Information Fusion 95 (2023) 120–142

Contents lists available at ScienceDirect

Information Fusion

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

Full length article

PARIS: Partial instance and training set selection. A new scalable approach to multi-label classification☆

Nicolás García-Pedrajas *, José M. Cuevas-Muñoz, Juan A. Romero del Castillo, Aida de Haro-García

University of Córdoba, Campus de Rabanales, 14071, Córdoba, Spain

ARTICLE INFO

Keywords:
Multi-label classification
Instance selection
Scaling-up
Instance-based learning

ABSTRACT

Multi-label classification has recently attracted research interest as a data mining task. Many current applications in data mining address problems that have instances belonging to more than one class. This requires the development of new efficient methods. Instance selection has been used in multi-label learning to improve the execution time and classification performance of many learning methods. Following the single-label approach, instance selection has been applied by selecting or unselecting the same instances for all labels. In this paper, we present a different and novel approach. An instance might be useful for some labels and harmful for others; therefore, our algorithm allows each instance to be discarded, selected, or only partially selected for use in the classification of certain labels. An extensive comparison using 45 datasets shows the usefulness of our approach in improving the current instance selection methods for multi-label problems, as well as the ability of our algorithm to compete with other more complex multi-label classification methods.

1. Introduction

Many modern applications involve vast amounts of data for classification in increasingly complex categorization schemes, where one instance of data may simultaneously belong to several topics. This task is typically termed multi-label learning [1]. In contrast to single-label classification, where each instance is associated with only one class, multi-label classification is concerned with learning where each instance can be associated with multiple labels. The generality of multi-label problems makes them more difficult than their single-label counterparts. Multi-label learning is a special case of multi-output learning [2]. In multi-output learning we have a set of discrete labels, while in multi-label learning all the labels have only binary values. Although our work is devoted to multi-label problems, it can be extended to multi-output data.

Multi-label classification has received much attention over the past few years, and a variety of methods have been developed and applied to diverse problems: text categorization, automatic annotation for multimedia contents, web mining, rule mining, cheminformatics, bioinformatics, information retrieval, etc.

The key challenge of multi-label learning is to take advantage of the correlations among labels to mitigate the exponential growth of the label space with the number of distinct labels. Methods that deal with multi-label datasets without considering the relationships between labels are simply solving a group of independent binary problems. There are two broad categories of methods to deal with multi-label problems [1]: problem transformation methods and algorithm adaptation methods. Problem transformation methods tackle the multi-label learning problem by transforming it into other well-established learning scenarios. Algorithm adaptation methods tackle a multi-label learning problem by adapting well-known learning techniques to deal with the multi-label data directly.

As stated, the success of any approach is based on taking advantage of the correlation of labels to come up with novel methods. These relationships can be complex and even structured in hierarchies. Multi-label classification methods are usually grouped into three families depending on the order of label correlations considered: first-order strategies, second-order strategies, and higher-order strategies. These three families can appear in both problem transformation and algorithm adaptation methods.

Among the best performing methods in multi-label learning are instance-based methods, such as the adaptation of the k-nearest neighbors to multi-label datasets, Multi-label k-Nearest Neighbors [3]

☆ This work was supported by grant number PID2019-109481GB-I00/AEI/q10.13039/501100011033 Spanish Ministry of Science and Innovation and grant number UCO-1264182 from the Junta de Andalucía, Spain Excellence in Research Program and FEDER Funds. Funding for open access Universidad de Córdoba/CBUA.

* Corresponding author.
E-mail addresses: npedrajas@uco.es (N. García-Pedrajas), i62cumuj@uco.es (J.M. Cuevas-Muñoz), aromero@uco.es (J.A. Romero del Castillo), adeharo@uco.es (A. de Haro-García).

https://doi.org/10.1016/j.inffus.2023.02.017
Received 14 December 2022; Received in revised form 1 February 2023; Accepted 13 February 2023
Available online 16 February 2023

(ML-kNN) method and its numerous variants, DML-kNN [4] or SML-kNN [5], among many others, instance-based logistic regression for multi-label problems [6] (IBLR-ML and IBLR-ML+), or soft relevance for multi-label classification [7] (Mr.KNN). However, as in the single label case, these methods have the drawback of needing the training set to be stored in the memory, thus increasing the runtime of the algorithm and the necessary resources. Instance selection is a way to address this problem and improve the performance of instance-based classifiers [8,9]. Furthermore, instance selection can also be applied to other classification methods, such as binary relevance or classifier chains, with successful results. In that case, the term 'training set selection' is most commonly used [10].

Instance and training set selection has been carried out for multi-label learning in the same way as for single-label learning [11]. That means that an instance is either selected or removed. However, in multi-label learning, an instance may be useful for some of the labels and harmful for others. Selecting or unselecting an entire instance might be a suboptimal method, and new more appropriate algorithms may be developed.

Thus, we propose a different approach. Instead of carrying out instance selection at the instance level, we perform instance selection at the instance and label levels simultaneously. In standard instance selection, an algorithm selects a subset of instances, S, from the training set T, $S \subset T$. In our approach, the result is different. We obtain a subset of labels to be considered for each instance. This means that an instance can be selected, when it is considered for all labels, partially selected, when it is considered for only some labels, and removed, when it is not considered for any label. This approach yields to the general definition of partial instance selection. It will be adapted for different multi-label classifiers in this paper.

Furthermore, the distribution of the imbalance ratio of the different labels is highly skewed for most datasets. Some of the labels are clearly imbalanced, whereas other present an approximately balanced distribution. It has been repeatedly shown that most classification methods suffer from an imbalanced distribution of the training instances among the classes [12]. However, dealing with class imbalance in instance selection in multi-label problems is a complex task. First, the concept of a minority class instance is not clear, as the list of relevant labels of a given instance can have labels from both majority and minority classes. Charte et al. [13] implemented a way of oversampling the minority class and undersampling the majority class by extending the concept of minority and majority instance to multi-label datasets. They defined a label as minority if its frequency was below the average frequency of all the labels. An instance was a minority instance if any of its labels was a minority label. However, with this definition, undersampling and oversampling affected many labels that were not in the considered set. For instance, an instance with only one label belonging to a minority class could be oversampled, which might result in damaging the performance of other labels.

