second alternative transforms the result of the characterization measures. It is more suitable for multi-valued measures. Both alternatives are not recommended for meta-features using summarization functions on a particular scale, such as *kurtosis* and *skewness*. Finally, the most conventional approach is to transform the meta-features result, which requires the characterization of all datasets beforehand.

Some rescaled meta-features are used along with (or instead of) their original version. The proportion of numeric and categorical attributes [31,76], the proportion of attributes with outliers and normal distribution [77,106] and the normalized entropy [17], are some examples found in the literature. However, only a few measures have their rescaled version named. The theoretical maximum and minimum values from the measures with a non-infinity range can be modified with the min–max scaling. The transformation of meta-features for some dataset characteristic (e.g. the number of instances) using absolute or relative values can be a better alternative.

In summary, the lack of information about the procedures adopted concerning the meta-data transformation is observed in many MtL studies. The different alternatives to transform the meta-features can suit some meta-features better than others. This is a very important research question that has not been satisfactorily addressed in the MtL literature.

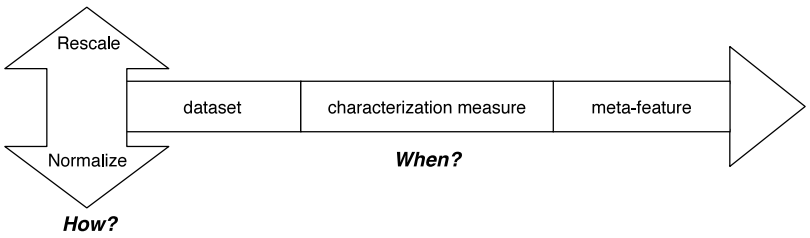

Fig. 2. Options to transform the range of the measures.

5.5. Summarization functions

In most MtL studies, summarization functions are combined with meta-features, either implicitly or explicitly. Implicitly when they are defined as part of the meta-feature formalization [17,18,72,74], where the average result is the most natural solution used. Explicitly when studies show the effectiveness of using other options to summarize measures [59–61,71], as reported in Section 4.1.

Some combinations of meta-features and summarization functions have a semantic meaning. For instance, the standard deviation (*sd*) applied to the frequencies of the classes (*freqClass*) shows how uniform the class distribution is, which may also indicate that the classes are unbalanced. Other combinations are meaningless, such as the use of the cardinality of the measure (*count*) to summarize the joint entropy (*jointEnt*), as the measure has a fixed cardinality. There are also some possible problematic combinations, such as the use of *histograms* to summarize meta-features with low cardinality and/or with the range that is defined according to a dataset characteristic. In this case, the histogram bins can be sparse and represent different scales of values for each dataset.

The use of many functions to summarize a measure proportionally increases the number of meta-features obtained. As many measures are multi-valued, hundreds of results can be easily obtained when combined with multiple summarization functions. The relatively low number of meta-examples usually observed in MtL experiments, together with the high number of meta-features, could generate meaningless models, affected by the curse of dimensionality [102]. The use of a feature-selection algorithm can be an alternative to deal with this problem [22,71].

Even though summarization functions are not strictly related to reproducibility issues, they are relevant to reproducibility because different choices can be made in a characterization process. The empirical analysis of summarization functions and the exploration of new ways to summarize meta-features could be the subject of future research.

5.6. Exceptions

As discussed previously, some measures can be incorrectly computed for some datasets. Their use requires specific conditions that cannot always be guaranteed. Operations such as division by zero and logarithm of negative values are the main causes of exceptions. Alternatives to deal with problematic measures are: (i) assuming it results in a missing value; (ii) using a default value; (iii) if the measure is multi-valued, ignore it.

The first option results in a meta-base with missing values, which eventually will be filled using some preprocessing technique [104] or removed from the meta-base. The other two alternatives fix the problem of having a missing value during the computation of the meta-feature. The use of a default value to represent exceptional cases can be positive when it properly characterizes the measure and the phenomenon that generates the exception. However it can introduce noise in the predictive meta-data. This does not occur when the defective results can be removed before the summarization. As a drawback, this alternative is valid only for the multi-valued measures. Furthermore, to discard few values for measures with high cardinality, the final result will not change drastically, but for the measures with low cardinality, this approach may lead to distortions in results.

Summarization functions can also generate exceptions. This is the case of *sd*, *kurtosis* and *skewness*. The *sd* cannot be applied to single values while the *kurtosis* and *skewness* cannot be applied to constant vectors. The alternatives *i* and *ii* can also be adopted for them. The value 0 is the default value suggested to fill the problematic cases, which represents no deviations for *sd* and constant values for *kurtosis* and *skewness*.

In summary, the use of these measures and summarization function does not imply that they will generate exceptions during the extraction of meta-features. However, there is an absence of information about the occurrence or lack of occurrence in empirical studies in MtL.

5.7. Dimensionality

The ratio between the number of meta-features and the number of meta-examples in MtL experiments is usually higher than in conventional ML experiments. Furthermore, it is well known that the most suitable meta-features vary for different MtL tasks [13]. Thus, some studies have investigated the use of feature selection techniques [71,106] and the transformation of the meta-features' space [11,13] to reduce the dimensionality of meta-bases, as well as to increase the predictive performance of meta-models [57].

Meta-feature selection is just an instance of feature selection [22]. Among the different approaches for meta-feature selection, the wrapper appeared more often in our literature review [3,18,20,49,57,60] than the use of a filter [16,41,71].

In noz et al. [11], the authors followed a new approach for meta-feature selection. They investigated the behavior of several meta-features in 12 classification challenges. By modifying a dataset to increase/decrease each investigated problem, the variance of the meta-features is statistically assessed, revealing those that better characterize each variation. After the repetition of the process using different datasets, the most relevant features for each challenge are obtained.

Another work for the meta-feature dimensionality reduction used Principal Component Analysis (PCA) [107] to obtain latent meta-features [13]. After computing the principal components, the most relevant (according to the cumulative total variance) are selected. Later, the authors used a filter based in a correlation with the target to select a subset of the latent meta-features.

As PCA does not take into account the target variable to transform the data, noz et al. [11] used optimization to transform a set of previously selected meta-features into a 2-D space. For such, the authors used the performance of several learning algorithms. Named instance space, it enables the visualization of the set of datasets used in an MtL study.

Some works have compared groups of meta-features [19,20,95,108], with different findings. For instance, landmarkings and model-based meta-features were the most important characterization measures in Reif et al. [20] and Filchenkov and Pendryak [18], respectively. In contrast, the feature selection wrapper did not improve the predictive performance of the meta-models in Garcia et al. [49].

To estimate the importance of a meta-feature, Filchenkov and Pendryak [18] uses a significance measure that associates the predictive performance of a model induced using each meta-feature alone. This process is repeated several times and the average performance obtained for each meta-feature is the meta-feature significance value. Pimentel and de Carvalho [44] define the meta-feature importance as the number of times it is selected when the Random Forest algorithm is applied to the meta-base. In Salama et al. [106] and Peng et al. [16], the authors use the correlation between them and the meta-target to select the meta-features [16,106].

The decision of whether to use the reduction and/or transformation is an important issue in the design of the pipeline for an MtL experiment. When used, a detailed specification of the procedures adopted is essential for the replication of the experiments. Moreover, while meta-feature selection may improve the interpretability of the meta-models, the same is not the case when a transformation is used.

5.8. Benchmarks

Benchmarks are the datasets used to generate meta-data. As in any learning task, the ideal situation would be to use a large number of datasets, increasing the chances of inducing a reliable and robust meta-model. To reduce the presence of bias, datasets from several data repositories, such as UCI [109], Keel [110] and standard repositories hosting services such as mldata.org[1] [111] and OpenML[2] [112], can be used.

Other strategies to increase the number of datasets are using artificial data or changing the distribution of the classes to increase the number of examples in the meta-data [113]. There are also more complex strategies, such as using active learning for instance selection, of datasetoids, a data manipulation method to obtain new datasets from existing datasets and of an interpretable projection approach [11,114,115].

