Neurocomputing 289 (2018) 68–85

Contents lists available at ScienceDirect

Neurocomputing

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

Tips, guidelines and tools for managing multi-label datasets: The mldr.datasets R package and the Cometa data repository

Francisco Charte[a,*], Antonio J. Rivera[a], David Charte[b], María J. del Jesus[a], Francisco Herrera[b,c]

[a] Department of Computer Science, University of Jaén, Jaén 23071, Spain
[b] Department of Computer Science and A.I., University of Granada, Granada 18071, Spain
[c] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia

ARTICLE INFO

Article history:
Received 7 December 2017
Revised 24 January 2018
Accepted 2 February 2018
Available online 8 February 2018

Communicated by Nianyin Zeng

Keywords:
Multi-label
Software
Tools
Datasets
Repository

ABSTRACT

New proposals in the field of multi-label learning algorithms have been growing in number steadily over the last few years. The experimentation associated with each of them always goes through the same phases: selection of datasets, partitioning, training, analysis of results and, finally, comparison with existing methods. This last step is often hampered since it involves using exactly the same datasets, partitioned in the same way and using the same validation strategy. In this paper we present a set of tools whose objective is to facilitate the management of multi-label datasets, aiming to standardize the experimentation procedure. The two main tools are an R package, mldr.datasets, and a web repository with datasets, Cometa. Together, these tools will simplify the collection of datasets, their partitioning, documentation and export to multiple formats, among other functions. Some tips, recommendations and guidelines for a good experimental analysis of multi-label methods are also presented.

1. Introduction

The need to automatically label data has significantly increased in recent years, in line with the growth of multimedia content online, especially all types of social networks. People and objects present in a photograph recently uploaded to Instagram or Facebook, subjects and areas related to an article published in a digital newspaper, or styles and emotions linked to a new melody must be determined as quickly and accurately as possible. The large flow of new information published every minute on the Internet requires this functionality, essential to catalog each piece of data. This demand is satisfied by multi-label learning algorithms [1,2], able to learn from prelabeled examples and then do this task automatically.

The knowledge obtained from these prelabeled data instances can be represented in disparate ways, i.e. decision trees [3], neural networks [4], support vector machines [5], etc. Since there are several labels associated to each data sample, the structure of these models tends to be slightly more complex than is usual in traditional classification. Alternatively, there also exist certain label transformation techniques, such as binarization [6] and label powerset [7], oriented to applying traditional classifiers to multi-label data. In addition, there are some very specific casuistries, such as imbalanced labels concurrence [8] or high dimensionality both on the feature space [9] and label space [10]. As a consequence, a plethora of multi-label classification (MLC) algorithms have been proposed lately, each of them claiming to perform better than the previous ones.

Proposing a new learning method implies comparing it with some existing algorithms. Doing so requires conducting an empirical experimentation. The experimental process customarily consists in the following steps:

1. Collect some multilabel datasets (MLDs), analyze their traits to choose those most suitable to the task, and properly document them. Additionally, some data preparation steps may be performed, such as binarization.
2. Run the proposed algorithm with the chosen data, and obtain a collection of performance indicators.
3. Compare the indicators of the new method with those of some existing ones, so as to assess the proposal.
4. Tune the algorithm as convenient and return to step 2 until it achieves a clear improvement.

Although the process is apparently clear, accomplishing it in a proper way is not always straightforward. Practitioners frequently

* Corresponding author.
E-mail addresses: fcharte@ujaen.es (F. Charte), arivera@ujaen.es (A.J. Rivera), fdavidcl@ugr.es (D. Charte), mjjesus@ujaen.es (M.J. del Jesus), herrera@ugr.es (F. Herrera).

https://doi.org/10.1016/j.neucom.2018.02.011
0925-2312/

fail in some steps, drawing conclusions of doubtful correctness. Sometimes the reason is in the scarcity of appropriate tools. Occasionally the obstacle is the lack of experience in such a specialized field.

As long-time MLC scholars we have developed several tools over the years to ease our work. In addition, we follow a standardized procedure for performing MLC experiments aiming to draw sound conclusions. Our goal in this paper is to present some of these tools, specifically the `mldr.datasets` R package and the comprehensive multi-label data source, Cometa. Furthermore, how to use these tools in order to avoid some of the pitfalls usually found during MLC experimentation is also explained.

The main contributions in this paper can be summarized as follows:

- We have identified the main pitfalls while performing multi-label experiments.
- A collection of good practices aimed to overcome the previous traps is provided.
- We have developed a new tool, the `mldr.datasets` package introduced in Section 5, with the goal of easing multi-label data selection and preparation.
- By means of the previous tool, a comprehensive data repository has been generated. Cometa, presented in Section 6, is a web application allowing to filter, search and select datasets, available in different file formats and partitioning schemes.
- A Docker image is provided to allow anyone to build their own multi-label data repository, automating the use of the previous tools.

The remainder of this paper is structured as follows. The foundations about multi-label learning are provided in Section 2. Section 3 describes how MLC experiments are usually conducted, while Section 4 explains some of the frequent pitfalls and the way them can be surpassed. Section 5 introduces the `mldr.datasets` R package. This tool has been used to build Cometa, the data repository presented in Section 6. Lastly, Section 7 provides the final conclusions.

2. Multi-label learning background

The main focus in this paper is put on the process to perform MLC and the tools needed to do it. However, MLC is part of broad field generically known as multi-label learning (MLL), where other kinds of tasks can be conducted as well. This section provides a brief introduction to MLL, a topic to which dozens of papers [2] and even full books [1] have been devoted.

2.1. MLL foundations

Most common supervised machine learning tasks, such as binary and multiclass classification or regression, usually are guided only by an objective value. This would be the class assigned to a data pattern, in classification, or the target value to obtain as result of some kind of regression computation. Even methods which perform frequently non-supervised tasks, such as clustering, sometimes use this objective value to improve their results.

Multi-label learning [1] differs from the previous ones by the nature of the objective that guides the process. It is a set of binary values stating which labels are relevant to each data pattern, rather than a single class. Assuming D is a dataset having f input features, and being L the full set of labels appearing in D, each data pattern would be constructed as shown in (1).

$$D_i = (X_i, Y_i) \mid X_i \in X^1 \times X^2 \times \cdots \times X^f, Y_i \subseteq L \tag{1}$$

From this definition, mainly from the set of relevant labels (labelset) for each data sample (Y_i), it is easy to compute certain characterization metrics, as described in the following subsection.

2.2. Characterization metrics

Characterization measurements are useful to know traits of multi-label data, such as the multi-labeleness degree of a dataset, its imbalance level, the label sparseness, etc., thus being fundamental to choose the proper datasets for each case. The described below are among the most used ones.

Label cardinality. It is defined in [1] as shown in (2), where D is any MLD, n the number of instances, k the number of labels, and Y_i the labelset corresponding to the ith data sample. *Card* is the average number of relevant labels (number of labels active per instance) for the MLD D.

$$Card(D) = \frac{1}{n}\sum_{i=1}^{n}|Y_i| \tag{2}$$

Label density. It is defined as (3). Usually, *Card* changes along with the total number of distinct labels. So a normalized version, named *Dens*, is defined as *Card* divided by the total number of labels.

$$Dens(D) = \frac{1}{k}\frac{1}{n}\sum_{i=1}^{n}|Y_i| \tag{3}$$

meanIR. This measure is computed as (4) the average imbalance ratio of each label, the *IRLbl* (5). In these equations L stands for the full set of labels appearing in the MLD. Both measures were introduced in [11] to assess the imbalance level in an MLD.

$$MeanIR = \frac{1}{k}\sum_{l\in L} IRLbl(l). \tag{4}$$

$$IRLbl(l) = \frac{\max_{l'\in L}\left(\sum_{i=1}^{n}[\![l' \in Y_i]\!]\right)}{\sum_{i=1}^{n}[\![l \in Y_i]\!]}. \tag{5}$$

SCUMBLE and SCUMBLE.CV. These two metrics are aimed to evaluate the level of concurrence among minority and majority labels. Introduced in [12], the former is defined (6) as the average *SCUMBLE* of each instance (7) in the dataset. The latter is simply the coefficient of variation associated to this average.

$$SCUMBLE(D) = \frac{1}{n}\sum_{i=1}^{n} SCUMBLE_i \tag{6}$$

$$SCUMBLE_i = 1 - \frac{1}{\overline{IRLbl}_i}\left(\prod_{l\in L} IRLbl_{il}\right)^{(1/k)} \tag{7}$$

TCS. This metric (8) was presented in [13] as a straightforward way to assess the theoretical complexity of an MLD. It is based on just three traits of the dataset: f stands for the amount of input features, k for the number of labels, and ls is the total number of label combinations in D. The larger is the value returned by this measurement, the harder would be to learn a predictive model from the dataset.

$$TCS(D) = \log(f \times k \times ls) \tag{8}$$

All these metrics can be easily obtained through the `mldr.datasets`, as described in Section 5.

2.3. Main MLL tasks

Aside from performing exploratory data analysis, different machine learning tasks can be faced while working with multi-label data. The following are among the most usual ones:

Classification. It is arguably the most studied problem in multi-label learning. The goal is to find a model able to predict the labelset for new data patterns. As described in [1], two main approaches to model a new classifier exist. The first one aims to transform the original data so that traditional classifiers can be used. The two main transformation techniques are known as BR (*Binary Relevance*) [6] and LP (*Label Powerset*) [7]. The former deals with each label separately, using a set of binary classifiers to make the prediction. while the latter joins the labels to create a class identifier, relying in a multiclass classifier as predictive model. The second approach consists in adapting existing classification methods to handle multi-label patterns, instead of transforming them. The use of ensembles is also very popular in the field, with proposals as ECC (*Ensemble of Classifier Chains*) [14], EPS (*Ensemble of Pruned Sets*) [15], etc.

Ranking. As the name denotes, label ranking methods are used to elaborate a ranking of labels according to their relevance for a data pattern. Therefore, instead of producing a labelset, a string of 0s and 1s stating which labels are predicted as classifiers do, these methods assign a weight to each label. This label ranking can be used directly, as well as transformed into a predicted labelset by applying a specific threshold. There are many proposed label ranking methods, RPC (*Ranking by Pairwise Comparison*) [16] and CLR (*Calibrated Label Ranking*) [17] are two of the best known.

Clustering. Usually, clustering methods [18] work in an unsupervised manner. Therefore, there would be no difference between clustering binary, multiclass or multi-label data. However, sometimes class information is taken into account to improve clustering results. Both basic and hierarchical clustering have been used as tools to create predictive multi-label methods. For instance, the ML-RBF algorithm [19] relies on the classic k-means algorithm to cluster the data points and use the clusters as centers of the radial basis functions in the hidden layer. The HOMER algorithm [20] produces an hierarchical model by clustering the patterns in each node, introducing the concept of *meta-label* to represent similar labels. Only a few multi-label specific clustering methods have been proposed until now. One of them [21] is an evolutionary algorithm able to perform distance metric learning. The method considers multiple labels per cluster, computing a cluster validity measure from the relationships among neighbors. In [22] a density-based algorithm, similar to DBSCAN [23], is proposed as potential solution to perform multi-label clustering.

2.4. MLL evaluation metrics

The metrics used to evaluate a result depend on the performed task, but also on the own nature of the analyzed data. While in binary classification *Accuracy* or *Precision* are the most usual performance measures, and there are only a handful more to choose from, when the class of patterns is not binary but multi-label the group of available metrics is considerably larger. The difference is that the output of a binary classifier is either correct or incorrect, while that of a multi-label can be totally or partially correct. This justifies the existence of around twenty metrics for multi-label classification only, as explained in [1, Chapter 3].

A multi-label classifier outputs a bipartition as result. That is a sequence or array of 0s and 1s, stating which labels are predicted as relevant for a data sample and which not (the predicted labelset). These predictions are aggregated to produce several confusion matrices, from which the usual classification performance metrics can be computed. Depending on how the aggregation is conducted, the metrics are grouped into two large categories: example-based and label-based metrics.