Our approach is effective for this problem, as the instances are selected depending on the labels. In that way an underrepresented label can be kept, while another one that is overrepresented can be removed. Thus, partial instance selection can present the learning algorithm with a more balanced dataset, which cannot be performed with standard instance selection.

As stated, our approach can be also applied to other multi-label methods for partial training set selection. We show results in this paper for two problem transformation methods, binary relevance (BR) and classifier chains (CC), as well as an algorithm adaptation method, ML-kNN. The adaptation to other methods can be more or less straightforward depending on the specific algorithm. We tested our approach on a broad benchmark of 45 datasets. Our results prove the superiority of our proposal compared with ML-kNN, BR, and CC methods with the whole dataset and standard instance selection. The results also show that our method is competitive when compared with state-of-the-art multi-label classification methods.

As a summary the main contributions of this paper are the following:

- A new framework for performing partial instance selection is developed. This partial instance selection is based on selecting instances depending on the label instead of selecting whole instances.
- An implementation of that framework for different multi-label methods, such as, multi-label k-nearest neighbors rule, binary relevance and classifier chains, is presented.
- A study of the compared performance of the proposal with other instance selection and classification methods as well as a study of the behavior or the proposed method depending on the characteristics of the datasets is carried out.

This paper is organized as follows: Section 2 reviews some related work; Section 3 describes our proposal; Section 4 describes the experimental setup; Section 5 presents the results of our experiments; and Section 6 summarizes the conclusions of our work.

2. Related work

Compared with the single-label case, very few works have been devoted to instance selection for multi-label problems. Arnáiz-González et al. [14] extended the concept of local sets used for single label instance selection to the multi-label case [8]. The developed method achieved good performance in a set of 11 problems. Kordos et al. [15] developed an evolutionary method for a multi-output regression.

Romero del Castillo et al. [16] developed a scalable evolutionary algorithm to deal with the problem of instance selection for large datasets. García-Pedrajas and Cerruela-García [11] developed a cooperative coevolutionary algorithm to also address the problem of large datasets using the problem decomposition inherent to cooperative coevolution. All these methods were based on the standard approach of selecting instances as a whole. To the best of our knowledge, no previous approach has been developed to select partial instances for multi-label learning.

Other works have been devoted to simultaneous instance and feature selection. Ma and Chow [17] developed a formulation for multi-label learning from a topic view that exploited the dependence between features and labels in a topic space. They performed effective instance and feature selection in the latent topic space, as they argued that the relationship between the input and output spaces are well captured in that space. Mansouri and Benabdeslem [18] also developed a simultaneous instance and feature selection method that they coupled with label selection. A different approach was addressed by Valero et al. [19]. They developed a method for prototype generation rather than instance selection with the same aim of data reduction.

3. Partial instance selection (PARIS) for multi-label datasets

Formally, we can define a multi-label problem as follows [1]: Let T be a multi-label evaluation dataset consisting of p multi-label instances $\mathbf{x}_i$ and their associated label set Y_i, where $T = \{(\mathbf{x}_i, Y_i)\}, 1 \leq i \leq p$, $(\mathbf{x}_i \in X, Y_i \in \mathcal{Y} = \{0,1\}^q)$ with a label set L, where $|L| = q$. Let h be a multi-label classifier and $h(\mathbf{x}_i) = \{0,1\}^q$ be the set of labels predicted by h for the instance $\mathbf{x}_i$. Let $f(\mathbf{x}_i, Y_i)$, $\mathbf{x}_i \in X$, $Y_i \in \mathcal{Y}$ be a real-valued function $f : X \times \mathcal{Y} \rightarrow R$. A successful learning system would tend to output larger values for function f for the labels in Y_i versus those not in Y_i. The real-valued function f can be easily transformed to a ranking function, $rank_f(\mathbf{x}_i, Y_i)$, where $rank_f$ is the predicted rank of label Y_i, for instance $\mathbf{x}_i$. $h(\mathbf{x}_i)$ can be obtained from $f(\mathbf{x}_i)$ when an appropriate threshold is set.

The set of Y_i is the set of relevant labels for $\mathbf{x}_i$, whereas $\overline{Y}_i$ is the set of irrelevant labels. In standard instance selection, a subset S of the instances is kept for training the model, $S \subset T$. When implemented using an evolutionary approach, a one-dimensional array $\mathbf{s}$ is usually associated with the training, where for each instance $\mathbf{x}_i$, $s_i = 1$ means

that the instance is selected, and $s_i = 0$ means that the instance is removed from the training set.

Our approach is based on selecting different instances for different labels; so, we consider a two-dimensional array $\mathbf{S}$, where $S_{il} = 1$ means that instance $\mathbf{x}_i$ is considered for training the learning model when label l is involved. For instance, for ML-kNN it means that $\mathbf{x}_i$ is considered for obtaining the prior and posterior probabilities of label y_l, and $S_{il} = 0$ means it is not. This formulation of "**Par**tial **i**nstance **s**election" (PARIS) is general enough to be used with most of the multi-label learning methods [1]. In this paper, we describe the application of PARIS to ML-kNN, BR, and CC, although it could be adapted to many others.

For ML-kNN, our first step is to redefine how ML-kNN works with this array $\mathbf{S}$. Given a dataset T, we consider, for each sample, the instance value, $\mathbf{x}_i$, the set of relevant labels, Y_i, and the set of irrelevant labels, $\bar{Y}_i$: $(\mathbf{x}_i, Y_i, \bar{Y}_i)$. In our selection process, we select the labels for which the instance will be considered, resulting in a new sample $(\mathbf{x}_i, Y_i', \bar{Y}'_i)$, where $Y_i' \subseteq Y_i$, and $\bar{Y}'_i \subseteq \bar{Y}_i$. Within this framework, an instance is selected if $Y_i' = Y_i$, and $\bar{Y}'_i = \bar{Y}_i$, and an instance is partially selected if $Y_i' \subseteq Y_i$ and $\bar{Y}'_i \subseteq \bar{Y}_i$ and $Y_i' \neq Y_i$ or $\bar{Y}'_i \neq \bar{Y}_i$. Only when $Y_i' \cup \bar{Y}'_i = \emptyset$ is the instance removed.