6. Tools

Characterization tools have an important role in the development of research in MtL. Besides simplifying an essential step of the work, their use corroborates the reproducibility of MtL experiments. However, the approach used in the development of the tool can generate two different perspectives: (i) a black box tool with abstracted choices, which promotes reproducibility, but only for the users that use the same tool or, (ii) a white box tool that exposes all the options to the user promoting reproducibility even with different tools, but forcing them to make the explicit decisions about the parameter values.

The Data Characterization Tool (DCT)[3] [69] is the most referenced characterization tool in the MtL literature [15,20,95,116], to cite a few. The DCT contains a representative subset of meta-features from simple, statistical and information-theoretic groups.

Matlab Statistics Toolbox [117] have also been used to characterize statistical measures [10,35,43]. Weka [118], RapidMiner [119] and other general data mining tools can be employed to compute landmarking meta-features [108,120].

OpenML [112] is a publicly available online tool that includes, among other functionalities, several meta-feature extraction techniques for dataset characterization. Many of the reported measures are available in the platform, which is also a benchmarking repository that contains the characterization of several datasets. OpenML uses an extension of the Fantail library [84], also available on GitHub.[4] A drawback may be that the characterization process is performed automatically when a new dataset is submitted to the platform, which abstracts the users' choices. On the other hand, anyone can compute and upload their meta-features to OpenML through its API.[5]

The framework proposed by Pinto et al. [71] is available as an open GitHub project[6], but without implementing the meta-features, which could be an expensive task. Except for that, all the reviewed tools are black-box tools.

Another publicly available tool, Meta-Feature Extractor (MFE) tool[7] [121] has the implementation of most of the meta-features and summarization functions described in this paper. MFE package allows the user to compute a specific, a group of or all meta-features available. It is possible to define which summarization functions should be computed and, optionally, to obtain all computed values for a given set of measures, without summarizing the results. A white box tool allows its users to set the hyperparameter values of those meta-feature extraction techniques that have hyperparameters.

Other characterization tools publicly available are: the Extended Complexity Library (ECoL)[8] [93] covering the complexity measures; and, the BYU-DML,[9] a Python tool that includes general, statistical, information-theoretic, landmarking and model-based meta-features. In parallel, many authors have used their own implementation of meta-features [18,20,49,60], without reporting and making publicly available their implementation. When the implementation codes are not available, it is difficult to reproduce the reported results.

7. Conclusion

Recommending techniques by using MtL is an effective alternative to deal with the selection of the most suitable techniques among a large number of possibilities. In this context, meta-features and the characterization process play a crucial role to determine the success of the MtL approach. Despite their important role in MtL experiments, most articles do not describe the methodology used for the extraction of meta-features and different works have used different approaches; Thus, by discussing topics that have been frequently ignored in the MtL literature and suggesting possible alternatives to address them, this paper reviewed the main meta-features for classification tasks and important issues related to the dataset characterization process in MtL experiments. Another contribution from this study is the proposal of a new taxonomy for meta-features and a list of dataset characterization tools.

The new taxonomy organized and formalized the current meta-features and their usefulness across different types of task, domain, range, and several other characteristics that can impact

1 http://dataverse.org/

2 http://www.openml.org/

3 https://github.com/openml/metafeatures/dct

4 https://github.com/quansun/fantail-ml, https://github.com/openml/EvaluationEngine

5 https://www.openml.org/api_docs#!/data/post_data_qualities

6 https://github.com/fhpinto/systematic-metafeatures

7 Available in Python (https://pypi.org/project/pymfe/) and R (https://cran.r-project.org/package=mfe) languages

8 https://github.com/lpfgarcia/ecol

9 https://github.com/byu-dml/metalearn

MtL tasks. Based on this review, the authors enumerated the main decisions a researcher faces when using meta-features. Moreover, a detailed discussion is provided on the cutting edge subgroup of meta-features, their predictive power and the use cases where these measures have been applied.

Future work will investigate meta-features for other types of tasks, such as regression and clustering; increase the interpretability of the meta-features; and explore empirical analysis showing how some choices related to the hyperparameters, cardinality and the summarization functions can affect dataset characterizations to best distinguish the performance of meta-models. A review of regression and clustering meta-features could improve the task representation and could also look at a different perspective and validate the taxonomy proposed. The exploration of interpretability of the meta-features and the empirical analysis over hyperparameters, cardinality and summarization function could improve the meta-model representation and performance.

Declaration of competing interest

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

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq (152098/2016-0) and FAPESP (2016/18615-0 and 2013/07375-0). The first, second and fifth authors would like to thank CeMEAI-FAPESP for the computational resources and Intel for the hardware and software server used in part of the experiments.

Appendix. Characterization measure formalization

A.1. Simple

attrToInst Ratio of the number of attributes per the number of instances [59], also known as dimensionality: $\frac{d}{n}$.

catToNum Ratio of the number of categorical attributes per the number of numeric attributes [73]: $\frac{nrCat_{\mathbf{X}}}{nrNum_{\mathbf{X}}}$.

classToAttr Ratio of the number of classes per the number of attributes [60]: $\frac{q}{d}$

freqClass Frequencies of the class values [69]: $\left[prop_{c_1}, \ldots, prop_{c_q}\right]$, such that

$$prop_{c_j} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i = c_j). \tag{A.1}$$

instToAttr Ratio of the number of instances per the number of attributes [72]: $\frac{n}{d}$.

instToClass Ratio of the number of instances per the number of classes [70]: $\frac{n}{q}$.

ntAttr Number of attributes [75]: d.

nrAttrMissing Number of attributes with missing values [73]:

$$\sum_{j=1}^{d} \mathbb{1}(\sum_{i=1}^{n} \mathbb{1}(\vec{x}_{ij} = \varnothing) > 0)$$

nrBin Number of binary attributes [75]: $\sum_{i=1}^{d} \mathbb{1}(\phi_{\vec{a}_i} = 2)$. It includes numerical and categorical attributes that contain only two distinct values.

nrCat Number of categorical attributes [68]: $d - nrNum_{\mathbf{X}}$.

nrClass Number of classes [75]: q.

nrInst Number of instances [75]: n.

nrInstMissing Number of instances with missing values [69]:

$$\sum_{i=1}^{n} \mathbb{1}(\sum_{j=1}^{d} \mathbb{1}(\vec{x}_{ij} = \varnothing) > 0)$$

nrMissing Number of missing values [69]:

$$\sum_{i=1}^{n}\sum_{j=1}^{d} \mathbb{1}(\vec{x}_{ij} = \varnothing)$$

nrNum Number of numeric attributes [68]: $\sum_{i=1}^{d} \mathbb{1}(\vec{a}_i \in \mathbb{R}^n)$.

numToCat Ratio of the number of numeric attributes per the number of categorical attributes [73]: $\frac{nrNum_{\mathbf{X}}}{nrCat_{\mathbf{X}}}$.