Example-based measurements are computed individually from each data pattern in the evaluated set. These values are then averaged, simply dividing by the number of evaluated samples. Some of the most usual metrics in this group are *Hamming loss, Accuracy, Precision, Recall* and *F-measure*. Unlike the other ones, *Hamming loss* (9) is not common in traditional classification. Being n the number of data points and k the number of considered labels, it calculates the symmetric difference (Δ operator) between the predicted labelset Y_i and the ground truth Z_i, thus counting the number of mismatches. Therefore, it is a performance metric to be minimized instead of maximized.

$$Hamming\ loss = \frac{1}{n}\frac{1}{k}\sum_{i=1}^{n}|Y_i \Delta Z_i| \tag{9}$$

Instead of mixing the results of all labels in each instance, those can be separately aggregated and the evaluation metrics computed for each label. This is the way label-based performance metrics work. In fact, there are two ways to perform the averaging, as shown in Eqs. (10) and (11). The macro-averaging approach sums the number of true positives, false positives, true negatives and false negatives for each label, and independently computes the measurement for each label. Thus the metric, such as *Precision, Recall, F-measure*, etc., is calculated several times, as many as labels there are. Lastly, these measurements are added and divided by the number of labels (k). By contrast, in the micro-averaging approach the metric is computed only once, after the counters for all label have been aggregated.

$$Macro\ metric = \frac{1}{k}\sum_{l\in\mathcal{L}} EvalMet(TP_l, FP_l, TN_l, FN_l) \tag{10}$$

$$Micro\ metric = EvalMet\left(\sum_{l\in\mathcal{L}} TP_l, \sum_{l\in\mathcal{L}} FP_l, \sum_{l\in\mathcal{L}} TN_l, \sum_{l\in\mathcal{L}} FN_l\right) \tag{11}$$

The third main group of multi-label evaluation metrics is aimed to work over label rankings, instead of bipartitions. *One error, Ranking loss, Coverage* and *Average precision* are among the best known metrics in this category. These metrics usually check if a true relevant label is in the ranking produced by the algorithm, the number of steps to walk until a relevant label is found in the ranking, or whether a non-relevant label has been ranked above a relevant one. They can be computed even when the used algorithm produces a bipartition instead of a ranking, by relying in some other kind of real value as can be the confidence or a set of weights in a neural network.

Aside from these three main groups, some more specific multi-label metrics have been defined to evaluate hierarchical [24] multi-label classification or the quality of multi-label clustering [21].

3. Conducting multi-label learning experiments

Let us assume we are designing a new MLC algorithm aimed to improve the results produced by existing ones. Aside from explaining the theoretical hypotheses underlying the new proposal, stating why it should perform better, we also commit to providing real evidence of this enhancement. Therefore, an empirical study has to be conducted.

The main steps usually followed to carry out a data mining experiment are the four ones previously enumerated, also depicted in Fig. 2. Here we are delving into some specific aspects. Later, we will put the focus on the usual traps and on how to surpass them.

3.1. Data selection and preparation

The first step in this process is usually the selection of datasets to be used in experiments. The criteria employed to select the data

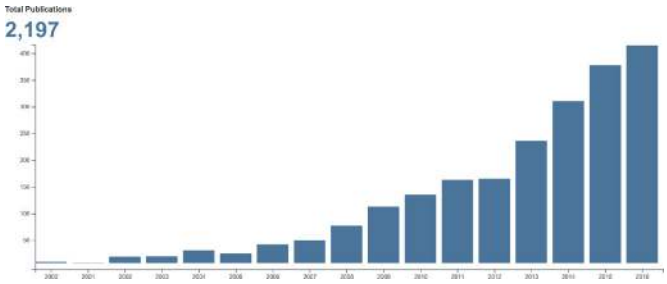


Fig. 1. Number of published articles dealing with MLC 2000–2016 (source: Web of science).

that will be involved in the experimentation may vary. In the MLC context aspects such as the number of distinct labels, label cardinality and density, imbalance levels and label concurrence, among others, are usually taken into consideration.

Depending on the case, these data can be either real or synthetic. The use of data collected from actual sources has multiple advantages. In this way, the tested algorithm will be exposed to the characteristics and complexities of real-world information, in the same context in which it is presumably intended to be used. However, sometimes data of this kind do not meet the needs of ongoing experimentation or do not fit the specific traits required. In these cases specialized tools, capable of generating data that match the required characteristics, tend to be the solution.

Once the datasets to be included in the experimental study have been collected, it is necessary to apply the preprocessing steps that are considered appropriate. For instance, a data transformation technique such as label powerset or binary relevance, would allow to apply a multiclass or binary classifier. Moreover, training and test partitions have to be extracted from the original MLD. Sometimes, depending on the algorithm, the training set can be also divided into two subsets, using one of them to train the model and the other to validate its parameters.

3.2. Competing methods

One of the objectives of proposing a new learning method is to improve the results of others. Although sometimes the algorithm presented may be completely new, in most cases it will be an improvement over a method already in use. In any case, it is to be assumed that the authors have certain knowledge of the field under study and, in particular, of methods similar to their own.

A large portion of MLC algorithms are based on transformation techniques, such as binarization [6] and label powerset [7]. Those allow to transform a complex problem (MLC) into several easier ones, but at the expense of increasing the computational costs consumed to process them. Therefore, running a large set of MLC methods over several datasets can be very time consuming, thus the importance of choosing the proper ones.

The alternative to the previous transformation techniques are classification methods adapted to deal with raw multi-label data. A plethora of proposals have been presented in this field, including multi-label decision trees [3], instance-based classifiers [25], neural networks [4] and support vector machines [5], among other adaptations of traditional classification methods.

Deep learning techniques have also found their niche in the classification field. Different deep neural networks architectures [26] have been proposed in late years, and several of them, including DBNs (*Deep-Belief Networks*) and CNNs (*Convolutional Neural Networks*) have been proposed to classify multi-label data [27–29]. The superior performance of most deep learning methods is due to the integrated feature learning phase, able to extract a reduced set of new, more informative features. This task can be conducted as a preprocessing phase, for instance by relying on autoencoders [30], then applying any multi-label classifier over the reduced feature set.

Once the methods that are going to compete with the new one are selected, attention must be paid to the configuration parameters present in each one of them. These should adjust to the recommendations given by their authors, otherwise their behavior could be unexpected. This is important only if we intend to run such algorithms, rather than take the published results. However, in the latter case, other aspects need to be taken into account as indicated below.

3.3. Performance metrics

The selection of performance metrics is another key factor in the experimental process, specially in the MLC field. Unlike traditional classification, where usually a pair of performance indicators such as precision or accuracy tend to be enough, more than twenty evaluation measures are of common use in MLC. That is why picking the right ones for each case can usually have a large impact in the final conclusions.

As the compilation of performance metrics provided in [1] shows, the assessment of successes and failures can originate in a bipartition matrix or in a ranking. Moreover, the number of

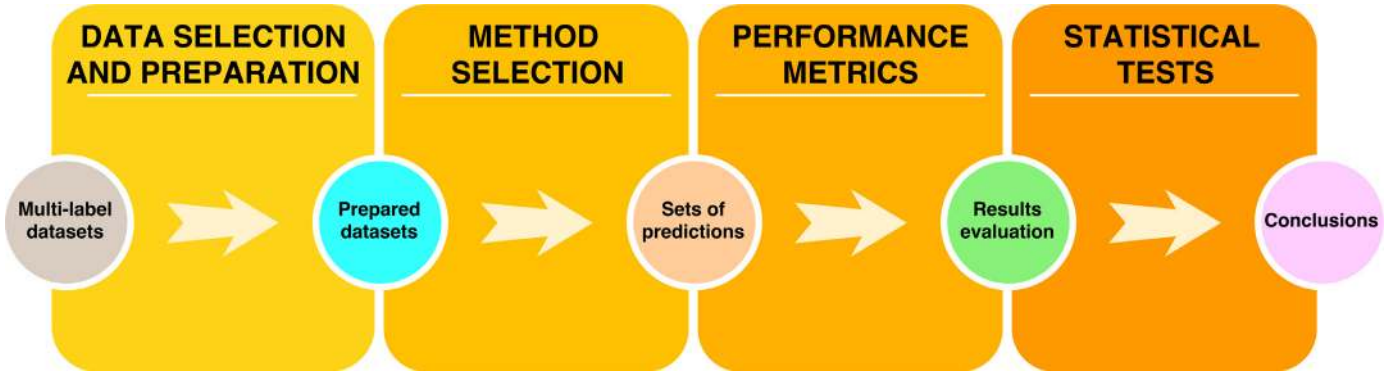


Fig. 2. Steps in a typical MLC experiment. (1) Datasets have to be selected from the available repositories, taking into account their traits in order to choose the more appropriate ones for the task at hand. Some data preparation could be needed, depending on possible data peculiarities (i.e. an imbalanced dataset could be balanced through a resampling algorithm). (2) Methods related to the proposed one have to be selected and run, obtaining a set of predictions from each one of them. (3) These predictions have to be evaluated, picking from the large set of available performance metrics those more adequate to the analyzed problem. (4) It is advisable to draw conclusions including statistical tests over the previous results.

hits and misses can be aggregated following different strategies, by label or by sample.

It is obvious that the same performance metrics must be obtained for all the methods involved in the experimentation, otherwise it would not be possible to make a correct evaluation of the results. In addition to performing a detailed comparison, method against method and measure by measure, it is also necessary to obtain an overall appraisal for these comparison results. To accomplish this task it is usual to rely on the proper statistical tests.

4. Tips and pitfalls while performing multi-label experiments

Any new algorithm proposal should begin with a review of existing literature on related methods, detailing their similarities, strengths and weaknesses, etc. Authors should establish the niche they intend to occupy with their proposal or the direction in which they want to improve the methods already published. Hereafter, the experimentation is usually carried out.

Most of the mistakes made during experimentation involve the data used and how they are treated. You must select the appropriate datasets for the task at hand and the proposed algorithm type. In addition, such data should be prepared in line with the experiments already published on related methods. Occasionally there are obstacles that hinder the correct use and preparation of the data. This is why in the following sections we focus mainly on this aspect, and why in Sections 4 and 5 we present tools to overcome these obstacles.

Logically, the rest of aspects mentioned above, such as the selection of methods to be compared against the proposal, the set of metrics used to evaluate their performance and the usage of statistical tests, are also important. They will therefore be addressed later, albeit more briefly.

4.1. Selecting the proper datasets

While designing a new algorithm, sometimes the aim is to make it a general purpose method, with the goal of improving prediction in a broad way. However, in other cases the proposal is more specific. In the multi-label field, it is common to present methods tailored for dealing with large quantities of labels, with data showing imbalance among labels, with missing labels, etc. Each scenario requires a specific type of dataset to perform the experiments.

The basic principle must be that the data used in experimentation should present the problem for which the algorithm is proposed. This, however, is not always the case. There are authors who propose a new method to deal with large sets of labels, but in their experimentation they use MLDs that only have a few dozen of them. Logically, to support the behavior of the proposed method, it must be demonstrated that it works correctly with the right configuration: against datasets having hundreds or thousands of labels. This scenario can be extrapolated to similar ones: if a method attempts to address multi-label imbalance, the selected datasets should present this problem (very common in this field, on the other hand); there would be no point in experimenting using only balanced datasets.

When choosing datasets, it would be advisable to observe the following recommendations:

- Begin by performing an exploration of the characteristics of the available datasets, obtaining metrics that allow you to know the number of labels, cardinality, degree of imbalance, level of concurrence between labels, etc. All this information is vital to be able to select those sets that best suit your needs.
- Which datasets have other authors used to address the same task? By answering this question, a list of datasets commonly included in similar experiments can be obtained. It is a work that can be done at the same time as reviewing related works. It will also make it easier to compare results with previously published methods.

If the new method is presented as a generic multi-label classifier, datasets with characteristics as diverse as possible should be included in the experimentation. This approach will make it possible to identify the strengths and weaknesses of the proposal, key aspects when comparing it with existing methods. Again, the selection of such datasets should be based on an exploration of the characteristics of all publicly available ones.

4.2. Preparation of data and methods

The main objective of a multi-label experimentation is to compare a new algorithm with existing ones. For this, it is essential that the same training data are used in all cases, since the model obtained depends fundamentally on this condition. Aspects such as the rate of committed errors, the classifier's bias towards certain classes, its ability to generalize, etc., are highly influenced by the samples used during training.

During our years in the MLC research field we have had the opportunity to find, in relation to data preparation and processing, the following mistakes:

- Comparing the results obtained by running the proposed algorithm with those previously published for other methods. Unless exactly the same datasets have been used in the experimentation, with exactly the same pre-processing and partitioning strategy, such comparison will not be valid. Taking the performance measures published in an article for a certain method is highly convenient, but if you train that same method using unidentical partitions for training and testing the results will differ.
- Delivering complete datasets to each method and allowing them to be partitioned internally. Many tools automate random partitioning when evaluating an algorithm. However, in this situation, the exact same training partitions would not be used to generate the models, which would induce dissimilarities that benefit some and harm others.
- Running the algorithms to be compared but using different datasets in some cases. For example, using hold-out with certain methods because they are slower and cross validation with others, or reducing the set of input attributes for some methods and not for others. Obviously in these cases the authors are artificially favoring some algorithms over others.