The adaptation of ML-kNN to PARIS, ML-kNN(PARIS), is shown in Algorithm 1. s_{il} indicates whether label l is selected for instance $\mathbf{x}_i$. We must stress that this does not mean that $y_l \in Y_i$, but that instance $\mathbf{x}_i$ is considered for any calculation involving label l. Thus, for every $l \in \mathcal{Y}$:

$$P(H_1^l) = (s + \sum_{i=1, s_{il}=1}^{m} \mathbf{y}_{x_i}(l))/(x \times 2 + m_l), \tag{1}$$

where $m_l = \sum_{i=1}^{m} S_{il}$.

The next decision is how to select the k neighbors that will be used for the computation of the posterior probabilities. We have two different alternatives. We can select the same set of neighbors for all the labels. In that case, we consider an instance for computing the neighbors if it is selected for at least one of the labels. That is, we consider instance $\mathbf{x}_i$ if and only if $\mathbf{S}_{i\cdot} \neq \mathbf{0}$. With this setup, the number of actual neighbors considered for each label is different because for a certain label l some of the members $\mathbf{x}_i \in N(\mathbf{x}_j)$ may be unselected, $S_{il} = 0$.

The second alternative is to consider the same number of neighbors for every label, meaning that the set of neighbors for every label will be different. In such a case, an instance $\mathbf{x}_i$ is considered for computing the neighbors for label l if and only if $S_{il} = 1$. In this case, for an instance $\mathbf{x}_j$, we have a different set of k-nearest neighbors $N_l(\mathbf{x}_j)$ for every label $l \in \mathcal{Y}$.

Algorithm 1 summarizes the first version, and Algorithm 2 summarizes the second version. In the experimental setup, we will refer to the first algorithm as PARIS and to the second version as PARIS+LK, PARIS with local k. We name this algorithm PARIS with local k because the practical result of the method is the use of a different k value for every label depending on $\mathbf{S}$.

For BR method, the adaptation of PARIS is straightforward. In BR method, we have a set of binary classifiers $(g_1, g_2, \ldots, g_q)$, where classifier g_j is constructed using the training set $\mathcal{D}_j = \{(\mathbf{x}_i, \phi(Y_i, y_j)) | 1 \leq i \leq m\}$, where

$$\phi(Y_i, y_j) = \begin{cases} +1, & \text{if } y_j \in Y_i \\ -1, & \text{otherwise.} \end{cases} \tag{2}$$

The definitions of $\mathcal{D}_j$ and $\phi(\cdot)$ are modified according to the description of our algorithm, $\mathcal{D}_j = \{(\mathbf{x}_i, \phi(\mathbf{x}_i, Y_i', \bar{Y}'_i)) | 1 \leq i \leq m\}, y_j \in Y_i' \vee y_j \in \bar{Y}'_i$, and

$$\phi(\mathbf{x}_i, Y_i', \bar{Y}'_i) = \begin{cases} +1, & \text{if } y_j \in Y_i' \\ -1, & \text{if } y_j \in \bar{Y}'_i. \end{cases} \tag{3}$$

The testing stage of BR is not modified. Once the classifiers are trained using the subsets defined above, the algorithm proceeds as in its standard definition.

Algorithm 1: PARIS implementation for ML-kNN.

Data: A training set $T = \{(\mathbf{x}_i, y_i, \mathbf{S})\}, 1 \leq i \leq p$,
$\mathbf{x}_i \in X, y_i \in \mathcal{Y} = \{0,1\}^q, S_{il} \in \{0,1\}$, k, t, s
Result: $[\mathbf{y}_t, \mathbf{r}_t]$
/* Computing prior probabilities */
[1] **for** $l \in \mathcal{Y}$ **do**
[3] $P(H_1^l) = (s + \sum_{i=1, S_{il}=1}^{m} \mathbf{y}_{x_i}(l))/(x \times 2 + m_l)$, $m_l = \sum_{i=1}^{m} S_{il}$
[5] $P(H_0^l) = 1 - P(H_1^l)$
/* Computing posterior probabilities */
[7] Identify $N(x_i)$, $i \in \{1, 2, \ldots, m\}$
[8] **for** $l \in \mathcal{Y}$ **do**
[9] **for** $j \in \{0, 1, \ldots, k_l\}$ **do**
[11] $c[j] = c'[j] = 0$
[12] **for** $i \in \{1, 2, \ldots, m\}$ **do**
[14] $\delta = \mathbf{C}_{x_i}(l) = \sum_{a \in N(x_i)} \mathbf{y}_a(l)$
[15] **if** $\mathbf{y}_{x_i} == l$ **then**
[17] $c[\delta] = c[\delta] + 1$
[18] **else**
[20] $c'[\delta] = c'[\delta] + 1$
[21] **for** $j \in \{0, 1, \ldots, k_l\}$ **do**
[23] $P(E_j^l | H_1^j) = (s + c[j])/(s \times (k_l + 1) + \sum_{p=0}^{k_l} c[p])$
[25] $P(E_j^l | H_0^j) = (s + c'[j])/(s \times (k_l + 1) + \sum_{p=0}^{k_l} c'[p])$
/* Computing $\mathbf{y}_t$ and $\mathbf{r}_t$ */
[27] Identify $N(t)$,
[28] **for** $l \in \mathcal{Y}$ **do**
[30] $\mathbf{C}_t(l) = \sum_{a \in N(t)} \mathbf{y}_a(l)$
[32] $\mathbf{y}_t(l) = \arg\max_{b \in \{0,1\}} P(H_b^l) P(E^l_{\mathbf{C}_t(l)} | H_b^l)$
[34] $\mathbf{r}_t(l) = P(H_1^l | E^l_{\mathbf{C}_t(l)}) = (P(H_1^l) P(E^l_{\mathbf{C}_t(l)} | H_1^l)) / P(E^l_{\mathbf{C}_t(l)}) =$
$(P(H_1^l) P(E^l_{\mathbf{C}_t(l)} | H_1^l)) / P(E^l_{\mathbf{C}_t(l)}) / (\sum_{b \in \{0,1\}} (P(H_b^l) P(E^l_{\mathbf{C}_t(l)}) | H_b^l))$
[36] Return $[\mathbf{y}_t, \mathbf{r}_t]$