A.2. Statistical

canCor Canonical correlations between the predictive attributes and the class [122]: $[\rho_1, \ldots, \rho_z]$, such that $\rho_i = cor_{\vec{w}_x^{(i)}\mathbf{X}, \vec{w}_y^{(i)}\mathbf{Y}}$, where $\vec{w}_x^{(i)}$ and $\vec{w}_y^{(i)}$ maximizes ρ_i and are orthogonal to the $\vec{w}_x^{(i-1)}$ and $\vec{w}_y^{(i-1)}$, $\mathbf{Y}$ is the binarized version of $\vec{y}$ and $z \leq min\,[q, d]$ is the number of distinct $\vec{w}_x$ and $\vec{w}_y$ vectors found by using discriminant analysis. Frequently, the canonical correlation is reported in the literature as the eigenvalues of the canonical discriminant matrix, such that

$$\rho_i = \sqrt{\frac{\lambda_i}{1+\lambda_i}}. \tag{A.2}$$

cor Absolute attributes correlation [17]: $\left[\ |cor_{\vec{a}_1,\vec{a}_3}|, \ldots, |cor_{\vec{a}_{d-1},\vec{a}_d}|\ \right]$, such that $cor_{x,y}$ is obtained by using a correlation algorithm. The most common one used is the Pearson's Correlation coefficient, given by

$$cor_{x,y} = \frac{cov_{x,y}}{sd_x sd_y}, \text{ where} \tag{A.3}$$

$$cov_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}, \text{ and} \tag{A.4}$$

$$sd_x = \sqrt[2]{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{A.5}$$

cov Attributes covariance [17]: $\left[\ |cov_{\vec{a}_1,\vec{a}_2}|, \ldots, |cov_{\vec{a}_{d-1},\vec{a}_d}|\ \right]$, where $cov_{x,y}$ is given by Eq. (A.4).

nrDisc Number of discriminant functions [69]: $|canCor_{\mathcal{D}}|$.

eigenvalues Eigenvalues of the covariance matrix [43]: $\left[\lambda_1, \ldots, \lambda_d\right]$, such that $S\vec{v} = \lambda_i\vec{v}$ for some $\vec{v} \neq 0$, where $S_{d\times d}$ is the covariance matrix of $\mathbf{X}$.

gMean Geometric mean of attributes [35]: $\left[gMean_{\vec{a}_1}, \ldots, gMean_{\vec{a}_d}\right]$, such that $gMean_x = \left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$.

hMean Harmonic mean of attributes [35]: $\left[hMean_{\vec{a}_1}, \ldots, hMean_{\vec{a}_d}\right]$, such that

$$hMean_x = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}.$$

iqRange Interquartile range of attributes [35]: $\left[iqRange_{\vec{a}_1}, \ldots, iqRange_{\vec{a}_d}\right]$, such that $iqRange_x = Q3_x - Q1_x$, where $Q1_x$ and $Q3_x$ represent the first and third quartile values of x, respectively.

kurtosis Kurtosis of attributes [75]: $\left[kurt_{\vec{a}_1}, \ldots, kurt_{\vec{a}_d}\right]$, such that

$$kurt_x = \frac{m_4}{sd_x^4} - 3,$$

where m_j represents a statistical moment, given by

$$m_j = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^j. \tag{A.6}$$

mad Median absolute deviation of attributes [43]: $\left[mad_{\vec{a}_1}, \ldots, mad_{\vec{a}_d}\right]$, such that $mad_x = median\,[\,|x_1 - median_x|, \ldots, |x_n - median_x|\,]$, where

$$median_x = \begin{cases} \frac{1}{2}(x_{(r)} + x_{(r+1)}) & \text{if } |x| \text{ is even } (|x| = 2r) \\ x_{(r+1)} & \text{otherwise } (|x| = 2r+1) \end{cases} \tag{A.7}$$

max Maximum value of attributes [68]: $\left[max\ \vec{a}_1, \ldots, max\ \vec{a}_d\right]$.

mean Mean value of attributes [68]: $\left[\overline{\vec{a}_1}, \ldots, \overline{\vec{a}_d}\right]$.

median Median value of attributes [68]:

$\left[median_{\vec{a}_1}, \ldots, median_{\vec{a}_d}\right]$, where $median_x$ is given by Eq. (A.7).

min Minimum value of attributes [68]: $\left[min\ \vec{a}_1, \ldots, min\ \vec{a}_d\right]$.

nrCorAttr Number of attributes pairs with high correlation [106]:

$$\frac{2}{d(d-1)}\sum_{i=1}^{d-1}\sum_{j=i+1}^{d}\mathbb{1}(|cor_{\vec{a}_i,\vec{a}_j}| \geq \tau),$$

where τ is a threshold value between 0 and 1, usually $\tau = 0.5$. This is the normalized version adapted by the authors.

nrNorm Number of attributes with normal distribution [63]:

$\sum_{i=1}^{d}\mathbb{1}(isNormal_{\vec{a}_i})$. To check if an attribute has or does not have a normal distribution, the W-Test for normality [123] can be applied, for instance.

nrOutliers Number of attributes with outliers values [95]:

$\sum_{i=1}^{d}\mathbb{1}(hasOutlier_{\vec{a}_j})$. To test if an attribute has or does not have outliers, the Tukey's boxplot algorithm [124] can be used, for instance.

range Range of Attributes [35]:

$\left[(max\ \vec{a}_1 - min\ \vec{a}_1), \ldots, (max\ \vec{a}_d - min\ \vec{a}_d)\right]$.

sd Standard deviation of the attributes [68]: $\left[sd_{\vec{a}_1}, \ldots, var_{\vec{a}_d}\right]$, such that sd_x is given by Eq. (A.5).

sdRatio Statistic test for homogeneity of covariances [75]:

$$exp(M/d\sum_{i=1}^{q}(n_{c_i} - 1)), \text{ where } M = \gamma\sum_{i=1}^{q}(n_{c_i} - 1)log|S_i^{-1}S|;$$

$$\gamma = 1 - \frac{2d^2 + 3d - 1}{6(d+1)(q-1)}\sum_{i=1}^{q}\frac{1}{n_{c_i} - 1} - \frac{1}{n-q};$$

$$S = \frac{1}{n-q}\sum_{i=1}^{q}(n_{c_i} - 1)S_i$$

such that, n_{c_i} is the number of instances related to the class c_i, S is called pooled covariance matrix and S_i is the sample covariance matrix of the instances for the ith class.

skewness Skewness of attributes [75]: $\left[skewness_{\vec{a}_1}, \ldots, skewness_{\vec{a}_d}\right]$, such that

$$skewness_x = \frac{m_3}{sd_x^3},$$

where sd_x and m_3 are given by Eqs. (A.5) and (A.6), respectively.

tMean Trimmed mean of attributes [68]: $\left[tMean_{\vec{a}_1}, \ldots, tMean_{\vec{a}_d}\right]$, such that

$$tMean_x = \frac{x_{(i+1)} + x_{(i+2)} + \cdots + x_{(n-i-2)} + x_{(n-i-1)}}{n - 2i},$$

where $i = \lceil n\alpha \rceil$ and α is a hyperparameter, such that $0 < \alpha < 0.5$. The suggested value is $\alpha = 0.2$.

var Attributes variance [17]: $\left[var_{\vec{a}_1}, \ldots, var_{\vec{a}_d}\right]$, such that

$$var_x = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}.$$

wLambda Wilks Lambda [69]:

$$\prod_{i=1}^{z}\frac{1}{1+\lambda_i},$$

where $z = nrDisc_{\mathcal{D}}$ and λ_i is defined in Eq. (A.2).

A.3. Information-theoretic

Let H_x be the entropy of a given attribute, such that

$$H_x = -\sum_{i=1}^{\phi_x} P(x = \varphi_i^x)\log_2 P(x = \varphi_i^x),$$

and let $H_{x,y}$ be the joint entropy of a predictive attribute x and the class y, such that

$$H_{x,y} = \sum_{i=1}^{\phi_x}\sum_{j=1}^{\phi_y}\pi_{ij}\log_2\pi_{ij},$$

where $\pi_{ij} = P(x = \varphi_i^x, y = \varphi_j^y)$. The mutual information shared between them is given by $MI_{x,y} = H_x + H_y - H_{x,y}$. Mainly from these concepts, the information-theoretic measures are computed as following:

attrEnt Attributes entropy [75]: $\left[H_{\vec{a}_1}, \ldots, H_{\vec{a}_d}\right]$.

classEnt Class entropy [75]: $H_{\vec{y}}$

eqNumAttr Equivalent number of attributes [75]:

$$\frac{H_{\vec{y}}}{\frac{1}{d}\sum_{i=1}^{d}MI_{\vec{a}_i,\vec{y}}}$$

jointEnt Joint Entropy of attributes and classes [75]: $\left[H_{\vec{a}_1,\vec{y}}, \ldots, H_{\vec{a}_d,\vec{y}}\right]$.

mutInf Mutual information of attributes and classes [75]: $\left[MI_{\vec{a}_1,\vec{y}}, \ldots, MI_{\vec{a}_d,\vec{y}}\right]$.

nsRatio Noisiness of attributes [75] : $\frac{\frac{1}{d}\sum_{j=1}^{d} H_{\vec{a}_j} - \frac{1}{d}\sum_{j=1}^{d} MI_{\vec{a}_j,\vec{y}}}{\frac{1}{d}\sum_{j=1}^{d} MI_{\vec{a}_j,\vec{y}}}$.