For a fair comparison of several classifiers, it is therefore essential to train the models with exactly the same data samples. Starting from this premise, from our point of view there are the following alternatives:

- If you want to compare one algorithm with the published results of another, without the need to run the latter, it is essential to have the data partitions used by its authors. This is only possible if, together with the results of such method, these partitions are also made publicly available. This approach is complicated as soon as it becomes necessary to compare with two or more already published methods, since different authors will have used disparate data partitions. In this situation the new algorithm should be run with the appropriate data for each case and pairwise comparisons should be made. In our opinion it is the least advisable alternative. It should only be used when it is impossible to run an existing method, but the data used to evaluate its performance is available.
- In any other case, without having the data originally used by other authors, the procedure to follow is always the same.

The data must first be prepared, including partitioning, so that all algorithms to be tested receive the same set of training and testing samples. Then, the code of all methods compared should be obtained, preferably from their authors. If this is not possible, they should be implemented as accurately as possible from the article in which they are described. Finally, all the algorithms must be executed using the same data and obtaining the corresponding predictions.

Trying to follow these instructions can easily lead to a number of obstacles. If you have access to the original data, its format may not be appropriate for the tools used to implement the new algorithm. In fact, almost every time several methods are going to be run, coming from different authors, there is diversity in file formats. These are the kind of issues the tools described in Sections 5 and 6 strive to overcome.

4.3. Assessing algorithm performance

As stated in Section 3.3, a multi-label classifier can be evaluated using a wide range of performance metrics. Each of them offers a different quality indicator, so it is essential to use a good set of them to obtain the most balanced possible assessment.

If the experimentation includes results taken directly from previous publications, and bearing in mind the assumptions indicated in the previous sections, then the selection of measures will be given to us. It will be necessary to use the metrics already used in these publications, not others. The results obtained by the proposed method should therefore be used to calculate these measurements, thus allowing direct comparison with the algorithms already published.

Even if we have exactly the same data partitions to train our model, as explained above, and calculate the same set of measures, the comparison with other algorithms may not be completely fair. The calculation of some of the multi-label performance metrics is relatively complex, and there may be differences in the way they are computed between different multi-label software packages. Consequently, the only way to be absolutely certain that the results are comparable would be to ensure that the computation of the measures is also carried out in the same way.

If we are going to run all the algorithms ourselves, we will have total freedom in choosing the performance metrics. In this case, as recommended in [31], we should select a broad set of evaluation metrics, including both sample-based and label-based measures, as well as measures calculated on bipartite and label rankings. In particular, it is advisable to include at least one or two metrics from each of the following groups:

Sample-based metrics based on bipartition results. As explained in Section 2, these metrics are computed sample by sample and then an average evaluation is obtained. *Hamming loss* is maybe the most common metric in this group; *Accuracy, Precision, Recall* and *F-measure* being popular as well. Most of them are obtained from the confusion matrix for each label. We recommend including always *Hamming loss* and *F-measure*, since the former one is the complement of *Accuracy* and the latter is the harmonic mean of *Precision* and *Recall*.

Metrics based on ranking results. Although many algorithms produce a bipartition as result, stating which labels are predicted for the data patterns, many others provide a label ranking. In this case, the set of predicted labels is obtained after applying a certain threshold over this ranking. Ranking-based metrics evaluate the performance operating with a label ranking, being *One error, Ranking loss* and *Average precision* among the most frequently used.

Label-based metrics. Most metrics obtained from a confusion matrix, such as *Precision* and *Recall*, can be computed by label instead of by sample, as described in Section 2. Since two averaging strategies exist, named *macro-averaging* and *micro-averaging*, two measurements can be retrieved for most of these metrics. In our opinion, *Macro F-measure* and *Micro F-measure* should be included in most evaluations as they provide two additional views on method performance.

The selection of evaluation metrics may be influenced if the proposed algorithm addresses a specific problem. For example, if the analysis corresponds to methods for working with imbalanced data, metrics such as *AUC* or *Macro F-measure* would be more appropriate than *Hamming loss* or *Accuracy*, as they are less biased towards majority labels.

4.4. Reaching global conclusions

When an experimental study is conducted involving multiple datasets, several algorithms and various performance metrics, the number of indicators to be evaluated is so large that it may be difficult to reach an overall conclusion. Usually, a classifier will be better than others while working with certain MLDs, but its behavior will worsen with other datasets. The same is applicable to the use of several evaluation metrics.

A first approach to a global assessment could be to count the number of times each algorithm wins or loses against the rest. From here, a ranking to determine the position of each method, usually grouped by performance metric, is almost immediate. In our opinion this approach is better than others we have found sometimes in the literature, such as averaging all the results from each algorithm. One of them can be the best one when evaluated with a certain dataset or metric, but perform horribly in all other cases. The authors should strive to show a realistic perspective of its behavior, for instance by means of a ranking instead of an average, which would hide it.

The conclusions of an experimental analysis can be reinforced by using more formal procedures, carrying out the relevant statistical tests. In most cases it is not possible to guarantee conditions of normality (the results to be evaluated following a normal distribution) and homoscedasticity (the variance being homogeneous). For this reason, non-parametric statistical tests are commonly used. Depending on the number of methods to be compared, and if such comparison is pairwise or multiple, the proper tests have to be chosen as explained in [32,33].

4.5. Summary, advantages and disadvantages

To make it easier to follow up on previous advice, it has been summarized in Table 1. The first column indicates the stage of the process, second the trap to avoid and third one what to do instead. In our opinion, adhering to these recommendations would bring several advantages to any MLL study:

- The selection of the proper datasets, those that presumably present the problem that the proposed method aims to solve, for instance a high imbalance level, will back the behavior of the method.
- Correctly comparing our results with published ones, either by using the same data partitions or by running the other methods with our data, will make the study more solid.
- Choosing a good set of evaluation metrics, specially those designed to expose strengths and weaknesses of a method while facing specific problems, will be more convincing to other researchers. This confidence in the results can be increased if the appropriate statistical tests are conducted.

Table 1
General tips on how to avoid common mistakes during an experimentation in the multi-label field.

Stage	Avoid...	Instead, ...
Data selection	using datasets which not present the tackled problem.	explore the characteristics of available datasets for proper selection.
	skipping popular datasets for the task.	review related works and look for recurring datasets.
Method preparation	comparing your results to previously published ones.	perform new runs of each competing method under same conditions.
	partitioning datasets differently for each method.	build partitions prior to running methods.
	performing different validations for each method.	determine the validation strategy taking into account possible slow methods.
Algorithm assessment	choosing metrics that dismiss the tackled problem.	select those which are affected by it.
Conclusions	hiding the behavior of a method behind average values.	rank methods according to their performance.
	performing statistical tests without guaranteeing their assumptions.	possibly compare methods with non-parametric statistical tests.

Faced with these benefits, the main disadvantage in following these tips is the increased amount of work to be done. Time has to be devoted to explore dataset traits, in order to choose the best ones, as well as to obtain data partitions and other methods implementations. Moreover, finding the tools to tackle all this work is not always easy. Aiming to mitigate this last obstacle, the tools we have developed to overcome these tasks are introduced in the following sections.

5. mldr.datasets: the tool for managing multi-label datasets

As the previous section has shown, selecting and preparing datasets to be used for experimentation are essential steps. However, when it comes to obtaining the appropriate datasets, partitioning them and obtaining them in the right format for each learning algorithm, multiple obstacles arise. It would be desirable to have a tool that facilitates such operations, as well as easing the exploration of its characteristics, being able to provide the reference for each dataset, etc. That is the motivation behind the development of the `mldr.datasets` package.

The first version of this software package for R users was introduced in [34]. Since then, its functionality has been extended by including new multi-label partitioning algorithms, new functions to export the data to disparate formats, automatic checking for data sparsity, etc. The main goal has been to facilitate all the tasks needed to select and prepare MLDs for conducting a experimentation.

This section provides a didactic description of the above mentioned package, explaining how to complete each of the tasks from obtaining a set of data to its exploration, documentation, partitioning and export.

5.1. Installing mldr.datasets in our computer

The `mldr.datasets` software is an R package. As a consequence, anyone interested in using it needs to have the R interpreter [35] installed in their computer. Assuming that this is the case, the installation procedure is the same followed for any package available in CRAN (*Comprehensive R Archive Network*), to issue the following command at the R console:

```
> install.packages(''mldr.datasets'')
```

This will install the last stable version of the package, which is 0.4 at this time. Development versions, with added functionality, are available in GitHub[1]. Assuming that the `devtools` package [36] is installed and loaded, the most recent version of `mldr.datasets` can be always installed from GitHub as shown below:

```
> install_github(''fcharte/mldr.datasets'')
```

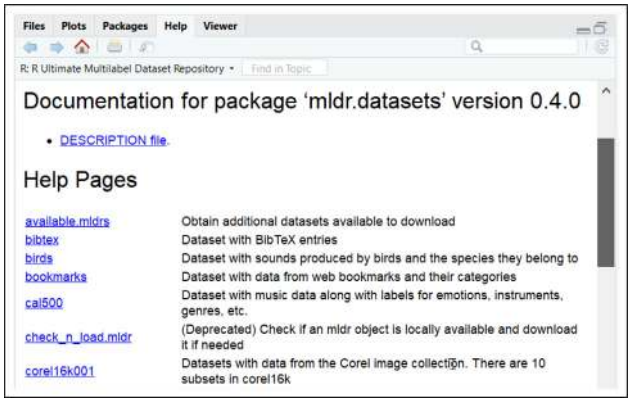


Fig. 3. Help index of the mldr.datasets package.

Once installed, the package has to be loaded into memory each time a new R session is started. This can be done with the usual `library()` or `require()` R commands. Since this moment the user can access the functions provided by the package. The index of all available functions (partially shown in Fig. 3) can be retrieved with the following command:

```
> help(package=''mldr.datasets'')
```

If you need help with a particular function, you can click on its name in the previous index. You can also use the command `help(''function.name'')` or, if you have already entered the name of the function into the R console or editor, you can prepend a question mark to it. In all cases, a description of the function and its parameters will be obtained, as shown in Fig. 4.

The subsequent sections introduce most of the available functions in the `mldr.datasets` package, showing how they can be used to perform each type of task.

5.2. How to load and import multi-label datasets

The package includes not only the functions mentioned below, able to import MLDs from a web repository, but also a set of 10 already integrated MLDs. They are available immediately, as soon as `mldr.datasets` is loaded into memory. These MLDs are birds [37], cal500 [38], emotions [39], flags [40], genbase [41], langlog [42], medical [43], ng20 [44], slashdot [14] and stackex_chess [45]. The following command can be entered into the R console to obtain a list of built-in datasets:

```
> data(package=''mldr.datasets'')
```

Each dataset is an R object, specifically an object of class `mldr`. This is the format defined in the homonymous package[2] [46] which, among other functionality, facilitates the reading of

[1] The source code of the package is publicly available at github.com/fcharte/mldr.datasets.

[2] The mldr package establishes the format of MLDs in R. It provides a user interface to ease the exploratory analysis and also performs data transformations, such

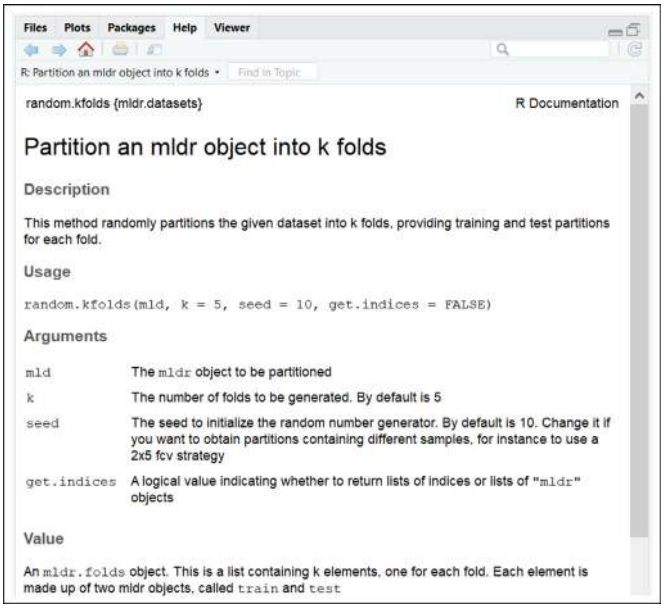


Fig. 4. Help about the random.kfolds() function.

multi-label data in various formats and a user interface to carry out exploratory analysis of MLDs. These are implementation details that are not essential for the regular user of `mldr.datasets`. You can access these datasets simply by entering their name in the console, as you would with any R object.