The third method we have adapted is classifier chains (CC). The adaptation is almost as straightforward as for BR. CC is defined as follows [1]: For q possible class labels $\{y_1, y_2, \ldots, y_q\}$, a permutation function $\tau : \{1, \ldots, q\} \to \{1, \ldots, q\}$ defines an ordering of the labels. For the jth label, $y_{\tau(j)}, (1 \leq j \leq q)$, a binary training set is constructed by appending each instance with its relevant labels preceding $y_{\tau j}$:

$$\mathcal{D}_{\tau(j)} = \left\{ \left([\mathbf{x}_i, \mathbf{pre}^i_{\tau(j)}], \phi(Y_i, y_{\tau(j)}) \right) | 1 \leq i \leq m \right\}, \tag{4}$$

where $\mathbf{pre}^i_{\tau(j)} = \phi(Y_i, y_{\tau(1)}), \ldots, \phi(Y_i, y_{\tau(j-1)})$. With this training set, a certain binary learning algorithm is used to induce the binary classifier: $g_{\tau(j)} : \mathcal{X} \times \{-1, +1\}^{j-1} \to \mathcal{R}$. To predict an unseen instance $\mathbf{q}$, the outputs of the chain must be obtained. Let $\lambda^{\mathbf{q}}_{\tau(j)}$ represent the predicted assignment of $y_{\tau(j)}$ for $\mathbf{q}$:

$$\lambda^{\mathbf{q}}_{\tau(1)} = \text{sign}(g_{\tau(j)}(\mathbf{q})) \tag{5}$$

$$\lambda^{\mathbf{q}}_{\tau(j)} = \text{sign}(g_{\tau(j)}[\mathbf{q}, \lambda^{\mathbf{q}}_{\tau(1)}, \ldots, \lambda^{\mathbf{q}}_{\tau(j-1)}]), (2 \leq j \leq q). \tag{6}$$

The predicted set of labels is given by:

$$Y = \{y_{\tau(j)} | \lambda^{\mathbf{q}}_{\tau(j)} = +1, 1 \leq j \leq q\}. \tag{7}$$

As in the case for BR, we only need to modify the training stage of this method, as the testing stage will remain the same. The training set for the classifier jth is given by

$$\mathcal{D}_{\tau(j)} = \left\{ \left([\mathbf{x}_i, \mathbf{pre}^i_{\tau(j)}], \phi(Y_i, y_{\tau(j)}) \right) | 1 \leq i \leq m \right\}, y_j \in Y_i' \vee y_j \in \bar{Y}'_i. \tag{8}$$

Other methods can also be adapted to the partial instance selection. However, with these three descriptions, we have clearly shown the

Algorithm 2: PARIS(LK) implementation for ML-kNN.

Data: A training set $T = \{(\mathbf{x}_i, y_i, \mathbf{S})\}, 1 \leq i \leq p$,
$\mathbf{x}_i \in X, y_i \in \mathcal{Y} = \{0,1\}^q, S_{il} \in \{0,1\}$, k, t, s
Result: $[\mathbf{y}_t, \mathbf{r}_t]$
/* Computing prior probabilities */
for $l \in \mathcal{Y}$ **do**
  $P(H_1^l) = (s + \sum_{i=1,S_{il}=1}^{m} \mathbf{y}_{x_i}(l))/(s \times 2 + m_l)$, $m_l = \sum_{i=1}^{m} S_{il}$
  $P(H_0^l) = 1 - P(H_1^l)$
/* Computing posterior probabilities */
Identify $N_l(x_i)$, $i \in \{1, 2, \ldots, m\}, l \in \{1, 2, \ldots, q\}$
for $l \in \mathcal{Y}$ **do**
  for $j \in \{0, 1, \ldots, k_l\}$ **do**
    $c[j] = c'[j] = 0$
  for $i \in \{1, 2, \ldots, m\}$ **do**
    $\delta = \mathbf{C}_{x_i}(l) = \sum_{a \in N_l(x_i)} \mathbf{y}_a(l)$
    if $\mathbf{y}_{x_i} == l$ **then**
      $c[\delta] = c[\delta] + 1$
    else
      $c'[\delta] = c'[\delta] + 1$
  for $j \in \{0, 1, \ldots, k_l\}$ **do**
    $P(E_j^l | H_1^j) = (s + c[j])/(s \times (k_l + 1) + \sum_{p=0}^{k_l} c[p])$
    $P(E_j^l | H_0^j) = (s + c'[j])/(s \times (k_l + 1) + \sum_{p=0}^{k_l} c'[p])$
/* Computing $\mathbf{y}_t$ and $\mathbf{r}_t$ */
Identify $N_l(t)$,
for $l \in \mathcal{Y}$ **do**
  $\mathbf{C}_t(l) = \sum_{a \in N_l(t)} \mathbf{y}_a(l)$
  $\mathbf{y}_t(l) = \arg\max_{b \in \{0,1\}} P(H_b^l) P(E^l_{\mathbf{C}_t(l)} | H_b^l)$
  $\mathbf{r}_t(l) = P(H_1^l | E^l_{\mathbf{C}_t(l)}) = (P(H_1^l) P(E^l_{\mathbf{C}_t(l)} | H_1^l))/P(E^l_{\mathbf{C}_t(l)}) =$
  $(P(H_1^l) P(E^l_{\mathbf{C}_t(l)} | H_1^l))/P(E^l_{\mathbf{C}_t(l)})/(\sum_{b \in \{0,1\}} (P(H_b^l) P(E^l_{\mathbf{C}_t(l)} | H_b^l))$
Return $[\mathbf{y}_t, \mathbf{r}_t]$

procedure to adapt our proposal to the current methods for multi-label learning.