A.4. Model-based

For DT-model meta-features, let ψ be the set of leaves, η be the set of nodes, such that $\psi \cap \eta = \emptyset$ and $\Gamma = \psi \cup \eta$ are the whole structure of the tree that represents the DT learning model. In addition, consider the following tree properties:

$attr_{\eta_i}$ Predictive attribute used in the node η_i.

$class_{\psi_i}$ Class predicted by the leaf ψ_i.

$inst_{\Gamma_i}$ Number of training instances used to define the tree element Γ_i.

$level_{\Gamma_i}$ Level of the tree element Γ_i. In other words, it is the number of nodes in the tree hierarchy necessary to reach the root of the tree, such that $level_{\Gamma_i} = 0$ iff $\Gamma_i = root_\Gamma$.

$prob_{\psi_i}$ Probability of reaching the leaf ψ_i from the root in a random walk through the tree hierarchy, such that $prob_{\psi_i} = \frac{1}{2^{level_{\psi_i}}}$.

$root_\Gamma$ Root node of a tree, such that $root_\Gamma \in \eta$.

The DT-model meta-features are the following:

leaves Number of leaves [74]: $|\psi|$.

leavesBranch Size of branches [74]: $\left[level_{\psi_1}, \ldots, level_{\psi_z}\right]$, where $z = |\psi|$.

leavesCorrob Leaves corroboration [62]: $\left[\frac{inst_{\psi_1}}{n}, \ldots, \frac{inst_{\psi_z}}{n}\right]$, where $z = |\psi|$.

leavesHomo Homogeneity [62]: $\left[\frac{z}{shape_{\psi_1}}, \ldots, \frac{z}{shape_{\psi_z}}\right]$, where $z = |\psi|$.

leavesPerClass Leaves per class [18]: $\left[lpc_{c_1}, \ldots, lpc_{c_q}\right]$, such that

$$lpc_{c_j} = \frac{1}{|\psi|}\sum_{i=1}^{|\psi|} \mathbb{1}(class_{\psi_i} = c_j)$$

nodes Number of nodes [74]: $|\eta|$.

nodesPerAttr Ratio of the number of nodes per the number of attributes [62]: $\frac{|\eta|}{d}$.

nodesPerInst Ratio of the number of nodes per the number of instances [62]: $\frac{|\eta|}{n}$.

nodesPerLevel Number of nodes per level [74]: $\left[npl_1, \ldots, npl_{level_w}\right]$, such that

$$w = \arg\max_{\eta_i \in \eta} level_{\eta_i}, \text{ and}$$

$$npl_j = \sum_{i=1}^{|\eta|} \mathbb{1}(level_{\eta_i} = j).$$

nodesRepeated Repeated nodes [62]: $[nrp_1, \ldots, nrp_d]\ \forall\ nrp_j > 0$, such that

$$nrp_j = \sum_{i=1}^{|\eta|} \mathbb{1}(attr_{\eta_i} = j).$$

treeDepth Tree depth [74]: $\left[level_{\Gamma_1}, \ldots, level_{\Gamma_w}\right]$, where $w = |\Gamma|$.

treeImbalance Tree imbalance [62]: $\left[imb_{\psi_1}, \ldots, imb_{\psi_w}\right]$, such that $w = |\psi|$ and $imb_{\psi_j} = -z_{\psi_j}(log_2\ z_{\psi_j})$, where $z_{\psi_j} = prob_{\psi_j}\sum_{i=1}^{w} \mathbb{1}(prob_{\psi_i} = prob_{\psi_j})$.

treeShape Tree shape [62]: $\left[shape_{\psi_1}, \ldots, shape_{\psi_w}\right]$, such that $shape_{\psi_j} = -prob_{\psi_j}(log_2\ prob_{\psi_j})$ and $w = |\psi|$.

varImportance Variable importance [62]: $\left[imp_{\vec{a}_1,\vec{y}}, \ldots, imp_{\vec{a}_d,\vec{y}}\right]$, where $imp_{\vec{a}_j,\vec{y}}$ describes the homogeneity of the class $\vec{y}$ produced by some split of a given attribute $\vec{a}_j$. Each DT learning algorithm uses a specific procedure to define the importance of the variables.

A.5. Landmarking

Let $\mathcal{A}$, θ_A and ξ be, respectively, a learning algorithm, a learning model and an evaluation measure. All landmarking meta-features are computed in the same way, such as the model is induced using the learning algorithm and a train data:

$$\mathcal{A}(\mathbf{X}_{train}, \vec{y}_{train}) \rightarrow \theta_A$$

and the prediction of the learning model for a test data is evaluated using the given measure, such as

$$landmarking_A = \xi(\theta_A(\mathbf{X}_{test}), \vec{y}_{test}),$$

where *train* and *test* subset are defined for each fold.

The differences between the landmarking measures are given by the learning-algorithm family and the predictive attributes used to induce the model, as described below:

bestNode Decision Node: $\mathcal{A}_{\text{DT}}(\mathbf{X}_{train,battr}, \vec{y}_{train}) \rightarrow \theta_{\text{bestNode}}$, where $\mathbf{X}_{train,battr}$ is the content of the most informative attribute, which is defined using the *varImportance* result.

eliteNN Elite Nearest Neighbor: $\mathcal{A}_{\text{KNN}}(\mathbf{X}_{train,battrset}, \vec{y}_{train}, k = 1) \rightarrow \theta_{\text{eliteNN}}$, where $\mathbf{X}_{train,battrset}$ contains only the subset of the most informative attributes for the train data. They are defined using the *varImportance* result.

linearDiscr linear Discriminant: $\mathcal{A}_{\text{LD}}(\mathbf{X}_{train}, \vec{y}_{train}) \rightarrow \theta_{\text{linearDiscr}}$.

naiveBayes Naive Bayes: $\mathcal{A}_{\text{NB}}(\mathbf{X}_{train}, \vec{y}_{train}) \rightarrow \theta_{\text{naiveBayes}}$.

oneNN One Nearest Neighbor: $\mathcal{A}_{\text{KNN}}(\mathbf{X}_{train}, \vec{y}_{train}, k = 1) \rightarrow \theta_{\text{oneNN}}$.

randomNode Random node: $\mathcal{A}_{\text{DT}}(\mathbf{X}_{train,rattr}, \vec{y}_{train}) \rightarrow \theta_{\text{randomNode}}$, where $\mathbf{X}_{train,rattr}$ is the content of a random attribute.

worstNode Worst node: $\mathcal{A}_{\text{DT}}(\mathbf{X}_{train,wattr}, \vec{y}_{train}) \rightarrow \theta_{\text{worstNode}}$, where $\mathbf{X}_{train,wattr}$ is the content of the least informative attribute.

A.6. Others

The following subsections specify the non-traditional characterization measures, which include groups and standalone meta-features.