There are many other MLDs available online. Most of them are not embedded into the package, but can be downloaded and saved locally by means of the following two functions:

- `available.mldrs()`: Retrieves the most up to date list of additional datasets from the Internet. This function does not need parameters. It returns as result an R `data.frame` containing the name and description, among other details, about available MLDs.
- `get.mldr()`: Loads any of the available datasets into memory, downloading it from the Internet if it were necessary. Once loaded, users will be able to work with it as they would with any of the already built-in MLDs.

Below is an explanation on how to use these functions to complete each of the tasks associated with loading and downloading MLDs.

5.2.1. Browsing the available datasets

In addition to the 10 MLDs already integrated in the package, a much larger number is available online. The list of datasets is maintained and updated independently of the `mldr.datasets` software, so it can be extended in the future. The `available.mldrs()` function is in charge of obtaining the most recent list of online MLDs, returning it as a `data.frame` object.

A `data.frame` is a data structure made up of several rows (records) and columns (fields). In this case each row contains details of a dataset, while the columns provide the following data:

- `Name`: The name of the MLD. The usual denomination found in the literature is used to refer to each MLD. This is the name to be given as input to the `get.mldr()` function explained below.
- `Description`: A brief description of the MLD's origin and/or nature.
- `Instances`: Number of data instances in the MLD.
- `Attributes`: Number of input attributes (features) in the MLD.
- `Labels`: Number of output attributes (labels) in the MLD.
- `URL`: Full URL from which the MLD can be downloaded. It is the address automatically used by the `get.mldr()` function to download a dataset when needed.

The result returned by the `available.mldrs()` function can be treated like any other `data.frame` in R. Fig. 5 shows how to get its structure, with the `str()` command (upper part), and how to recover the name and URL from some of the 60 MLDs initially available.

5.2.2. Setting the download directory

Before downloading any MLDs, it is possible to set the directory in which they will be stored locally. For this purpose, the environment option `mldr.download.dir` is used, whose value indicates the absolute or relative path of the desired directory. The aim is to provide a means for each user to save the downloaded MLDs where they want, so that they can load them later into memory without having to retrieve them again from the Internet.

The `options()` command from R allows to set the value of `mldr.download.dir`. It gets one parameter, indicating the name of the variable and the value to be assigned. The bottom of Fig. 6 shows how to do this, as well as how to check the current value through the `getOption()` command.

If a value for the previous variable is not set, and the user does not specify a download directory when invoking the `get.mldr()` function, the default download path will be used. This corresponds to a subdirectory `.mldr/datasets` that will be created in the home folder of the current user. In the case of GNU/Linux the path is usually `/home/user`, while in Windows it would correspond to the user's documents folder, as shown in the upper part of Fig. 6.

5.2.3. Downloading new datasets

In order to download an MLD you first need to know its name. It can be retrieved from the `Name` column of the `data.frame` returned by `available.mldrs()`. This name is the only parameter required to invoke the `get.mldr()` function. Optionally, the directory where you want to store the downloaded file can be specified by means of the `download.dir` parameter. If this argument is not provided by the user, the directory indicated by the option `mldr.download.dir`, as described in the previous section, will be used.

The `get.mldr()` function works in three steps:

1. It starts by determining the download directory. If the parameter `download.dir` has been provided by the user, this path is used, otherwise the option `mldr.download.dir` is checked. If it has been previously set, this directory is used, resorting to the default path otherwise.
2. The next step is to check whether the requested MLD is already in the download directory. If this is the case, simply skip to the next step. Otherwise, it is downloaded and stored locally.
3. Finally, the requested dataset is loaded into memory and returned as a result. This will be an object of class `mldr`, as explained above. It can then be used like any of the built-in MLDs.

Fig. 7 shows how to set the download folder, retrieve the names of all MLDs available online, and download one of them. Once the transfer is completed, the dataset is loaded into memory and stored in the `imdb` variable.

As a shortcut, in `mldr.datasets` there are individual functions defined to make it easier to download each of the MLDs

as binarization and label powerset, among other functions. Please refer to [46] for an extended description.

```
> str(available.mldrs())
'data.frame':   51 obs. of  6 variables:
 $ Name       : chr  "bibtex" "bookmarks" "corel16k001" "corel16k002" ...
 $ Description: chr  "Dataset with BibTeX entries" "Dataset with data from web bookmarks and their ca
tegories" "Datasets with data from the Corel image collection. There are 10 subsets in corel16k" "Dat
asets with data from the Corel image collection. There are 10 subsets in corel16k" ...
 $ Instances  : int  7395 87856 13766 13761 13760 13837 13847 13859 13915 13864 ...
 $ Attributes : int  1836 2100 500 500 500 500 500 500 500 500 ...
 $ Labels     : int  159 208 153 164 154 162 160 162 174 168 ...
 $ URL        : chr  "https://cometa.ml/public/full/bibtex.rds" "https://cometa.ml/public/full/bookma
rks.rds" "https://cometa.ml/public/full/corel16k001.rds" "https://cometa.ml/public/full/corel16k002.r
ds" ...
> available.mldrs()[18:25,c("Name", "URL")]
           Name                                          URL
18 eurlexev_test https://cometa.ml/public/full/eurlexev_test.rds
19  eurlexev_tra  https://cometa.ml/public/full/eurlexev_tra.rds
20 eurlexsm_test https://cometa.ml/public/full/eurlexsm_test.rds
21  eurlexsm_tra  https://cometa.ml/public/full/eurlexsm_tra.rds
22          imdb          https://cometa.ml/public/full/imdb.rds
23     mediamill     https://cometa.ml/public/full/mediamill.rds
24   nuswide_BoW   https://cometa.ml/public/full/nuswide_BoW.rds
25  nuswide_VLAD  https://cometa.ml/public/full/nuswide_VLAD.rds
```

Fig. 5. Browsing the available MLDs.

```
> file.path(normalizePath("~"), ".mldr", "datasets")
[1] "C:\\Users\\Francisco\\Documents/.mldr/datasets"
>
>
>
>
> options(mldr.download.dir = "/home/fcharte/mymlds")
>
> getOption("mldr.download.dir")
[1] "/home/fcharte/mymlds"
>
>
```

Fig. 6. Setting the directory where new MLDs will be stored.

available online. For example, to download the MLD `tmc2007` we can use either of the following two commands in the R console, obtaining exactly the same result.

```
> tmc2007 <- get.mldr(''tmc2007'')
> tmc2007 <- tmc2007()
```

The operations described in the next sections are available for any `mldr` class object, be it a dataset obtained with `mldr.datasets`, generated by the `mldr` package or by any other software that produces this object format.

5.3. Obtaining descriptive meta-data

All `mldr` objects have several fields which provide meta-data about the MLD they contain. The name and information on these fields is as follows:

- `name`: Contains the original name of the MLD. Sometimes this name does not coincide with the usual name given to the dataset.
- `dataset`: A `data.frame` holding the actual data samples of the MLD.
- `attributes` and `attributesIndexes`: The former is a vector with all the MLD's attributes, including input features and output labels. For each attribute its name and data type is provided. The latter states the numeric index of input features in the MLD, since labels can be located at the beginning or at the end of the dataset.
- `labels`: It is a `data.frame` object with details about each label in the MLD (see example in Fig. 8), such as its name, number of occurrences, frequency, and the *IRLbl* [11], *SCUMBLE* and *SCUMBLE.CV* [12] metrics.
- `labelsets`: A vector containing each label combination (*labelset*) appearing in the MLD along with their number of occurrences.
- `measures`: This list provides a set of measures aimed to characterize the MLD. When the `mldr` package is loaded, the user can retrieve this list by means of the standard `summary()` command, as shown in Fig. 9. They can also be retrieved individually through the syntax `mld$measures$measureName`.
- `bibtex`: Holds the BibTeX entry needed to reference the source the MLD is coming from. This information can be also retrieved with the `toBibtex()` function. The returned string is ready to be copied to the clipboard, but can also be printed into the R console using the `cat()` command (see Fig. 10).

The `measures` list is the tool aimed to ease the selection of the proper MLDs. Most of them were described in Section 2. It contains 13 different metrics:

- `num.attributes`, `num.inputs` and `num.labels`: The total count of attributes in the dataset, how many of them are input features and how many output labels, respectively. The first field always is the sum of the other two.
- `num.instances`: The number of instances in the dataset.
- `num.labelsets` and `num.single.labelsets`: The former indicates how distinct labelsets appear in the MLD, while the latter states how many of them appear only once.
- `max.frequency`: Provides the number of times that the most common labelset occurs in the MLD.
- `cardinality` and `density`: These fields contain the measures known as label cardinality (*Card*) and label density (*Dens*).
- `meanIR`: Holds the *meanIR* measure. The *IRLbl* metric is stored in the homonymous column of the `labels` field described above.

```
Console  Terminal
D:/FCharte/Estudios/
>
> options(mldr.download.dir = "D:/FCharte/Estudios")
> available.mldrs()$Name
 [1] "bibtex"               "bookmarks"            "corel16k001"          "corel16k002"
 [5] "corel16k003"          "corel16k004"          "corel16k005"          "corel16k006"
 [9] "corel16k007"          "corel16k008"          "corel16k009"          "corel16k010"
[13] "corel5k"              "delicious"            "enron"                "eurlexdc_test"
[17] "eurlexdc_tra"         "eurlexev_test"        "eurlexev_tra"         "eurlexsm_test"
[21] "eurlexsm_tra"         "imdb"                 "mediamill"            "nuswide_BoW"
[25] "nuswide_VLAD"         "ohsumed"              "rcv1sub1"             "rcv1sub2"
[29] "rcv1sub3"             "rcv1sub4"             "rcv1sub5"             "reutersk500"
[33] "stackex_chemistry"    "stackex_coffee"       "stackex_cooking"      "stackex_cs"
[37] "stackex_philosophy"   "tmc2007"              "tmc2007_500"          "yahoo_arts"
[41] "yahoo_business"       "yahoo_computers"      "yahoo_education"      "yahoo_entertainment"
[45] "yahoo_health"         "yahoo_recreation"     "yahoo_reference"      "yahoo_science"
[49] "yahoo_social"         "yahoo_society"        "yeast"
>
> imdb <- get.mldr("imdb")
Looking for dataset imdb in the download directory
Looking for dataset imdb online...
Downloading dataset imdb
trying URL 'https://cometa.ml/public/full/imdb.rds'
Content type 'application/octet-stream; charset=utf-8' length 5560440 bytes (5.3 MB)
downloaded 5.3 MB
```

Fig. 7. Downloading a new MLD and loading it into memory.

```
Console  Terminal
D:/FCharte/Estudios/
>
>
> imdb$labels
           index count        freq      IRLbl    SCUMBLE SCUMBLE.CV
Sci-Fi         1  5488 0.045385754   7.977223 0.13456030  0.8884458
Crime          2  9031 0.074686360   4.847636 0.18261815  0.7460513
Romance        3 12379 0.102374317   3.536554 0.17687118  0.6432376
Animation      4  7458 0.061677652   5.870072 0.17700698  0.5235917
Music          5  3687 0.030491486  11.873881 0.25892907  0.6021957
Comedy         6 31216 0.258156286   1.402454 0.12208385  1.2031447
War            7  3297 0.027266186  13.278435 0.35033939  0.3725959
Horror         8  8044 0.066523871   5.442442 0.09753873  1.0352863
Film-Noir      9   411 0.003398970 106.518248 0.74167975  0.1397466
Adventure     10  8037 0.066465981   5.447182 0.12229903  0.9161755
News          11   499 0.004126729  87.733467 0.53074149  0.5453036
Western       12  4190 0.034651295  10.448449 0.10298233  1.3460870
Thriller      13 11623 0.096122198   3.766583 0.11872109  0.9655549
Adult         14  1386 0.011462219  31.586580 0.18934304  1.3377658
Mystery       15  5204 0.043037074   8.412567 0.18637228  0.7366114
Short         16 32423 0.268138175   1.350245 0.11843045  1.2497104
Talk-Show     17   290 0.002398300 150.962069 0.28007742  1.2436135
```

Fig. 8. List of labels in an MLD.

- `scumble` and `scumble.cv`: Provide the *SCUMBLE* and *SCUMBLE.CV* metrics for the dataset.
- `tcs`: Holds the *TCS* metric, a evaluation of the theoretical complexity of the dataset.

All these metrics are automatically computed each time a new `mldr` object is created, for instance by means of the loading functions provided by the `mldr` R package. Jointly, these measures would ease for a practitioner the selection of the proper MLDs to be included in a experimental study.