3.1. Evolutionary algorithm description

Having defined how the description of the multi-label learning methods was modified for partial instance selection, we now describe the details of the algorithm used to obtain the partial selection $\mathbf{S}$. We formulate our problem as an optimization problem that consists of obtaining the best selection $\mathbf{S}$ from the space of all possible selections. It is evident that we face a large search space and a complex way to evaluate each solution. In such situations, evolutionary algorithms have proven their efficiency.

For the evolutionary algorithm, we propose a method that is as simple as possible, while achieving efficiency. The population is evolved using a CHC genetic algorithm. We use this algorithm due to its simplicity and the fact that it has been shown to obtain good results for instance selection for the single-label case [20]. CHC [21] stands for *Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation*. The nontraditional CHC genetic algorithm differs from the traditional GAs in several ways [22]:

1. To obtain the next generation for a population of size N, the parents and offspring are put together, and the N best individuals are selected.
2. To avoid premature convergence, only different individuals, separated by a threshold Hamming distance – in our implementation $n/4$ bits, n being the length of the chromosome – are allowed to mate.
3. During crossover, two parents exchange exactly half of their non-matching bits. This operator is the *Half Uniform Crossover* (HUX) [21].
4. Mutation is not used during the regular evolution in order to avoid premature convergence or stagnation of the search. Instead, the population is reinitialized when the individuals are not diverse. In such a case, only the best individual is kept in the new population.

Each individual i of the population is a binary matrix $\mathbf{S}^i$, where S^i_{jl} means whether label l is selected for instance j. In the following sections, we describe the different aspects of the implementation.

3.2. Fitness of the individuals

Our population, P, is composed of n individuals, $\mathbf{S}^i$, as defined above. For the fitness measure of an individual $\mathbf{S}^i$, f_i, we use the simplest approach. The fitness first term is the classification performance of the dataset represented by the individual. As stated below, particularly in Section 4.2, we consider three different performance metrics in the comparison of the methods: F1, F1 macro-averaged, and F1 micro-averaged. Thus, the classification performance fitness term of an individual is given by

$$cf_i = F1(C(\mathbf{S}^i)) + F1_{\text{macro}}(C(\mathbf{S}^i)) + F1_{\text{micro}}(C(\mathbf{S}^i)), \tag{9}$$

where $M(C(\mathbf{S}^i))$ is the value of metric M when evaluated using classifier C, and the partial selection is given by $\mathbf{S}^i$. In evolutionary instance selection [20], it is common to add a reduction ability term to the fitness of the individual:

$$r_i = 1 - \frac{1}{pq} \sum_{j=1}^{p} \sum_{l=1}^{q} [\![S^i_{jl} = 0]\!], \tag{10}$$

where $[\![t]\!]$ is 1 if predicate t holds and 0 otherwise. The fitness of the individual, f_i, is the weighted combination of these two terms:

$$f_i = cf_i + \alpha r_i. \tag{11}$$

In the experiments, we study the behavior of our method in terms of α. These experiments show that the method using a reduction term provides better results than the same algorithm without reduction.

3.3. Initialization of the population

The initialization of the individuals can be carried out as normal for evolutionary instance selection. A certain probability, p_i, is set, and the initial population is obtained accordingly. However, our partial approach can use a more efficient initialization. As stated, most of the instances present a large imbalance ratio for many labels; thus, using the same probability for every label might be a suboptimal approach. As an example, Fig. 1 shows a histogram of the imbalance ratio for all the labels of all the datasets used in this paper. The figure shows clearly the large imbalance of most of the labels. For that reason, we developed a second method based on undersampling the majority class for every label in the initialization process.

For this method, we also use an initialization probability p_i; however, the process takes into account the imbalance ratio of the label. For label l initialization, we obtain the imbalance ratio of the label ir_l, as the ratio between the majority and minority class samples:

$$ir_l = \frac{\sum_p \mathbf{x}_i : y_l \in Y_l}{\sum_p \mathbf{x}_i : y_l \in \bar{Y}_l}, \tag{12}$$

for the usual case, when the minority class correspond to the case of label l being in the set of relevant labels. For the minority class initialization, p_i is used; for the majority class, we use p'_i, $p'_i = ir_l p_i$. With this method, all classes have an approximately balanced distribution for the initial population. In the experiments, this latter method is designated PARIS(US) for PARIS with undersampling (US) initialization.