A.6.1. Clustering

The clustering measures use the result of a clustering algorithm to extract the data characteristics. The k partitions obtained from the use of a clustering algorithm are denoted by $C_i \subset \mathcal{D}$, such that $\vec{\bar{x}}_{C_i}$ denotes the center of cluster i. Without loss of generality, $dist_{x,y}$ represents a distance between two instances $\vec{x}_i \in \mathcal{D}$, $\vec{x}_j \in \mathcal{D}$, regardless of the type of attributes they have.

compactness Quantify the compactness of the partitions [46]: $[c_1, \ldots, c_k]$, such that

$$c_i = \sum_{\vec{x}_j \in C_i} dist_{\vec{x}_j, \vec{\bar{x}}_{C_i}}. \tag{A.8}$$

connectivity Amount of neighboring instances that are not in the same partition [46]:

$$\sum_{i=1}^{n} \mathbb{1}(\vec{x}_j \in C_i \wedge nn_{\vec{x}_i} \notin C_i),$$

where $nn_{\vec{x}_i}$ is the nearest neighbor of instance $\vec{x}_i$.

nrClusters Number of clusters [45]: $|C| = k$.

nre Normalized relative entropy [45]:

$$\frac{\sum_{j=1}^{k} P(C_j) \log \frac{P(C_j)}{1/k}}{2 \log k},$$

where $P(C_j) = \frac{|C_j|}{n}$ represents the probability of the uniform partition distribution.

purityRatio Ratio of the number of clusters with a given class [87]: $\left[\frac{s_1}{k}, \ldots, \frac{s_q}{k}\right]$, where

$$s_i = \sum_{j=1}^{k} \mathbb{1}((\vec{x}_l, y_i) \in C_j)$$

sizeDist Proportion of instances present in each cluster [87]: $\left[\frac{|C_1|}{n}, \ldots, \frac{|C_k|}{n}\right]$.

A.6.2. Complexity measures

The complexity measures are specified, well described and explained in Lorena et al. [93].

A.6.3. Miscellaneous

Data distribution measures:

attrConc Attributes concentration coefficient [76]:

$\left[conc_{\vec{a}_1,\vec{a}_2}, conc_{\vec{a}_2,\vec{a}_1}, \ldots, conc_{\vec{a}_{d-1},\vec{a}_d}, conc_{\vec{a}_d,\vec{a}_{d-1}}\right]$, such that

$$conc_{x,y} = \frac{\sum_{i=1}^{\phi_x} \sum_{j=1}^{\phi_y} \frac{\pi_{ij}^2}{\pi_{i+}} - \sum_{j=1}^{\phi_y} \pi_{+j}^2}{1 - \sum_{j=1}^{\phi_j} \pi_{+j}^2}, \text{ where}$$

$$\pi_{ij} = P(x = \varphi_i^x, y = \varphi_j^y), \quad \pi_{i+} = \sum_{j=1}^{\phi_y} \pi_{ij} \quad \text{and} \quad \pi_{+j} = \sum_{i=1}^{\phi_x} \pi_{ij}. \tag{A.9}$$

classConc Class concentration coefficient [76]:

$\left[conc_{\vec{a}_1,\vec{y}}, \ldots, conc_{\vec{a}_d,\vec{y}}\right]$, where $conc_{x,y}$ is given by Eq. (A.9).

propPCA Proportion of principal components that explain a specific variance of the dataset [73]:

$$\frac{|\Lambda| - \sum_{i=1}^{|\lambda|} \mathbb{1}\left(\sum_{j=1}^{i} \lambda_i > \alpha\right) + 1}{|\Lambda|},$$

where Λ is the set of all eigenvalues λ_i inversely ordered according to their variance and α is a user defined threshold indicating the amount of variance desired, e.g. 0.95.

sparsity Attributes sparsity [106]: $\left[sparsity_{\vec{a}_1}, \ldots, sparsity_{\vec{a}_d}\right]$, such that

$$sparsity_x = \frac{1}{n-1}\left(\frac{\sum_{i=1}^{\phi_x} N(x = \varphi_i^x)}{\phi_x} - 1\right),$$

where $N(x = \varphi_i^x)$ is the number of times that the ith distinct value of x are present in the vector. This is the normalized version adapted by the authors.

Case base measures:

consistencyRatio Proportion of repeated instances that have different targets [95]:

$$\frac{1}{n} \sum_{i=2}^{n} \sum_{j=1}^{i-1} \mathbb{1}(dist_{\vec{x}_i,\vec{x}_j} = 0 \wedge y_i \neq y_j)$$

incoherenceRatio Ratio of instances that does not overlap with any other instances in a predefined number of attributes [95]:

$$\frac{1}{n} \sum_{i=2}^{n} \mathbb{1}\left(\sum_{j=1}^{i-1} \mathbb{1}(o(\vec{x}_i, \vec{x}_j) > \alpha) = 0\right) \qquad o(\vec{x}_i, \vec{x}_j) = \sum_{l=1}^{d} \mathbb{1}(v_{il} = v_{jl}),$$

where α is a user hyperparameter to set the number of similar attributes to define when two instances overlap.

uniquenessRatio Proportion of repeated instances [95]:

$$\frac{1}{n} \sum_{i=2}^{n} \sum_{j=1}^{i-1} \mathbb{1}(dist_{\vec{x}_i,\vec{x}_j} = 0)$$

Concept based measures:

cohesiveness Density of the example distribution [96]: $\left[v(\vec{x}_1), \ldots, v(\vec{x}_n)\right]$.

$$v(\vec{x}_i) = \frac{1}{|\mathcal{K}|} \sum_{(\vec{x}_j, y_j) \in \mathcal{K}_{\vec{x}_i}} 1 - \mathbb{1}(y_i = y_j),$$

where $\mathcal{K}_{\vec{x}_i}$ contains the k nearest neighbors of instance $\vec{x}_i$. The k is a user hyperparameter.

distInst Distance between all pairs of instances [125]:

$$\left[dist_{\vec{x}_1,\vec{x}_2}, dist_{\vec{x}_1,\vec{x}_3}, \ldots, dist_{\vec{x}_{n-2},\vec{x}_n}, dist_{\vec{x}_{n-1},\vec{x}_n}\right] \tag{A.10}$$

distCorrInst Distance and correlations of all pairs of instances [44]: $\left[c', d'\right]$, where

$$c' = \frac{c+1}{2}, \qquad c = \left[cor_{\vec{x}_1,\vec{x}_2}, cor_{\vec{x}_1,\vec{x}_3}, \ldots, cor_{\vec{x}_{n-2},\vec{x}_n}, cor_{\vec{x}_{n-1},\vec{x}_n}\right],$$

$$d' = \frac{d - \min(d)}{\max(d) - \min(d)},$$

such that $cor_{x,y}$ (Eq. (A.3)) is used to compute the correlation between 2 instances and d is given by Eq. (A.10).

gravity Center of gravity [43]: $dist_{\bar{\vec{x}}_{C_m}, \bar{\vec{x}}_{C_n}}$, where $\bar{\vec{x}}_{C_m}$ and $\bar{\vec{x}}_{C_n}$ are the center points of the instances related to the majority and minority classes, respectively.

wgDist Weighted distance [97]: $\left[v(\vec{x}_1), \ldots, v(\vec{x}_n)\right]$, where

$$v(\vec{x}_i) = \frac{\sum_{j=1, j\neq i}^{n} W(\vec{x}_i, \vec{x}_j) dist_{\vec{x}_i, \vec{x}_j}}{\sum_{j=1, j\neq i}^{n} W(\vec{x}_i, \vec{x}_j)}$$

$$W(\vec{x}_i, \vec{x}_j) = \frac{1}{2^{2d}} \qquad d = \frac{dist_{\vec{x}_i, \vec{x}_j}}{\sqrt{n - dist_{\vec{x}_i, \vec{x}_j}}}$$

Structural information measures:

oneItemset Frequency of the predictive attributes after they are binarized [99]:

$$\left[\frac{\sum_{i=1}^{n} v_{i1}}{n}, \ldots, \frac{\sum_{i=1}^{n} v_{id}}{n}\right].$$

twoItemset Frequency of predictive attributes' pairs after they are binarized [99]:

$$[v(1,2), v(1,3), \ldots, v(d-2,d), v(d-1,d)]$$

$$v(i,j) = \frac{1}{n} \sum_{l=1}^{n} \mathbb{1}(v_{li} \neq v_{lj}).$$

Time based measures:

infotheoTime The elapsed time to compute the information theoretical meta-features [19].

landTime The elapsed time to compute the landmarkings meta-features [19].

modelTime The elapsed time to compute the model-based meta-features [19].

statTime The elapsed time to compute the statistical meta-features [19].