5.4. Partitioning the datasets

Once the proper set of MLDs has been selected, the next step usually consists in partitioning them so that some samples are used to train a model, while the remaining ones allow to test its performance. The `mldr.datasets` package provides us with the functions needed to accomplish this task. Three different partitioning strategies can be applied:

- Random: Randomly separates the data samples into a certain number of partitions. As a result, the number of patterns for each label can be distributed non-uniformly. In some extreme cases, this strategy could gather all the instances for a label in the same partition.
- Stratified: Follows the stratified algorithm described in [13] in an attempt to distribute the data samples as evenly as possible among different partitions.
- Iterative stratification: The partitioning method introduced in [47] has the same goal of the previous one, but it tackles the problem iteratively.

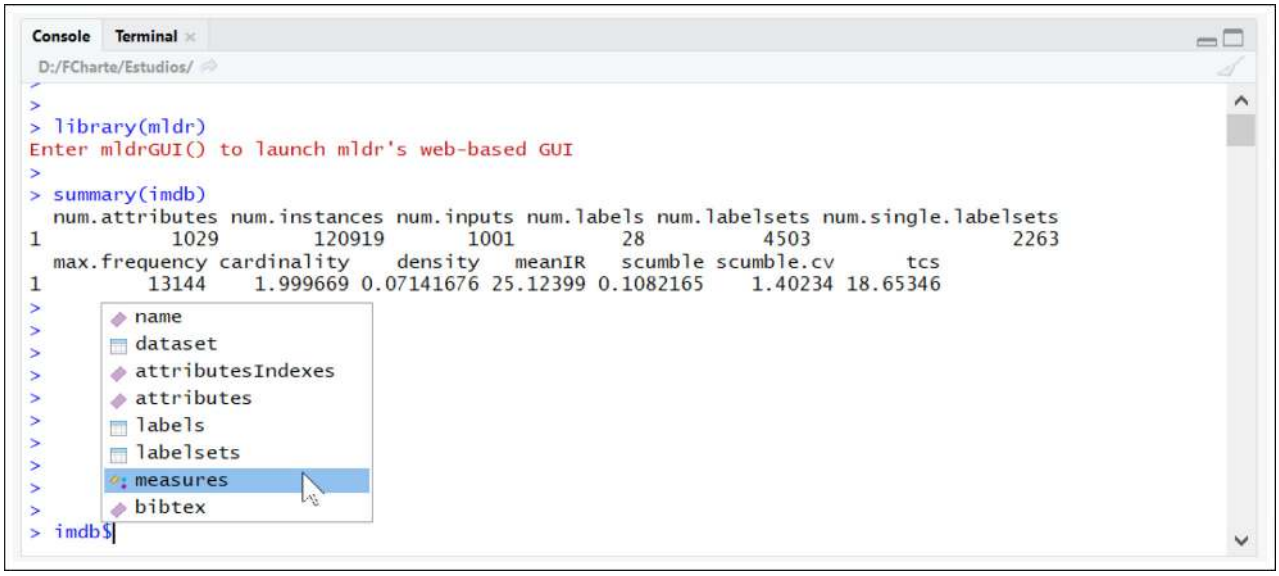


Fig. 9. Exploring the MLD traits.

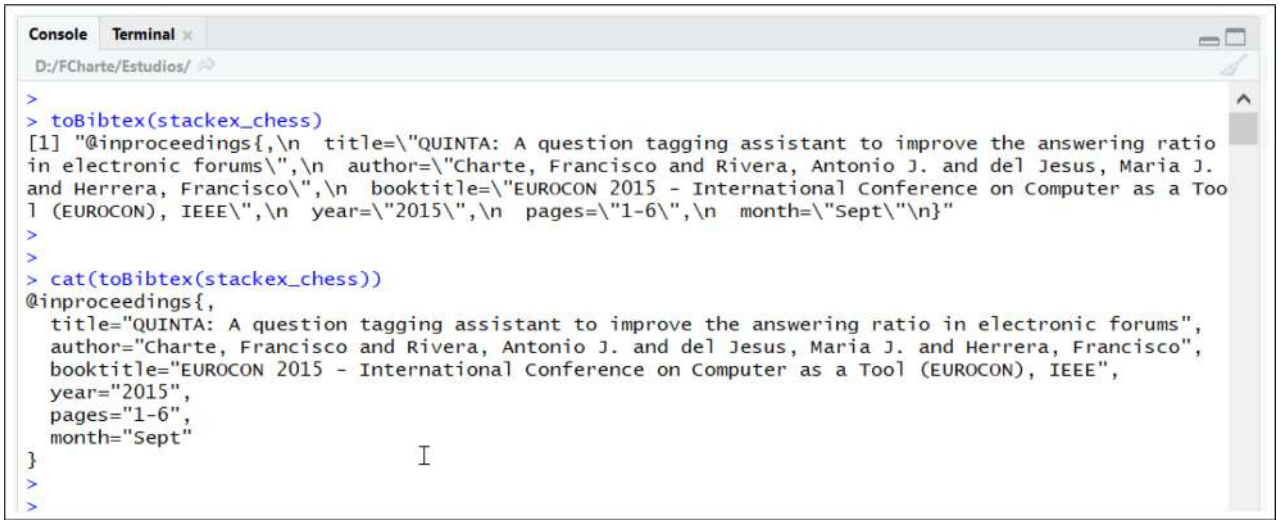


Fig. 10. Getting bibliographic data to cite a dataset.

For each one of these strategies, there exist three functions in the package. The first and more generic family of functions allows to create any amount of partitions with a given distribution of instances. The second group builds two partitions in a hold-out manner, one for training and one for test. The last family of functions generates partitions oriented to performing k-fold cross validation. In total, users can access the following 9 functions:

- `random.partitions`
- `stratified.partitions`
- `iterative.stratification.partitions`
- `random.holdout`
- `stratified.holdout`
- `iterative.stratification.holdout`
- `random.kfolds`
- `stratified.kfolds`
- `iterative.stratification.kfolds`

Regardless of which of the previous methods we use, the parameters to be given to the respective function are the same in all cases:

- `mld`: The only compulsory argument is the `mldr` object containing the MLD to be partitioned.
- `r` (only for `partitions` functions): A vector indicating the percentages of instances desired in each partition. For example, a value of `c(35, 25, 40)` would indicate three partitions, with 35%, 25% and 40% of instances respectively.
- `p` (only for `holdout` functions): Indicates the desired percentage of instances in the training subset. It has a default value of 60.
- `k` (only for `kfolds` functions): Indicates the desired number of folds. By default it is 5, so five different folds of training/testing samples would be produced.
- `seed`: The seed used to initialize the random generator. Its default value is 10. It should be changed if we want to obtain different sets of folds, for instance to generate two sets of 5 folds (2x5fcv).
- `get.indices`: By default the partitioning functions generate a list with as many elements as partitions requested. Each element will be of class `mldr` by default for the generic and hold-out function; in the case of k-folds functions, each element will consist of two `mldr` objects, one for training and one for test-

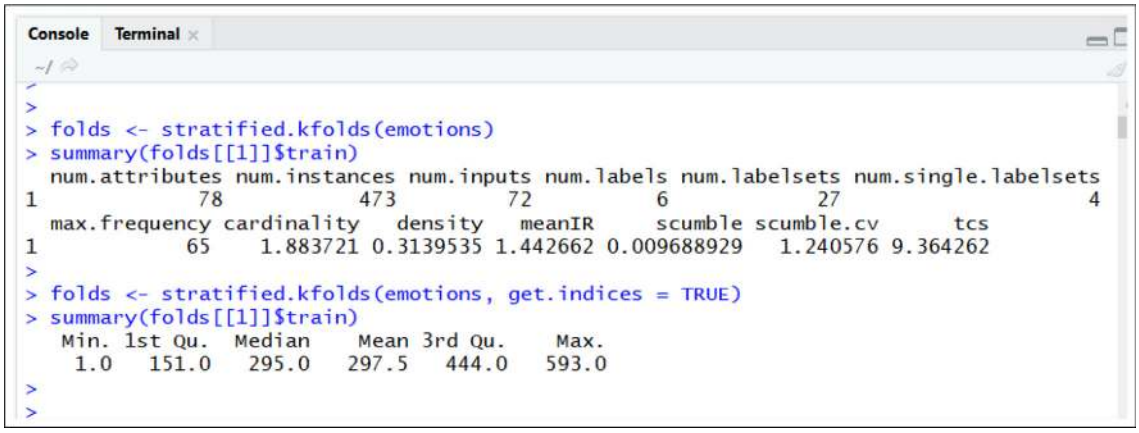


Fig. 11. Partitioning a dataset obtaining a list of mldr object (above) or a list of samples' indexes.

ing. If we assign the TRUE value to this parameter, lists with the indexes of the samples will be provided instead of generating `mldr` objects with the data. These indexes can be used over the original MLD to select instances, taking up much less memory space than several `mldr` objects.

The example shown in Fig. 11 partitions the same dataset twice. Firstly, a list of five objects is obtained holding two members, `train` and `test`. Each one is an `mldr` object, so the same operations previously described for this class of objects are applicable. Secondly, the same MLD is partitioned enabling the `get.indices` option. In this case the `train` and `test` members are numeric vectors rather than `mldr` objects.

5.5. How to export data to other formats

Although the MLDs in R format can be useful to perform exploratory analysis with the `mldr` package, or conduct some experimentation using the `mlr` package [48], most users would need to export them to other formats. This is the goal of the `write.mldr()` function. Currently it is able to write the content of any `mldr` object to the following formats:

- MULAN: The data is written in ARFF file format following the MULAN [49] multi-label standard: the labels are usually located at the end of each data row, and a separate XML file containing label names is also generated.
- MEKA: As for MULAN, the MEKA [50] file format is also ARFF-based. However, the number and locations of labels in the data is stated in the ARFF header itself, so a separate XML file is not needed.
- KEEL: This machine learning tool [51] also relies on the ARFF file format, as the two previous ones. The ARFF header enumerates the attributes acting as inputs and as outputs. Therefore, the labels can be located at any position on the dataset.
- LibSVM: It is the file format used by the well-known SVM library LibSVM [52]. It uses sparse representation, locating the labels at the beginning of each data row.
- CSV: In case none of the previous formats fits the user's needs, they can always export to CSV format and import the data from the tool to use. This format is the standard CSV, with the attributes and labels separated by commas with these at the end. A second CSV file with the label names is also generated.

When calling the `write.mldr()` function, an `mldr` object or the value returned by one of the partitioning functions described in the previous section must be given as the first argument. In the first case a single data file will be created (and one with the labels if applicable), while in the second case as many files as partitions contained in the list will be generated.

The format to export the MLD in is indicated via the `format` parameter. This should be a string with any of the format identifiers previously enumerated, i.e. `''KEEL''`. A vector with several formats can also be given, in which case the dataset is simultaneously written in all of them.

Many MLDs are sparse, mainly those having hundreds or thousands of input attributes. This means that only some of these attributes have a useful value in each row, the remaining ones being 0. Writing all these zeros in a text file implies a waste of space. This is the reason to use the ARFF sparse format, far more compact for sparse MLDs. The `sparse` parameter of the `write.mldr()` function takes the `FALSE` value by default. Assigning it the `TRUE` value activates this functionality. However, it should only be used with truly sparse MLDs, otherwise it will not produce any benefit. The `sparsity()` function in `mldr.datasets` can be used to check the sparsity level of any MLD. For instance:

```
> sparsity(emotions)
[1] 0.05834739
>
> sparsity(stackex_chess)
[1] 0.9729319
```

As can be seen, the `emotions` dataset has less than a 6% of sparsity, while for `stackex_chess` the level is above 97%. So the former should not be written as sparse, while the latter should be.

The last parameter accepted by the `write.mldr()` function is `basename`. It is useful to set the root of the filenames when several MLDs are going to be exported, usually as a result of a previous partitioning task. The original name of the MLD, stored in the `name` attribute, is used by default. If it is not a valid name, the string `''unnamed_mldr''` will be used instead.

In Fig. 12 two typical use cases of `write.mldr()` are shown. First, a sparse MLD is written to MEKA and CSV formats. Second, a dataset is partitioned and these partitions are exported to MULAN format.