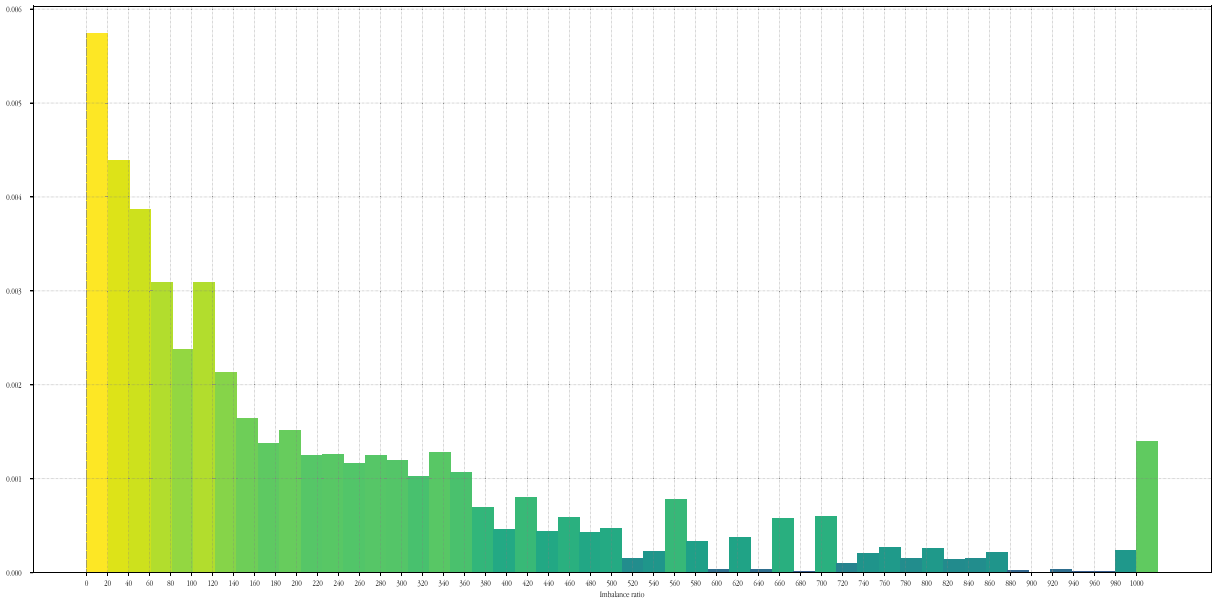


Fig. 1. Histogram of the label imbalance ratio distribution for all datasets used in the experiments. All ratios above 1,000 are summarized in the last bar for better legibility.

3.4. Scalability of our approach

One of the problems with evolutionary instance and training set selection is its computational cost, especially for the latter. Although evolutionary algorithms obtain very compact datasets where the classification performance usually matches or even improves that of the whole dataset, the time needed for the evolution is usually very long [23].

For single-label learning, random partition has been successfully applied for scaling up instance selection [24,25]. In that approach, the training set T is randomly partitioned into t subsets randomly, such as:

$$T = \bigcup_i t_i, \quad t_i \cap t_j = \emptyset, \forall i, j. \tag{13}$$

In the most common approach, a stratified sampling is used, where every subset t_i has a class distribution similar to T. For the multi-label case, we can also use the random partition for scaling up our method. However, the concept of the stratified partition for multi-label problems is not as straightforward as for single-label datasets. A few methods have been developed to improve the data partition for performance estimation [26,27]. However, we rely on simple random sampling with good results.

In the following experiments, we use a random partition of the data in subsets of 1,000 samples when the number of instances in the dataset is above this size. As each instance is in one and only one subset, the final result of the algorithm is the aggregation of the results of the partial selection for every subset.

4. Experimental setup

To make a fair comparison between the standard algorithms and our proposed approach, we selected a set of 45 datasets with a wide ranging number of patterns, features, and labels. A summary of these datasets is provided in Table 1. To estimate the storage reduction and classifier performance, we used tenfold cross-validation. The table shows a detailed description of the characteristics of the datasets, including the number of instances of the datasets, the number of inputs, the number of labels, the label cardinality, the label density, the label diversity, the proportion of distinct labels, and the MeanIR and CVIR measures. The label cardinality is measured as:

$$LCard(\mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m}|Y_i. \tag{14}$$

The label density is given by:

$$LDen(\mathcal{D}) = \frac{1}{|\mathcal{Y}|}LCard(\mathcal{D}). \tag{15}$$

The label diversity measures the number of distinct label sets present in the dataset:

$$LDiv(\mathcal{D}) = |\{Y|\exists \mathbf{x} : (\mathbf{x}, Y) \in \mathcal{D}\}. \tag{16}$$

The label diversity can be normalized by the number of instances to indicate the proportion of distinct label sets:

$$PLDiv(\mathcal{D}) = \frac{1}{|\mathcal{D}|}LDiv(\mathcal{D}). \tag{17}$$

The last two measures are used to study the imbalance of the labels of the datasets. Charte et al. [13] proposed new measures to evaluate the imbalance of a multi-label dataset due to the lack of information given by the known ones, such as density. First, an imbalance ratio per label (IRLbl) is defined for label $y \in \mathcal{Y}$, as the ratio between the majority label and label y:

$$IRLbl(y) = \frac{\arg\max_{y' \in \mathcal{Y}} \sum h(y', Y_i)}{\sum h(y, Y_i)}, \tag{18}$$

where

$$h(y, Y_i) = \begin{cases} 1, & y \in Y_i \\ 0, & y \notin Y_i. \end{cases} \tag{19}$$

IRLbl has a minimum value of 1 for the most frequent label and higher values for more imbalanced labels. In order to obtain a measure of the global imbalance of the dataset, the mean imbalance ratio (MeanIR) is also defined:

$$MeanIR = \frac{1}{|\mathcal{Y}|}\sum_{y \in \mathcal{Y}} IRLbl(y). \tag{20}$$

However, very different label distributions can have the same MeanIR value. Hence, the coefficient of variation of IRLbl (CVIR) is also defined to account for this. The CIVR indicates whether all labels suffer from a similar degree of imbalance or whether there are large differences among them. The measure is given by:

$$\begin{aligned} CVIR &= \frac{IRLbl_\sigma}{MeanIR}, \\ IRLbl_\sigma &= \sqrt{\sum_{y \in \mathcal{Y}} \frac{(IRLbl(y) - MeanIR)^2}{|\mathcal{Y}| - 1}}. \end{aligned} \tag{21}$$

Table 1
Characteristics of the datasets used in the experiments.