References

[1] D.H. Wolpert, Stacked generalization, Neural Netw. 5 (2) (1992) 241–259.
[2] S.P. Adam, S.-A.N. Alexandropoulos, P.M. Pardalos, M.N. Vrahatis, No free lunch theorem: A review, in: Approximation And Optimization : Algorithms, Complexity And Applications, Springer International Publishing, 2019, pp. 57–82.
[3] P. Brazdil, C. Giraud-Carrier, C. Soares, R. Vilalta, Metalearning: Applications to Data Mining, Springer-Verlag Berlin Heidelberg, 2009.
[4] J. Vanschoren, Meta-learning: A survey, 2018, pp. 1–29, arXiv:1810.03548.
[5] J.N. van Rijn, F. Hutter, Hyperparameter importance across datasets, in: 24th ACM SIGKDD International Conference On Knowledge Discovery & Data Mining, 2018, pp. 2367–2376.
[6] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[7] T.M. Hospedales, A. Antoniou, P. Micaelli, A.J. Storkey, Meta-learning in neural networks: A survey, 2020, CoRR abs/2004.05439.
[8] R. Elshawi, S. Sakr, Automated machine learning: Techniques and frameworks, in: European Big Data Management And Analytics Summer School, EBISS, 2019, pp. 40–69.
[9] F. Hutter, L. Kotthoff, J. Vanschoren, Automated Machine Learning, Springer International Publishing, 2019.
[10] K. Smith-Miles, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Comput. Surv. 41 (1) (2008) 6:1–6:25.
[11] M.A.M. noz, L. Villanova, D. Baatar, K. Smith-Miles, Instance spaces for machine learning classification, Mach. Learn. 107 (1) (2018) 109–147.
[12] H. Bensusan, A. Kalousis, Estimating the predictive accuracy of a classifier, in: 12th European Conference On Machine Learning, ECML, 2001, pp. 25–36.
[13] B. Bilalli, A. Abelló, T. Aluja-Banet, On the predictive power of meta-features in OpenML, Int. J. Appl. Math. Comput. Sci. 27 (4) (2017) 697–712.
[14] A. Rivolli, L.P.F. Garcia, A.C. Lorena, A.C.P.L.F. de Carvalho, A study of the correlation of metafeatures used for metalearning, in: International Work-Conference On Artificial Neural Networks, IWANN, 2021, pp. 471–483.
[15] B. Pfahringer, H. Bensusan, C. Giraud-Carrier, Meta-learning by landmarking various learning algorithms, in: 17th International Conference On Machine Learning, ICML, 2000, pp. 743–750.
[16] Y. Peng, P.A. Flach, C. Soares, P. Brazdil, Improved dataset characterisation for meta-learning, in: 5th International Conference On Discovery Science, DS, 2002, pp. 141–152.
[17] C. Castiello, G. Castellano, A.M. Fanelli, Meta-data: Characterization of input features for meta-learning, in: 2nd International Conference On Modeling Decisions For Artificial Intelligence, MDAI, 2005, pp. 457–468.
[18] A. Filchenkov, A. Pendryak, Datasets meta-feature description for recommending feature selection algorithm, in: Artificial Intelligence And Natural Language And Information Extraction, Social Media And Web Search FRUCT Conference, AINL-ISMW FRUCT, 2015, pp. 11–18.
[19] M. Reif, F. Shafait, A. Dengel, Prediction of classifier training time including parameter optimization, in: 34th German Conference On Advances In Artificial Intelligence, KI, 2011, pp. 260–271.
[20] M. Reif, F. Shafait, M. Goldstein, T. Breuel, A. Dengel, Automatic classifier selection for non-experts, Pattern Anal. Appl. 17 (1) (2014) 83–96.
[21] R. Vilalta, Y. Drissi, A perspective view and survey of meta-learning, Artif. Intell. Rev. 18 (2) (2002) 77–95.
[22] C. Lemke, M. Budka, B. Gabrys, Metalearning: a survey of trends and technologies, Artif. Intell. Rev. 44 (1) (2015) 117–130.
[23] I. Khan, X. Zhang, M. Rehman, R. Ali, A literature survey and empirical study of meta-learning for classifier selection, IEEE Access 8 (2020) 10262–10281.
[24] N. Macià, E. Bernadó-Mansilla, Towards UCI+: A mindful repository design, Inform. Sci. 261 (2014) 237–262.
[25] M. Reis, A.C. Lorena, sample bias effect on meta-learning, in: Anais do Encontro Nacional de Inteligência Artificial e Computacional, ENIAC 2020, 2020, pp. 294–305.
[26] A. Kalousis, J. ao Gama, M. Hilario, On data and algorithms: Understanding inductive performance, Mach. Learn. 54 (3) (2004) 275–312.
[27] D. Oreski, S. Oreski, B. Klicek, Effects of dataset characteristics on the performance of feature selection techniques, Appl. Soft Comput. 52 (2017) 109–119.
[28] T.R. França, P.B.C. de Miranda, R.B.C. Prudêncio, A.C. Lorenaz, A.C.A. Nascimento, A many-objective optimization approach for complexity-based data set generation, in: IEEE Congress On Evolutionary Computation, CEC, 2020, pp. 1–8.
[29] T.M. Mitchell, Machine Learning, McGraw Hill, 1997.
[30] D.H. Wolpert, W.G. Macready, No Free Lunch Theorems for Search, Technical Report SFI-TR-05-010, Santa Fe Institute, 1995, pp. 1–38.
[31] P. Brazdil, J. ao Gama, B. Henery, Characterizing the applicability of classification algorithms using meta-level learning, in: 7th European Conference On Machine Learning, ECML, 1994, pp. 83–102.
[32] J.R. Rice, The algorithm selection problem, Adv. Comput. 15 (1976) 65–118.
[33] C. Soares, J. Petrak, P. Brazdil, Sampling-based relative landmarks: Systematically test-driving algorithms before choosing, in: Portuguese Conference On Artificial Intelligence, vol. 2258, EPIA, 2001, pp. 88–95.
[34] M. Reif, A comprehensive dataset for evaluating approaches of various meta-learning tasks, in: 1st International Conference On Pattern Recognition Applications And Methods, ICPRAM, 2012, pp. 273–276.
[35] S. Ali, K.A. Smith-Miles, A meta-learning approach to automatic kernel selection for support vector machines, Neurocomputing 70 (1) (2006) 173–186.
[36] R.G. Mantovani, A.L.D. Rossi, E. Alcobaça, J. Vanschoren, A.C.P.L.F. de Carvalho, A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves SVM classifiers, Inform. Sci. 501 (2019) 193–221.
[37] J.A. Sáez, E. Corchado, A meta-learning recommendation system for characterizing unsupervised problems: On using quality indices to describe data conformations, IEEE Access 7 (2019) 63247–63263.
[38] L.P.F. Garcia, A. Rivolli, E. Alcobaça, A.C. Lorena, A.C.P.L.F. de Carvalho, Boosting meta-learning with simulated data complexity measures, Intelligent Data Analysis 24 (5) (2020) 1011–1028.
[39] V.H. Barella, L.P.F. Garcia, A.C.P.L.F. de Carvalho, Simulating complexity measures on imbalanced datasets, in: Brazilian Conference On Intelligent Systems, BRACIS, 2020, pp. 498–512.
[40] M.M. Meskhi, A. Rivolli, R.G. Mantovani, R. Vilalta, Learning abstract task representations, 2021, pp. 1–7, arXiv:2101.07852.
[41] J.W. Lee, C. Giraud-Carrier, Predicting algorithm accuracy with a small set of effective meta-features, in: 7th International Conference On Machine Learning And Applications, ICMLA, 2008, pp. 808–812.
[42] B. Bilalli, A. Abelló, T. Aluja-Banet, R. Wrembel, Intelligent assistance for data pre-processing, Comput. Stand. Interfaces 57 (2018) 101–109.