6. Cometa: the comprehensive multi-label data archive

By means of the functionality offered by the package `mldr.datasets` any user can examine the characteristics of the MLDs, select the most appropriate ones for their study, partition and export them to the desired format, and reference them appropriately. Logically, it would be desirable for such partitions to be made publicly available so that third parties can use them for comparisons. In fact, the interesting point would be that we could

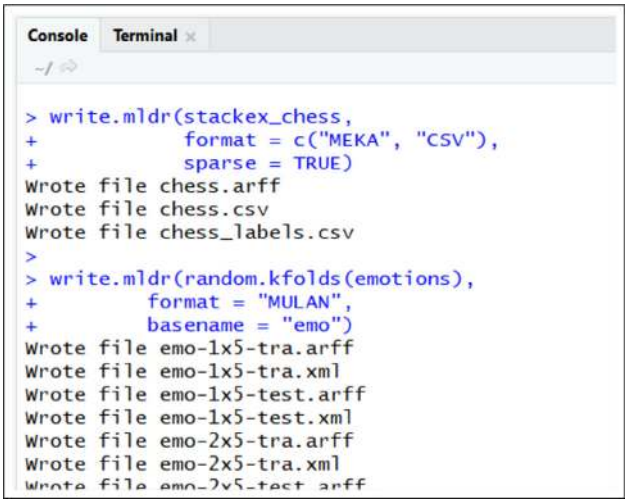


Fig. 12. Exporting a dataset and some partitions.

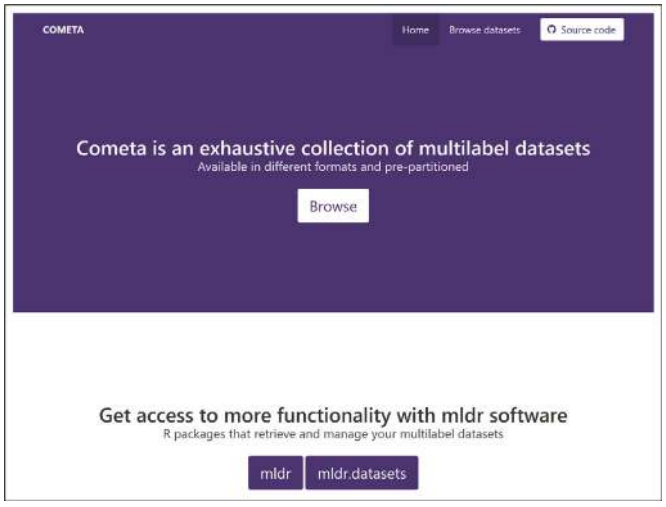


Fig. 13. Cometa main page.

all use the same data partitions, thus simplifying any comparative study. This is the goal behind the comprehensive multi-label data archive (Cometa).

Using the functions described in Section 5, we have taken many of the publicly accessible datasets, partitioned them according to different strategies, exported them to the most popular file formats and finally designed a website that acts as a repository of all that information. The repository is accessible at https://cometa.ml. Its purpose is to make it easier for researchers to use the same data partitions when conducting multi-label studies. This section describes Cometa's structure and the steps for creating your own Cometa repository with the desired datasets.

6.1. Browse the MLDs available at Cometa

The main page of Cometa (Fig. 13) provides several options, aimed to ease the access to related software packages, multi-label bibliography, the source code of Cometa itself, and the list of hosted MLDs. This is accessible through the **Browse** button

The list of MLDs initially available in Cometa, partially visible in Fig. 14, is provided in Table 2. Those marked with a √ symbol are built-in MLDs, available as soon as the `mldr.datasets` package is loaded into memory. The remaining ones can be obtained through the `get.mldr()` function explained in Section 5.2.3.

Table 2
Datasets initially available in Cometa.

Name	MLDs	Ref.	Field
bibtex	1	[53]	Text
birds √	1	[37]	Sound/Music
bookmarks	1	[53]	Text
cal500 √	1	[38]	Sound/Music
corel16k	10	[54]	Image
corel5k	1	[55]	Image
delicious	1	[20]	Text
emotions √	1	[39]	Sound/Music
enron	1	[56]	Text
EUR-Lex	3	[57]	Text
flags √	1	[40]	Image
foodtruck	1	[58]	Other
genbase √	1	[41]	Protein/Genetics
imdb	1	[14]	Text
langlog √	1	[42]	Text
mediamill	1	[59]	Video
medical √	1	[43]	Text
ng20 √	1	[44]	Text
nus-wide	2	[60]	Image
ohsumed	1	[61]	Text
rcv1v2	5	[62]	Text
reuters	1	[42]	Text
scene	1	[7]	Image
slashdot √	1	[14]	Text
stackexchange	6	[45]	Text
tmc2007	2	[63]	Text
yahoo	11	[64]	Text
yeast	1	[5]	Protein/Genetics

6.2. Filtering and searching MLDs

The dataset browsing page provides for each MLD a set of metrics, such as the number of features, labels, labelsets, the imbalance ratio, SCUMBLE and TCS measures, etc. That list is dynamic, so that the user can change the order simply by clicking the desired column header. A second click will reverse the order. This way looking for MLDs having certain traits, i.e. those with more labels or more imbalanced, becomes a simpler process.

6.3. Details about an MLD

In order to search a specific MLD or set of MLDs, all the user has to do is enter part of its name in the text box located at the top-left of the page. It is also valid to introduce a known value for any of the metrics shown in the list. The rows in it will be filtered as a result, easing the selection of the searched dataset.

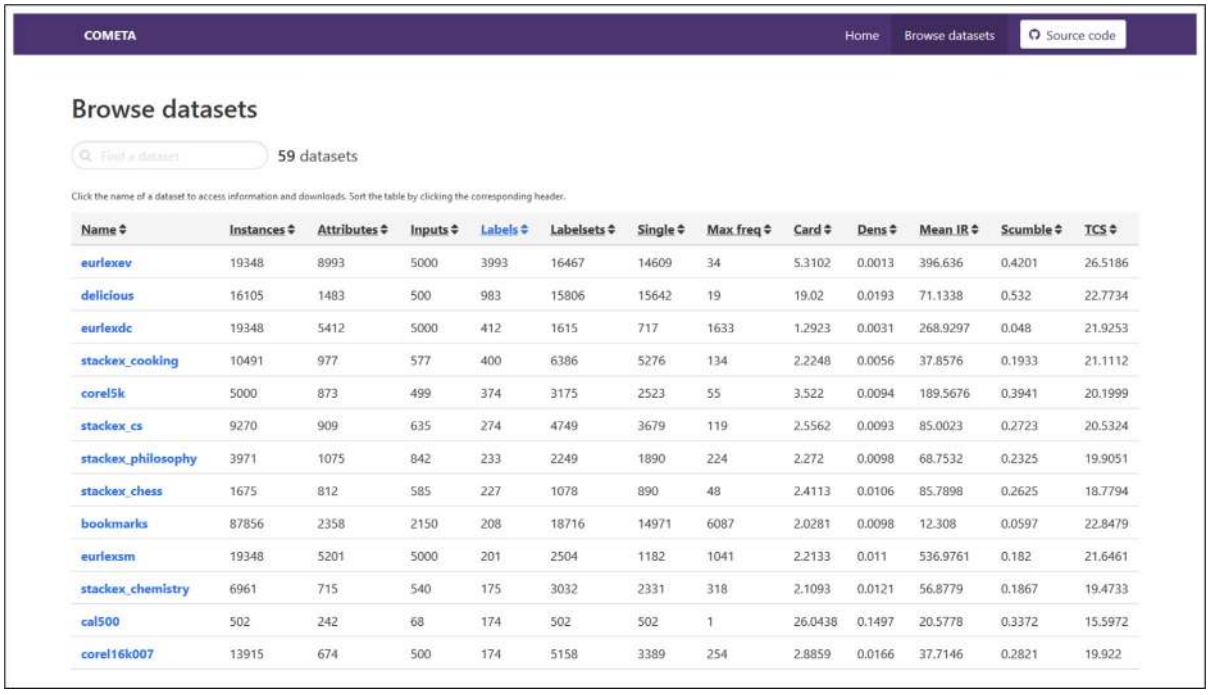
A click on the name of any of the datasets displays its detail page (see Fig. 15). This is composed of several panels, showing the measures that characterize the MLD, information on label concurrency, including a plot[3], and imbalance levels, the complete list of attributes indicating their type, the list of labels and finally, the source information necessary to reference it. This additional information should be useful to decide if the MLD is appropriate for the study at glance or not.

At the bottom of the page, a link provides all the information displayed on this page in JSON [65] format. This facilitates the automated treatment of meta-data related to MLDs.

6.4. Downloading data partitions

All the datasets available at Cometa can be downloaded from R through the `mldr.datasets` package. The tip located at the top-left of the details page, following the MLD's name, provides the command to use. This will provide the full dataset, without

[3] This kind of plot, among others, can be easily generated by the `mldr` package.



Name	Instances	Attributes	Inputs	Labels	Labelsets	Single	Max freq	Card	Dens	Mean IR	Scumble	TCS
eurlexev	19348	8993	5000	3993	16467	14609	34	5.3102	0.0013	396.636	0.4201	26.5186
delicious	16105	1483	500	983	15806	15642	19	19.02	0.0193	71.1338	0.532	22.7734
eurlexdc	19348	5412	5000	412	1615	717	1633	1.2923	0.0031	268.9297	0.048	21.9253
stackex_cooking	10491	977	577	400	6386	5276	134	2.2248	0.0056	37.8576	0.1933	21.1112
corel5k	5000	873	499	374	3175	2523	55	3.522	0.0094	189.5676	0.3941	20.1999
stackex_cs	9270	909	635	274	4749	3679	119	2.5562	0.0093	85.0023	0.2723	20.5324
stackex_philosophy	3971	1075	842	233	2249	1890	224	2.272	0.0098	68.7532	0.2325	19.9051
stackex_chess	1675	812	585	227	1078	890	48	2.4113	0.0106	85.7898	0.2625	18.7794
bookmarks	87856	2358	2150	208	18716	14971	6087	2.0281	0.0098	12.308	0.0597	22.8479
eurlexsm	19348	5201	5000	201	2504	1182	1041	2.2133	0.011	536.9761	0.182	21.6461
stackex_chemistry	6961	715	540	175	3032	2331	318	2.1093	0.0121	56.8779	0.1867	19.4733
cal500	502	242	68	174	502	502	1	26.0438	0.1497	20.5778	0.3372	15.5972
corel16k007	13915	674	500	174	5158	3389	254	2.8859	0.0166	37.7146	0.2821	19.922

Fig. 14. Browsing the list of datasets.

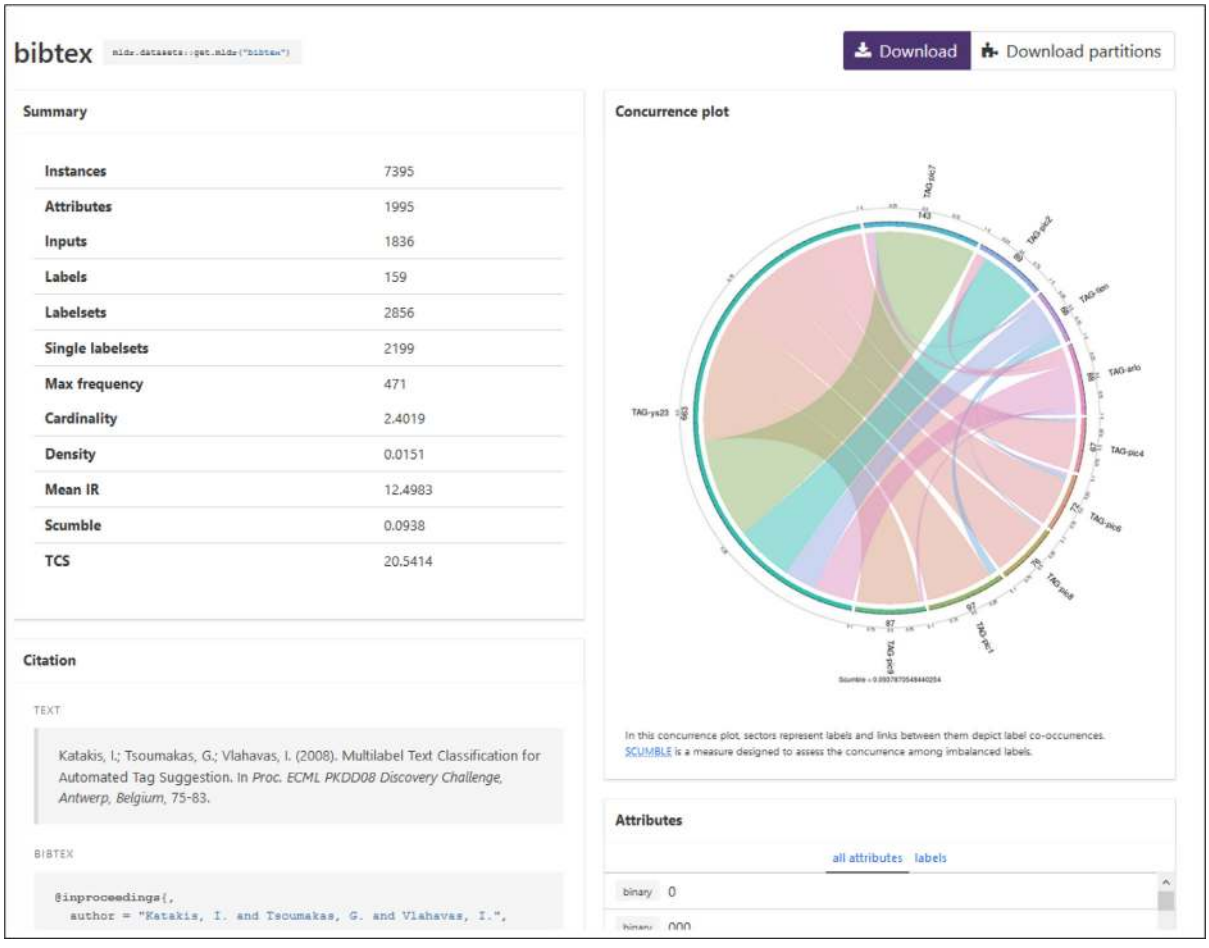


Fig. 15. MLD details page.