	Dataset	Instances	Inputs	#Labels	LCard	LDen	LDiv	PLDiv	MeanIR	CVIR
1	3s-bbc1000	352	1000	6	1.13	0.1875	15.0	0.0426	1.7182	0.2796
2	3s-guardian1000	302	1000	6	1.13	0.1876	14.0	0.0464	1.7733	0.3030
3	3s-inter3000	169	3000	6	1.14	0.1903	11.0	0.0651	1.7659	0.6096
4	3s-reuters1000	294	1000	6	1.13	0.1876	14.0	0.0476	1.7891	0.3130
5	corel16k005	13847	500	160	2.86	0.0179	1784.0	0.1288	34.9364	0.7282
6	delicious	16105	500	983	19.04	0.0194	3936.0	0.2446	71.1338	0.7405
7	emotions	593	72	6	1.87	0.3114	27.0	0.0455	1.4781	0.1796
8	enron	1702	1001	53	3.38	0.0637	753.0	0.4424	73.9528	1.9596
9	flags	194	43	7	3.39	0.4845	54.0	0.2784	2.2547	0.7648
10	foodtruck	407	31	12	2.29	0.1908	116.0	0.2850	7.0945	0.6914
11	genbase	662	1185	27	1.25	0.0464	32.0	0.0483	37.3146	1.4494
12	GnegativePseAAC	1392	440	8	1.05	0.1307	19.0	0.0136	18.4476	1.3945
13	IMDB-ECC-F	95424	1001	28	1.92	0.0687	3449.0	0.0361	26.2919	1.4853
14	LLOG-F	1460	1003	75	1.38	0.0183	278.0	0.2219	39.2669	1.3106
15	mediamill	43907	120	101	4.56	0.0451	3507.0	0.0832	256.4047	1.1749
16	medical	978	1449	45	1.25	0.0277	94.0	0.0961	89.5014	1.1476
17	OHSUMED-F	13929	1002	23	1.66	0.0723	1147.0	0.0823	7.8692	0.8920
18	PlantPseAAC	978	440	12	1.08	0.0899	32.0	0.0327	6.6904	0.7123
19	rcv1subset1	6000	47236	101	2.88	0.0285	837.0	0.1395	54.4923	2.0806
20	rcv1subset2	6000	47236	101	2.63	0.0261	800.0	0.1333	45.5138	1.7148
21	rcv1subset3	6000	47236	101	2.61	0.0259	783.0	0.1305	68.3326	2.9901
22	rcv1subset4	6000	47236	101	2.48	0.0246	698.0	0.1163	89.3713	2.3336
23	rcv1subset5	6000	47236	101	2.64	0.0262	782.0	0.1303	69.6815	2.6979
24	SLASHDOT-F	3782	1079	22	1.18	0.0537	156.0	0.0412	17.6931	2.4155
25	Stackex_coffee	225	1763	123	1.99	0.0162	149.0	0.6622	27.2415	0.5715
26	tmc2007	28596	49060	22	2.16	0.0981	1341.0	0.0469	15.1567	0.7633
27	Water-quality	1060	16	14	5.10	0.3644	824.0	0.7818	1.7671	0.3016
28	Yahoo_Arts	7484	23146	26	1.65	0.0636	599.0	0.0800	94.7379	3.8059
29	Yahoo_Computers	12444	34096	33	1.51	0.0457	428.0	0.0344	176.6952	1.9062
30	Yahoo_Education	12030	27534	33	1.46	0.0443	511.0	0.0425	168.1137	1.7756
31	Yahoo_Entertainment	12730	32001	21	1.41	0.0673	337.0	0.0265	64.4169	1.5398
32	Yahoo_Health	9205	30605	32	1.64	0.0514	335.0	0.0364	653.5306	1.9399
33	Yahoo_Recreation	12828	30324	22	1.43	0.0650	530.0	0.0413	12.2030	1.3899
34	Yahoo_Reference	8027	39679	33	1.17	0.0356	275.0	0.0343	461.8628	2.0073
35	Yahoo_Science	6428	37187	40	1.45	0.0362	457.0	0.0711	52.6318	1.6349
36	Yahoo_Social	12111	52350	39	1.28	0.0328	361.0	0.0298	257.7044	2.3431
37	Yahoo_Society	14512	31802	27	1.67	0.0619	1054.0	0.0726	302.0678	4.5633
38	yeast	2417	103	14	4.24	0.3026	198.0	0.0819	7.1968	1.8838
39	bibtex	7395	1836	159	2.40	0.0151	1654.0	0.2237	12.4983	0.4051
40	Corel5k	5000	499	373	3.52	0.0094	1453.0	0.2906	189.5676	1.5266
41	Stackex_chemistry	6861	540	175	2.11	0.0121	1452.0	0.2088	56.8779	0.8964
42	Stackex_chess	1675	585	227	2.42	0.0106	508.0	0.3038	85.7898	0.8167
43	Stackex_cooking	10491	577	400	2.25	0.0056	1712.0	0.1653	37.8576	0.6513
44	Stackex_cs	9270	635	274	2.57	0.0094	1489.0	0.1613	85.0023	0.7596
45	Stackex_philosophy	3971	842	233	2.28	0.0098	1072.0	0.2708	68.7532	0.7989

The comparison with other baseline methods has the problem that few methods have been developed for instance selection for multi-label datasets. We compared the methods described in the Related Work section and an evolutionary computation approach and found the latter outperformed the other methods. So, an evolutionary algorithm was used as the baseline method to compare with our method.

For this evolutionary algorithm, we designed a method that has proven its efficiency for single-label instance selection [28]. The population was evolved using an adaptation of the CHC genetic algorithm, as used for PARIS. The size of the population and the evolution time was the same for both the standard CHC algorithm and PARIS. The fitness function was also the same to avoid any bias towards any of the algorithms.

The source code, written in C and licensed under the GNU General Public License, used for all methods, as well as the partitions of the datasets, are freely available upon request from the authors.

4.1. Statistical tests

We used the Wilcoxon test [29] as the main statistical test for comparing pairs of algorithms. This test was chosen because it assumes limited commensurability and is safer than parametric tests because it does not assume normal distributions or the homogeneity of the variance. Furthermore, the empirical results [29] showed that this test was stronger than other tests.

The test [30] is formulated as follows: Let d_i be the difference between the error values of the methods on the ith dataset. These differences are ranked in accordance with their absolute values; in the case of a tie, an average rank is assigned. Let R^+ be the sum of the ranks of the datasets in which the second algorithm outperformed the first, and let be R^- the sum of the ranks of the datasets in which the first algorithm outperformed the second. The ranks for $d_i = 0$ are split evenly between the sums:

$$R^+ = \sum_{d_i>0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i) \tag{22}$$

and

$$R^- = \sum_{d_i<0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i). \tag{23}$$

Let T be the smaller of the two sums, and let N be the number of datasets. For a small N, there are tables providing the exact critical values for T. For a larger N, the statistic

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{24}$$

is distributed approximately following $N(0,1)$. In the tables of the results, we show the values of R^+ and R^- together with the p-value of the test.