[43] S. Ali, K.A. Smith, On learning algorithm selection for classification, Appl. Soft Comput. 6 (2) (2006) 119–138.
[44] B.A. Pimentel, A.C.P.L.F. de Carvalho, A new data characterization for selecting clustering algorithms using meta-learning, Inform. Sci. 477 (2019) 203–219.
[45] A.C.A. Nascimento, R.B.C. Prudêncio, M.C.P. de Souto, I.G. Costa, Mining rules for the automatic selection process of clustering methods applied to cancer gene expression data, in: 19th International Conference On Artificial Neural Networks, vol. 5769, ICANN, 2009, pp. 20–29.
[46] M. Vukicevic, S. Radovanovic, B. Delibasic, M. Suknovic, Extending meta-learning framework for clustering gene expression data with component-based algorithm design and internal evaluation measures, Int. J. Data Min. Bioinform. 14 (2) (2016) 101–119.
[47] L. Rokach, Decomposition methodology for classification tasks: a meta decomposer framework, Pattern Anal. Appl. 9 (2–3) (2006) 257–271.
[48] G.J. Aguiar, R.G. Mantovani, S.M. Mastelini, A.C.P.L.F. de Carvalho, G.F.C. Campos, S.B. Junior, A meta-learning approach for selecting image segmentation algorithm, Pattern Recognit. Lett. 128 (2019) 480–487.
[49] L.P.F. Garcia, A.C.P.L.F. de Carvalho, A.C. Lorena, Noise detection in the meta-learning level, Neurocomputing 176 (2) (2015) 1–12.
[50] A.L.D. Rossi, B.F. de Souza, C. Soares, A.C.P. de Leon Ferreira de Carvalho, A guidance of data stream characterization for meta-learning, Intell. Data Anal. 21 (4) (2017) 1015–1035.
[51] T. Cunha, C. Soares, A.C. de Carvalho, Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering, Inform. Sci. (ISSN: 0020-0255) 423 (2018) 128–144.
[52] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search: A survey, 2019, arXiv:1808.05377.
[53] T.M. Hospedales, A. Antoniou, P. Micaelli, A.J. Storkey, Meta-learning in neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. (2021) 1–20.
[54] R. Elshawi, M. Maher, S. Sakr, Automated machine learning: State-of-the-art and open challenges, 2019, arXiv:1906.02287.
[55] M.-A. Zöller, M.F. Huber, Benchmark and survey of automated machine learning frameworks, J. Artif. Intell. Res. 70 (2021) 409–472.
[56] M. Huisman, J.N. van Rijn, A. Plaat, A survey of deep meta-learning, Artif. Intell. Rev. 54, 4483–4541.
[57] A. Kalousis, M. Hilario, Feature selection for meta-learning, in: 5th Pacific-Asia Conference On Knowledge Discovery And Data Mining, vol. 2035, PAKDD, 2001, pp. 222–233.
[58] S.Y. Sohn, Meta analysis of classification algorithms for pattern recognition, IEEE Trans. Pattern Anal. Mach. Intell. 21 (11) (1999) 1137–1144.
[59] A. Kalousis, T. Theoharis, NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection, Intell. Data Anal. 3 (5) (1999) 319–337.
[60] L. Todorovski, P. Brazdil, C. Soares, Report on the experiments with feature selection in meta-level learning, in: PKDD Workshop On Data Mining, Decision Support, Meta-Learning And Inductive Logic Programming, 2000, pp. 27–39.
[61] M. Reif, F. Shafait, A. Dengel, Meta^2-features: Providing meta-learners more information, in: 35th German Conference On Artificial Intelligence, KI, 2012, pp. 74–77.
[62] H. Bensusan, C. Giraud-Carrier, C. Kennedy, A higher-order approach to meta-learning, in: 10th International Conference Inductive Logic Programming, ILP, 2000, pp. 33–42.
[63] C. Kopf, C. Taylor, J. Keller, Meta-analysis: From data characterisation for meta-learning to meta-regression, in: PKDD Workshop On Data Mining, Decision Support, Meta-Learning And Inductive Logic Programming, 2000, pp. 15–26.
[64] S. Segrera, J. Pinho, M.N. Moreno, Information-theoretic measures for meta-learning, in: Hybrid Artificial Intelligence Systems, HAIS, 2008, pp. 458–465.
[65] J. Fürnkranz, J. Petrak, An evaluation of landmarking variants, in: 1st ECML/PKDD International Workshop On Integration And Collaboration Aspects Of Data Mining, Decision Support And Meta-Learning, IDDM, 2001, pp. 57–68.
[66] J. Vanschoren, H. Blockeel, B. Pfahringer, G. Holmes, Experiment databases, Mach. Learn. 87 (2) (2012) 127–158.
[67] L.P.F. Garcia, F. Campelo, G.N. Ramos, A. Rivolli, A.C.P.L.F. de Carvalho, Evaluating clustering meta-features for classifier recommendation, in: 10th Brazilian Conference On Intelligent Systems, BRACIS, 2021, pp. 453–467.
[68] R. Engels, C. Theusinger, Using a data metric for preprocessing advice for data mining applications, in: 13th European Conference On On Artificial Intelligence, ECAI, 1998, pp. 430–434.
[69] G. Lindner, R. Studer, AST: Support for algorithm selection with a CBR approach, in: European Conference On Principles Of Data Mining And Knowledge Discovery, PKDD, 1999, pp. 418–423.
[70] J. Vanschoren, Understanding Machine Learning Performance with Experiment Databases (Ph.D. thesis), Leuven Univeristy, 2010.
[71] F. Pinto, C. Soares, J. ao Mendes-Moreira, Towards automatic generation of metafeatures, in: Pacific-Asia Conference On Knowledge Discovery And Data Mining, PAKDD, 2016, pp. 215–226.
[72] P. Kuba, P. Brazdil, C. Soares, A. Woznica, Exploiting sampling and meta-learning for parameter setting for support vector machines, in: 8th IBERAMIA Workshop On Learning And Data Mining, 2002, pp. 209–216.
[73] M. Feurer, J.T. Springenberg, F. Hutter, Using meta-learning to initialize Bayesian optimization of hyperparameters, in: International Conference On Meta-Learning And Algorithm Selection, MLAS, 2014, pp. 3–10.
[74] Y. Peng, P.A. Flach, P. Brazdil, C. Soares, Decision tree-based data characterization for meta-learning, in: 2nd ECML/PKDD International Workshop On Integration And Collaboration Aspects Of Data Mining, Decision Support And Meta-Learning, IDDM, 2002, pp. 111–122.
[75] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural And Statistical Classification, Ellis Horwood, 1994.
[76] A. Kalousis, M. Hilario, Model selection via meta-learning: a comparative study, Int. J. Artif. Intell. Tools 10 (4) (2001) 525–554.
[77] P.B. Brazdil, C. Soares, J.P. da Coasta, Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results, Mach. Learn. 50 (3) (2003) 251–277.
[78] J.L. Rodgers, W.A. Nicewander, Thirteen ways to look at the correlation coefficient, Amer. Statist. 42 (1) (1988) 59–66.
[79] D.N. Joanes, C.A. Gill, Comparing measures of sample skewness and kurtosis, J. R. Stat. Soc. 47 (1) (1998) 183–189.
[80] K. Smith, F. Woo, V. Ciesielski, R. Ibrahim, Modelling the relationship between problem characteristics and data mining algorithm performance using neural networks, in: Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, And Complex Systems, 2001, pp. 357–362.
[81] W.-Y. Loh, Fifty years of classification and regression trees, Internat. Statist. Rev. 82 (3) (2014) 329–348.
[82] P. Nguyen, J. Wang, M. Hilario, A. Kalousis, Learning heterogeneous similarity measures for hybrid-recommendations in meta-mining, in: IEEE International Conference On Data Mining, ICDM, 2012, pp. 1026–1031.
[83] R. Leite, P. Brazdil, Predicting relative performance of classifiers from samples, in: 22nd International Conference On Machine Learning, vol. 119, ICML, 2005, pp. 497–503.
[84] Q. Sun, B. Pfahringer, Pairwise meta-rules for better meta-learning-based algorithm ranking, Mach. Learn. 93 (1) (2013) 141–161.
[85] L.P.F. Garcia, A.C. Lorena, M.C.P. de Souto, T.K. Ho, Classifier recommendation using data complexity measures, in: 24th International Conference On Pattern Recognition, ICPR, 2018, pp. 874–879.
[86] G. Morais, R.C. Prati, Complex network measures for data set characterization, in: Brazilian Conference On Intelligent Systems, BRACIS, 2013, pp. 12–18.
[87] D. Ler, H. Teng, Y. He, R. Gidijala, Algorithm selection for classification problems via cluster-based meta-features, in: IEEE International Conference On Big Data, Big Data, 2018, pp. 4952–4960.
[88] J. Handl, J.D. Knowles, D.B. Kell, Computational cluster validation in post-genomic data analysis, Bioinformatics 21 (15) (2005) 3201–3212.
[89] B. Desgraupes, An R package for computing clustering quality indices, 2017, URL: https://CRAN.R-project.org/package=clusterCrit/vignettes/clusterCrit.pdf.
[90] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 289–300.
[91] J. Luengo, F. Herrera, An automatic extraction method of the domains of competence for learning classifiers using data complexity measures, Knowl. Inf. Syst. 42 (1) (2015) 147–180.
[92] M.R. Smith, T. Martinez, C. Giraud-Carrier, An instance level analysis of data complexity, Mach. Learn. 95 (2) (2014) 225–256.
[93] A.C. Lorena, L.P.F. Garcia, J. Lehmann, M.C.P. Souto, T.K. Ho, How complex is your classification problem? A survey on measuring classification complexity, ACM Comput. Surv. 52 (5) (2019).
[94] V.H. Barella, L.P.F. Garcia, M.C.P. de Souto, A.C. Lorena, A.C.P.L.F. de Carvalho, Data complexity measures for imbalanced classification tasks, in: International Joint Conference On Neural Networks, IJCNN, 2018, pp. 1–8.
[95] C. Kopf, I. Iglezakis, Combination of task description strategies and case base properties for meta-learning, in: 2nd ECML/PKDD International Workshop On Integration And Collaboration Aspects Of Data Mining, Decision Support And Meta-Learning, IDDM, 2002, pp. 65–76.
[96] R. Vilalta, Y. Drissi, A characterization of difficult problems in classification, in: International Conference On Machine Learning And Applications, ICMLA, 2002, pp. 133–138.
[97] R. Vilalta, Understanding accuracy performance through concept characterization and algorithm analysis, in: ECML Workshop On Recent Advances In Meta-Learning And Future Work, 1999, pp. 3–9.
[98] G. Wang, Q. Song, X. Zhu, An improved data characterization method and its application in classification algorithm recommendation, Appl. Intell. 43 (4) (2015) 892–912.