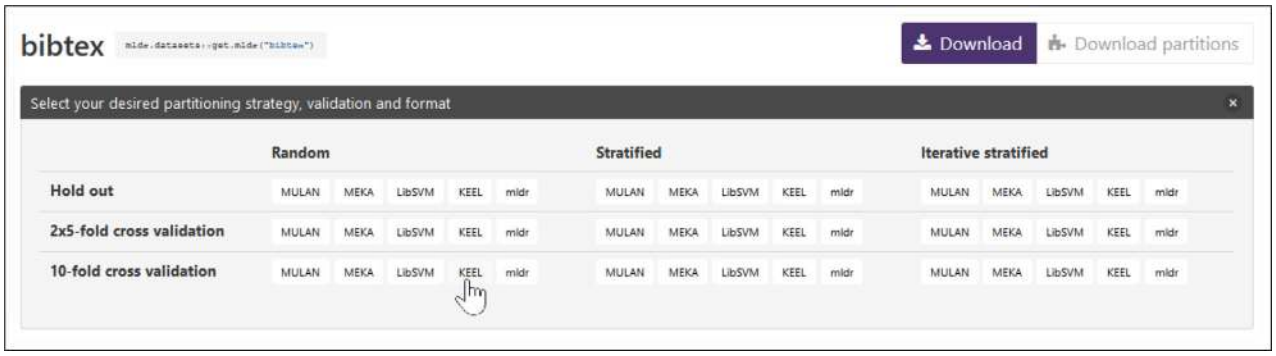


Fig. 16. Selecting the partition strategy, folds and format for downloading the MLD.

partitioning. This same version of the MLD can be also directly downloaded from the page, by clicking the **Download** button on the right.

To the right of the previous button there is another one whose objective is to allow the download of the chosen MLD already partitioned. Each of the datasets in the repository has been partitioned according to the following parameters:

- **Partitioning strategy**: All three partitioning methods supported by `mldr.datasets` (see Section 5.4) have been used with every MLD, so the user can choose between random, stratified and iteratively stratified partitions.
- **Number of folds**: Three configurations are provided for each partitioning strategy: hold out, 2x5-fcv and 10-fcv. The first consists of two partitions, a training partition with 60% of the instances and a test partition with the remaining 40%. The second one is made of two different sets of five folds, with 80% of samples for training and 20% for testing. The last configuration is a set of ten folds, each having 90% of instances for training and the remaining 10% for testing.
- **File format**: For each one of the previous nine strategy/folds configurations data have been exported to five file formats: MULAN, MEKA, KEEL, LibSVM and the mldr format.

This adds up to a total of 45 settings for each MLD, prepared to download and use in any experimental study. All of them are accessible through the **Download partitions** button previously mentioned. It opens a window as the one shown in Fig. 16, from where the a .tar.gz file for each case is available.

6.5. How to host your own Cometa repository

Since Cometa is based exclusively on open-source software, other researchers can build their own multi-label data repository by running the same software. However, installing the required software, manually partitioning the datasets and extracting their metadata might be a tedious task. In order to relieve them from this work, we provide a mostly automatic, menu-based assistant in the form of a Docker [66] image[4].

The assistant will automatically process datasets, but it will need them in mldr format for this. Provided that we are working in R, the `mldr()` or `mldr_from_dataframe()` functions from the mldr package can convert a dataset to mldr format. Afterwards, we just need to save this object into a file with the built-in function `saveRDS()`.

Assuming now that the directory containing the public data for the repository will be located in `~/public`, we should save our datasets in RDS file format inside `~/public/full`. At this point, the Cometa assistant can run by starting the Docker image in interactive mode and forwarding one port from the host to the 80 port in the container:

```
$ docker run -itp 8080:80 --mount \
  type=bind,source="~/public",target=/usr/app/public \
  fdavidcl/cometa
```

After downloading the required image, the main menu of the Cometa assistant will show some options:

1. Partition datasets
2. Create summaries of your data
3. Modify website configuration
4. Launch Cometa server
5. Quit

These options are intended to be processed in order. When choosing the first option, the assistant will scan the `public/full` folder for datasets, partition them according to different partitioning and validation strategies, and export them in a variety of formats into `public/partitions`. This process can take several hours depending on the size of the datasets, but it is only needed once. If serving dataset partitions is not desired, this option can be skipped safely.

The second option will again read the original datasets and output metadata in `public/json`. Running this option is required in order for the datasets to appear on the website. The third option will allow the user to modify parameters such as the website title or its accent color, and the fourth one will start a web server hosting the current datasets, which will be accessible at `localhost:8080`.

When partitions and metadata have been created, you may want to start the web server without requiring human interaction. This can be achieved by running Docker in detached mode:

```
$ docker run -dp 8080:80 --mount \
  type=bind,source="~/public",target=/usr/app/public \
  fdavidcl/cometa
```

After this command is run, the program will automatically build the website and serve it. That way, the server can be launched at system startup if desired.

7. Concluding remarks

The use of multi-label classification algorithms is becoming increasingly widespread, given the breadth of its applications. It is therefore important to design increasingly efficient methods that are tailored to specific needs. The behavior and performance of these new methods must always be validated experimentally. This requires appropriate procedures and tools.

[4] The Docker image is hosted on the Docker Hub at https://hub.docker.com/r/fdavidcl/cometa/.

This paper attempts to help improving the way multi-label experimentation is conducted through several contributions. First, we have identified the main traps that the practitioner can find while performing multi-label experiments, and then a set of good practices has been provided. In addition, we have developed and introduced the tools needed to follow these recommendations, thus easing this kind of work.

In the first sections of this article we have tried to compile a set of good practices regarding how a multi-label experimentation should be conducted. According to our experience, most mistakes are due to incorrect selection or processing of MLDs. Most of the pieces of advice provided relate to this aspect.

Aiming to ease the usual steps followed in a multi-label experimentation, we have developed a specific software: the `mldr.datasets` R package. As has been explained in Section 5, the functionality provided by this software makes easier the selection, partitioning, documentation and exporting of MLDs. `mldr.datasets` is free software available to any R user, and it is open to future extensions by the authors and the community.

Even those who are not R users can benefit from the functionality of this software package, thanks to Cometa, the repository from which 60 MLDs with different partitioning strategies, number of partitions and formats can be downloaded. The main objective of this repository is to facilitate that new multi-label studies always use the same MLD partitions. This would allow future comparisons between algorithms, without the need for each researcher to re-run all results for published methods. All that would have to be done is to take the same partitions of data used in the reference article. Like `mldr.datasets`, Cometa is free software and any user can set up their own repository, as well as contribute to Cometa by providing additional datasets.

As future work, we aim to enlarge the collection of MLDs hosted in Cometa, as well as extend the information provided for each one of them. The functionality of the `mldr.datasets` package could be also enhanced, for instance allowing any user to upload and process their datasets automatically from the R command line.

Acknowledgments

The authors are grateful to the editor and anonymous reviewers for their suggestions and advice, who have contributed to improving the content of this paper.

This work is supported by the Spanish National Research Projects TIN2014-57251-P and TIN2015-68454-R and the Project BigDaP-TOOLS - Ayudas Fundación BBVA a Equipos de Investigación Científica 2016.

References

[1] F. Herrera, F. Charte, A.J. Rivera, M.J. del Jesus, Multilabel Classification, in: Problem Analysis, Metrics and Techniques, Springer, 2016, doi:10.1007/978-3-319-41111-8.

[2] E. Gibaja, S. Ventura, A tutorial on multilabel learning, ACM Comput. Surv. 47 (3) (2015) 52:1–52:38., doi:10.1145/2716262.

[3] A. Clare, R.D. King, Knowledge discovery in multi-label phenotype data, in: Proceedings of the Fifth European Conference Principles on Data Mining and Knowledge Discovery, 2168, Freiburg, Germany, 2001, pp. 42–53, doi:10.1007/3-540-44794-6_4.

[4] M.L. Zhang, Multilabel neural networks with applications to functional genomics and text categorization, IEEE Trans. Knowl. Data Eng. 18 (10) (2006) 1338–1351, doi:10.1109/TKDE.2006.162.

[5] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, in: Proceedings of the Advances in Neural Information Processing Systems, 14, MIT Press, 2001, pp. 681–687.

[6] S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, in: Proceedings of the Advances in Knowledge Discovery and Data Mining, 3056, 2004, pp. 22–30, doi:10.1007/978-3-540-24775-3_5.

[7] M. Boutell, J. Luo, X. Shen, C. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (9) (2004) 1757–1771, doi:10.1016/j.patcog.2004.03.009.

[8] F. Charte, A. Rivera, M. del Jesus, F. Herrera, REMEDIAL-HwR: tackling multilabel imbalance through label decoupling and data resampling hybridization, Neurocomputing, 2018 In press, doi:10.1016/j.neucom.2017.01.118.

[9] N. Spolaôr, M.C. Monard, G. Tsoumakas, H.D. Lee, A systematic review of multilabel feature selection and a new method based on label construction, Neurocomputing 180 (2016) 3–15, doi:10.1016/j.neucom.2015.07.118.

[10] F. Charte, A. Rivera, M. del Jesus, F. Herrera, LI-MLC: a label inference methodology for addressing high dimensionality in the label space for multilabel classification, IEEE Trans. Neural Netw. Learn. Syst. 25 (10) (2014) 1842–1854, doi:10.1109/TNNLS.2013.2296501.

[11] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, Addressing imbalance in multilabel classification: measures and random resampling algorithms, Neurocomputing 163 (0) (2015) 3–16, doi:10.1016/j.neucom.2014.08.091.

[12] F. Charte, A. Rivera, M. del Jesus, F. Herrera, Dealing with difficult minority labels in imbalanced mutilabel data sets, Neurocomputing, 2018 In press, doi:10.1016/j.neucom.2016.08.158.

[13] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, On the impact of dataset complexity and sampling strategy in multilabel classifiers performance, in: Proceedings of the Eleventh International Conference on Hybrid Artificial Intelligent Systems, HAIS, 9648, Springer, 2016, pp. 500–511, doi:10.1007/978-3-319-32034-2_42.

[14] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. 85 (2011) 333–359, doi:10.1007/s10994-011-5256-5.

[15] J. Read, B. Pfahringer, G. Holmes, Multi-label classification using ensembles of pruned sets, in: Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008, pp. 995–1000.

[16] E. Hüllermeier, J. Fürnkranz, W. Cheng, K. Brinker, Label ranking by learning pairwise preferences, Artif. Intell. 172 (16) (2008) 1897–1916, doi:10.1016/j.artint.2008.08.002.

[17] J. Fürnkranz, E. Hüllermeier, E.L. Mencía, K. Brinker, Multilabel classification via calibrated label ranking, Mach. Learn. 73 (2008) 133–153, doi:10.1007/s10994-008-5064-8.

[18] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Ann. Data Sci. 2 (2) (2015) 165–193, doi:10.1007/s40745-015-0040-1.

[19] M.L. Zhang, MI-RBF: RBF neural networks for multi-label learning, Neural Process. Lett. 29 (2009) 61–74, doi:10.1007/s11063-009-9095-3.

[20] G. Tsoumakas, I. Katakis, I. Vlahavas, Effective and efficient multilabel classification in domains with large number of labels, in: Proceedings of the ECML/PKDD Workshop on Mining Multidimensional Data, Antwerp, Belgium, 2008, pp. 30–44.

[21] T. Megano, K.i. Fukui, M. Numao, S. Ono, Evolutionary multi-objective distance metric learning for multi-label clustering, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 2945–2952, doi:10.1109/CEC.2015.7257255.

[22] C. Braune, S. Besecke, R. Kruse, Density Based Clustering: Alternatives to DBSCAN, Springer International Publishing, Cham, 2015, pp. 193–213, doi:10.1007/978-3-319-09259-1_6.

[23] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 226–231.

[24] N. Cesa-Bianchi, C. Gentile, L. Zaniboni, Incremental algorithms for hierarchical classification, J. Mach. Learn. Res. 7 (2006) 31–54.

[25] M. Zhang, Z. Zhou, ML-KNN: a lazy learning approach to multi-label learning, 2007,. Pattern Recognit., 40, 7, 2038–2048, 10.1016/j.patcog.2006.12.019.

[26] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26, doi:10.1016/j.neucom.2016.12.038.

[27] J. Read, F. Pérez-Cruz, Deep learning for multi-label classification, ArXiv preprint arXiv:1502.05988 abs/1502.05988.

[28] K. Karalasa, G. Tsagkatakisb, M. Zervakisa, P. Tsakalidesa, Deep learning for multi-label land cover classification, in: Proceedings of the Remote Sensing, International Society for Optics and Photonics, 9643, 2015, p. 96430Q, doi:10.1117/12.2195082.

[29] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao, S. Yan, HCP: a flexible CNN framework for multi-label image classification, IEEE Trans. Pattern Anal. Mach. Intell. 38 (9) (2016) 1901–1907, doi:10.1109/TPAMI.2015.2491929.

[30] D. Charte, F. Charte, S. García, M.J. del Jesus, F. Herrera, A practical tutorial on autoencoders for nonlinear feature fusion: taxonomy, models, software and guidelines, Inf. Fusion 44 (2018) 78–96, doi:10.1016/j.inffus.2017.12.007.

[31] G. Madjarov, D. Kocev, D. Gjorgjevikj, S. Džeroski, An extensive experimental comparison of methods for multi-label learning, Pattern Recognit. 45 (9) (2012) 3084–3104, doi:10.1016/j.patcog.2012.03.004.

[32] S. Garca, F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, J. Mach. Learn. Res. 9 (66) (2008) 2677–2694.

[33] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064, doi:10.1016/j.ins.2009.12.010.

[34] F. Charte, D. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, R ultimate multilabel dataset repository, in: Proceedings of the Eleventh International Conference on Hybrid Artificial Intelligent Systems, 9648, Springer, 2016, pp. 487–499, doi:10.1007/978-3-319-32034-2_41.

[35] R.C. Team, A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria. 2014. http://www.R-project.org/R.
[36] H. Wickham, W. Chang, devtools: Tools to Make Developing R Packages Easier, r package version 1.8.0. 2015. http://CRAN.R-project.org/package=devtools.
[37] F. Briggs, B. Lakshminarayanan, L. Neal, X.Z. Fern, R. Raich, S.J.K. Hadley, A.S. Hadley, M.G. Betts, Acoustic classification of multiple simultaneous bird species: a multi-instance multi-label approach, J. Acoust. Soc. Am. 131 (6) (2012) 4640–4650, doi:10.1121/1.4707424.
[38] D. Turnbull, L. Barrington, D. Torres, G. Lanckriet, Semantic annotation and retrieval of music and sound effects, IEEE Audio, Speech, Lang.Process 16 (2) (2008) 467–476, doi:10.1109/TASL.2007.913750.
[39] A. Wieczorkowska, P. Synak, Z. Raś, Multi-label classification of emotions in music, in: Proceedings of the Intelligent Information Processing and Web Mining, 35, AISC, 2006, pp. 307–315, doi:10.1007/3-540-33521-8_30.
[40] E.C. Gonçalves, A. Plastino, A.A. Freitas, A genetic algorithm for optimizing the label ordering in multi-label classifier chains, in: Proceedings of the Twenty Fifth IEEE International Conference on Tools with Artificial Intelligence (ICTAI13), 2013, pp. 469–476, doi:10.1109/ICTAI.2013.76.
[41] S. Diplaris, G. Tsoumakas, P. Mitkas, I. Vlahavas, Protein classification with multiple algorithms, in: Proceedings of the Tenth Panhellenic Conference on Informatics, Volos, Greece, 2005, pp. 448–456, doi:10.1007/11573036_42.
[42] J. Read, Scalable multi-label classification, University of Waikato, 2010 (Ph.D. thesis).
[43] K. Crammer, M. Dredze, K. Ganchev, P.P. Talukdar, S. Carroll, Automatic code assignment to medical text, in: Proceedings of the Workshop on Biological, Translational, and Clinical Language Processing, Prague, Czech Republic, 2007, pp. 129–136.
[44] K. Lang, Newsweeder: Learning to filter netnews, in: Proceedings of the Twelfth International Conference on Machine Learning, 1995, pp. 331–339.
[45] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, QUINTA: a question tagging assistant to improve the answering ratio in electronic forums, in: Proceedings of the International Conference on Computer as a Tool (EUROCON), IEEE, 2015, pp. 1–6, doi:10.1109/EUROCON.2015.7313677.
[46] F. Charte, D. Charte, Working with multilabel datasets in R: the mldr package, R J. 7 (2) (2015) 149–162.
[47] K. Sechidis, G. Tsoumakas, I. Vlahavas, On the stratification of multi-label data, in: Proceedings of the Machine Learnig and Knowledge Discovery in Databases, Springer, 2011, pp. 145–158, doi:10.1007/978-3-642-23808-6_10.
[48] P. Probst, Q. Au, G. Casalicchio, C. Stachl, B. Bischl, Multilabel classification with R package mlr, R J. 9 (1) (2017) 352–369.
[49] G. Tsoumakas, E.S. Xioufis, J. Vilcek, I. Vlahavas, MULAN: a java library for multi-label learning, J. Mach. Learn. Res. 12 (2011) 2411–2414.
[50] J. Read, P. Reutemann, MEKA multi-label dataset repository. http://meka.sourceforge.net/#datasets.
[51] I. Triguero, S. González, J.M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M.J. del Jesús, L. Sánchez, F. Herrera, Keel 3.0: an open source software for multi-stage analysis in data mining, Int. J. Comput. Intell. Syst. 10 (1) (2017) 1238–1249, doi:10.2991/ijcis.10.1.82.
[52] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27:1–27:27, doi:10.1145/1961189.1961199.
[53] I. Katakis, G. Tsoumakas, I. Vlahavas, Multilabel text classification for automated tag suggestion, in: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Antwerp, Belgium, 2008, pp. 75–83.
[54] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D.M. Blei, M.I. Jordan, Matching words and pictures, J. Mach. Learn. Res. 3 (2003) 1107–1135.
[55] P. Duygulu, K. Barnard, J. de Freitas, D. Forsyth, Object recognition as machine translation: learning a lexicon for a fixed image vocabulary, in: Proceedings of the Seventh European Conference on Computer Vision, Part IV. Copenhagen, Denmark, 2002, pp. 97–112, doi:10.1007/3-540-47979-1_7.
[56] B. Klimt, Y. Yang, The Enron Corpus: a new dataset for email classification research, in: Proceedings of the European Conference on Machine Learning (ECML), Pisa, Italy, 2004, pp. 217–226, doi:10.1007/978-3-540-30115-8_22.
[57] E.L. Mencia, J. Fürnkranz, Efficient pairwise multilabel classification for large-scale problems in the legal domain, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 50–65, doi:10.1007/978-3-540-87481-2_4.
[58] A. Rivolli, L.C. Parker, A.C. de Carvalho, Food truck recommendation using multi-label classification, in: Portuguese Conference on Artificial Intelligence, Springer, 2017, pp. 585–596, doi:10.1007/978-3-319-65340-2_48.
[59] C.G.M. Snoek, M. Worring, J.C. van Gemert, J.M. Geusebroek, A.W.M. Smeulders, The challenge problem for automated detection of 101 semantic concepts in multimedia, in: Proceedings of the Fourteenth ACM Annual International Conference on Multimedia, Santa Barbara, CA, USA, 2006, pp. 421–430, doi:10.1145/1180639.1180727.
[60] T.S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, Y. Zheng, NUS-WIDE: a real-world web image database from National University of Singapore, in: Proceedings of the ACM International Conference on Image and Video Retrieval, 48, ACM, 2009, doi:10.1145/1646396.1646452.
[61] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of the Tenth European Conference on Machine Learning, Springer-Verlag, 1998, pp. 137–142, doi:10.1007/BFb0026683.
[62] D.D. Lewis, Y. Yang, T.G. Rose, F. Li, RCV1: a new benchmark collection for text categorization research, J. Mach. Learn. Res. 5 (2004) 361–397.
[63] A.N. Srivastava, B. Zane-Ulman, Discovering recurring anomalies in text reports regarding complex space systems, in: Proceedings of the Aerospace Conference, IEEE, 2005, pp. 3853–3862, doi:10.1109/AERO.2005.1559692.
[64] N. Ueda, K. Saito, Parametric mixture models for multi-labeled text,, Adv. Neural Inf. Process. Syst. (2002) 721–728.
[65] http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[66] https://www.docker.com/ Docker: Build, ship and run any Docker Inc., app anywhere.

Francisco Charte received his B.Eng. degree in Computer Science from the University of Jaèn in 2010 and his M.Sc. and Ph.D. in Computer Science from the University of Granada in 2011 and 2015. He is an Assistant Professor of Computer Architecture and Computer Technology with the Computer Science Department at the University of Jaèn (Spain). He is coauthor of the book "Multilabel Classification-Problem Analysis, Metrics and Techniques" (Springer, 2016). His main research interests include machine learning with applications to multilabel classification, high dimensionality and imbalance problems, as well as deep learning algorithms.

Antonio J. Rivera received his B.Sc. degree and his Ph.D. in Computer Science from the University of Granada in 1995 and 2003, respectively. He is a lecturer of Computer Architecture and Computer Technology with the Computer Science Department at the University of Jaèn (Spain). He is coauthor of the book "Multilabel Classification-Problem Analysis, Metrics and Techniques" (Springer, 2016). His research interests include areas such as multilabel classification, imbalance problems, evolutionary computation, neural network design, time series prediction and regression tasks.

David Charte received his B.Sc degrees in Mathematics and Computer Science from the University of Granada in 2017. He is currently studying for a M.Sc. in Data Science and Computer Engineering at the same university, while involved in research for the Department of Computer Science and Artificial Intelligence. He is coauthor of several pieces of software for Data Science. His research interests include unsupervised deep learning techniques and their applications, problems with high dimensionality and multilabel classification.

Marìa J. Del Jesus received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 1994 and 1999, respectively. She is a Professor with the Department of Computer Science at University of Jaèn (Spain). She has been the supervisor of 7 Ph.D. students. She has published more than 65 journal papers that have received more than 4500 citations (Scholar Google, H-index 31). She is coauthor of the book "Multilabel Classification-Problem Analysis, Metrics and Techniques" (Springer, 2016). Her current research interests include fuzzy rule-based systems, subgroup discovery, multilabel classification, data preparation, radial basis neural networks, knowledge extraction based on evolutionary algorithms, and data science and big data.

Francisco Herrera (SM'15) received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has been the supervisor of 41 Ph.D. students. He has published more than 350 journal papers, receiving more than 55000 citations (Scholar Google, H-index 118). He is coauthor of the books "Genetic Fuzzy Systems" (World Scientific, 2001) and "Data Preprocessing in Data Mining" (Springer, 2015), "The 2-tuple Linguistic Model. Computing with Words in Decision Making" (Springer, 2015), Multilabel Classification. Problem analysis, metrics and techniques" (Springer, 2016), among others. He currently acts as Editor in Chief of the international journals "Information Fusion" (Elsevier) and "Progress in Artificial Intelligence (Springer). He acts as editorial member of a dozen of journals. He received the following honors and awards: ECCAI Fellow 2009, IFSA Fellow 2013, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 and 2012 Paper Award (bestowed in 2011 and 2015, respectively), 2011 Lotfi A. Zadeh Prize Best paper Award (IFSA Association), 2013 AEPIA Award to a scientific career in Artificial Intelligence, 2014 XV Andalucìa Research Prize Maimnides, 2017 Security Forum I+D+I Prize, and 2017 Andalucìa Medal (by the regional government of Andalucìa). He has been selected as a Highly Cited Researcher http://highlycited.com/ (in the fields of Computer Science and Engineering, respectively, 2014 to present, Clarivate Analytics). His current research interests include among others, soft computing (including fuzzy modeling and evolutionary algorithms), information fusion and decision making, biometric, data preprocessing, data science, deep learning and big data.