[99] Q. Song, G. Wang, C. Wang, Automatic recommendation of classification algorithms based on data set characteristics, Pattern Recognit. 45 (7) (2012) 2672–2689.
[100] S.H. Burton, R.G. Morris, C. Giraud-Carrier, J.H. West, R. Thackeray, Mining useful association rules from questionnaire data, Intell. Data Anal. 18 (3) (2014) 479–494.
[101] C.C. Aggarwal, Data Mining, Springer International Publishing, 2015.
[102] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley Longman Publishing, 2005.
[103] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: 13th International Joint Conference On Artificial Intelligence, IJCAI, 1993, pp. 1022–1029.
[104] J. Han, M. Kamber, J. Pei, Data Mining: Concepts And Techniques, Morgan Kaufmann, 2005.
[105] G. Wang, Q. Song, H. Sun, X. Zhang, B. Xu, Y. Zhou, A feature subset selection algorithm automatic recommendation method, J. Artif. Intell. Res. 47 (2013) 1–34.
[106] M.A. Salama, A.E. Hassanien, K. Revett, Employment of neural network and rough set in meta-learning, Memet. Comput. 5 (3) (2013) 165–177.
[107] H. Hotelling, Analysis of a complex of statistical variables with principal components, J. Educ. Psychol. 24 (1933) 417–441.
[108] S.D. Abdelmessih, F. Shafait, M. Reif, M. Goldstein, Landmarking for meta-learning using RapidMiner, in: RapidMiner Community Meeting And Conference, RCOMM, 2010, pp. 1–6.
[109] D. Dua, C. Graff, UCI machine learning repository, 2017, URL: http://archive.ics.uci.edu/ml.
[110] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, Multiple-Valued Log. Soft Comput. 17 (2–3) (2011) 255–287.
[111] M.L. Braun, C.S. Ong, P.O. Hoyer, S. Henschel, S. Sonnenburg, mldata.org: machine learning data set repository, 2014, http://mldata.org/.
[112] J. Vanschoren, J.N. van Rijn, B. Bischl, L. Torgo, OpenML: Networked science in machine learning, ACM SIGKDD Explor. Newsl. 15 (2) (2013) 49–60.
[113] J. Vanschoren, H. Blockeel, Towards understanding learning behavior, in: 15th Annual Machine Learning Conference Of Belgium And The Netherlands, 2006, pp. 89–96.
[114] R.B.C. Prudêncio, T.B. Ludermir, Active learning to support the generation of meta-examples, in: 17th International Conference On Artificial Neural Networks, vol. 4668, ICANN, 2007, pp. 817–826.
[115] R.B.C. Prudêncio, C. Soares, T.B. Ludermir, Uncertainty sampling-based active selection of datasetoids for meta-learning, in: 21st International Conference On Artificial Neural Networks, vol. 6792, ICANN, 2011, pp. 454–461.
[116] H. Bensusan, C. Giraud-Carrier, Discovering task neighbourhoods through landmark learning performances, in: 4th European Conference On Principles Of Data Mining And Knowledge Discovery, PKDD, 2000, pp. 325–330.
[117] Mathworks, Statistics Toolbox: for Use with MATLAB: User's Guide, 2001.
[118] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: An update, ACM SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[119] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, T. Euler, YALE: rapid prototyping for complex data mining tasks, in: 12th International Conference On Knowledge Discovery And Data Mining, KDD, 2006, pp. 935–940.
[120] A. Balte, N. Pise, P. Kulkarni, Meta-learning with landmarking : A survey, Int. J. Comput. Appl. 105 (8) (2014) 47–51.
[121] E. Alcobaça, F. Siqueira, A. Rivolli, L.P. Garcia, J.T. Oliva, A.C. de Carvalho, MFE: Towards reproducible meta-feature extraction, J. Mach. Learn. Res. 21 (111) (2020) 1–5.
[122] A. Kalousis, Algorithm Selection via Meta-Learning (Ph.D. thesis), Faculty of Science of the University of Geneva, 2002.
[123] P. Royston, Remark AS R94: A remark on algorithm AS 181: The W-test for normality, J. R. Stat. Soc. Ser. C Appl. Stat. 44 (4) (1995) 547–551.
[124] P.J. Rousseeuw, M. Hubert, Robust statistics for outlier detection, Wiley Interdiscip. Rev. Data Min. And Knowledge Discovery 1 (1) (2011) 73–79.
[125] D.G. Ferrari, L.N. de Castro, Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods, Inform. Sci. 301 (2015) 181–194.