In our experiments, we also compared groups of methods. In such cases, it was not advisable to use pairwise statistical tests such as the Wilcoxon test. Instead, we first applied the Iman–Davenport test to ascertain whether there were significant differences among the methods. The Iman–Davenport test is based on the χ_F^2 Friedman test, which compares the average ranks of k algorithms, but the former is more powerful. After applying the Iman–Davenport test, we can use any of the general procedures available for controlling the family-wise error in multiple hypothesis testing. One of the simplest methods of this type is Holm's procedure [29], which was the one used in our experiments.

In Holm's procedure, the best-performing algorithm in terms of Friedman's ranks is compared in a stepwise manner against the other methods. The test statistic for comparing the ith and jth methods is

$$z = \frac{(R_i - R_j)}{\sqrt{k(k+1)/6N}}. \tag{25}$$

The z value is used to find the corresponding probability from the normal distribution table, which is then compared against an appropriate α. The tests differ in how the α value is adjusted to compensate for multiple comparisons. The ordered p-values are denoted by $p_1, p_2, \ldots$, such that $p_1 \leq p_2 \leq \cdots \leq p_{k-1}$. Holm's step-down procedure starts with the most significant p-value. If p_1 is less than $\alpha/(k-1)$, then the corresponding hypothesis is rejected, and p_2 with $\alpha/(k-2)$ can be tested. If the second hypothesis is rejected, then the test proceeds to the third, and so on. As soon as a certain null hypothesis cannot be rejected, all remaining null hypotheses are also retained.

When the Iman–Davenport test rejects a null hypothesis, we can also proceed with a post hoc Nemenyi test [31], which compares groups of methods. The performances of two classifiers are considered significantly different if the corresponding average ranks differ by at least the following critical difference:

$$CD = q_\alpha \sqrt{k(k+1)\frac{6}{N}}, \tag{26}$$

where the critical value q_α is based on the studentized range statistic divided by $\sqrt{2}$, N is the number of datasets, and k is the number of compared methods. As a graphical representation of the Nemenyi test, we used the plots described by Demšar [29]. When comparing the algorithms against one another, we connected each group of algorithms that were not significantly different with a horizontal line. We also showed the critical difference above the graph. For all statistical tests, we used a significance level of 0.05.

4.2. Multi-label evaluation metrics

The evaluation of multi-label classification methods is relatively difficult because the prediction for an instance is a set of labels, and the result can be fully correct, partially correct (with different levels of correctness), or fully incorrect [32,33]. Thus, many different metrics have been proposed [1]. The metrics can be divided into two major groups: *example-based* (EB) metrics and *label-based* (LB) metrics. The former evaluate the learning system on each instance (example) separately and then obtain a unique measure averaging the value across the test set. The latter obtain the performance of the learning system for each class label separately and then return a unique measure by means of macro/micro-averaging across all class labels. Furthermore, the metrics can be focused on classification (using $h(\cdot)$) or ranking (using $f(\cdot)$). There are many metrics defined in the literature [1]. We restricted our study to classification metrics.

4.2.1. Example-based metrics

The most important example-based metrics are given below:

1. *Subset accuracy* evaluates the fraction of correctly classified examples, that is, the examples for which the predicted set of labels is identical to the set of relevant labels:

$$\text{subsetacc}(h) = \frac{1}{p}\sum_{i=1}^{p} [\![h(\mathbf{x}_i) = \mathcal{Y}_i]\!], \tag{27}$$

where $[\![\pi]\!]$ returns 1 if predicate π is true and 0 otherwise. This metric can be considered the multi-label counterpart of the accuracy metric. It can be overly strict, especially if the number of labels, q, is large. As is the case for accuracy, a perfect model would have a value of 1 for this metric.

2. *Hamming Loss* [34] evaluates the number of times a label not belonging to an instance is predicted or a label belonging to an instance is not predicted.

$$\text{Hamming Loss}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{q}|h(\mathbf{x}_i) \Delta \mathcal{Y}_i|, \tag{28}$$

where Δ indicates the symmetric difference between two sets and corresponds to the XOR operation in Boolean logic. With respect to this metric, the performance is optimal when its value is zero, and higher values signify a decrease in performance.

3. *Accuracy* [34,35] is defined as the proportion of the number of correctly predicted labels to the total number (predicted and actual) of labels for an instance:

$$\text{Accuracy}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|h(\mathbf{x}_i) \bigcap \mathcal{Y}_i|}{|h(\mathbf{x}_i) \bigcup \mathcal{Y}_i|}. \tag{29}$$

With respect to this metric, the performance is optimal when its value is one, and lower values indicate a decrease in performance.

4. *Precision* [34,35] is the average proportion of the number of correctly predicted labels to the total number of predicted labels:

$$\text{Precision}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|h(\mathbf{x}_i) \bigcap \mathcal{Y}_i|}{|h(\mathbf{x}_i)|}. \tag{30}$$

With respect to this metric, the performance is optimal when its value is one, and lower values indicate a decrease in performance.

5. *Recall* is the average proportion of the number of correctly predicted labels to the total number of relevant labels:

$$\text{Recall}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|h(\mathbf{x}_i) \bigcap \mathcal{Y}_i|}{|\mathcal{Y}_i|}. \tag{31}$$

With respect to this metric, the performance is optimal when its value is one, and lower values indicate a decrease in performance.

6. F^β is an integrated version of recall and precision with a balancing factor β. The most common used value is $\beta = 1$. The metric is given by

$$F^\beta(h) = \frac{(1+\beta^2)\cdot Precision(h)\cdot Recall(h)}{\beta^2 \cdot Precision(h) + Recall(h)}. \tag{32}$$

With respect to this metric, the performance is optimal when its value is one, and lower values indicate a decrease in performance.

4.2.2. Label-based metrics

For label-based metrics, we first must define the four basic quantities that characterize the binary classification of each label based on function $h(\cdot)$:

$$\begin{aligned} TP_j &= |\{\mathbf{x}_i | y_j \in \mathcal{Y}_i \wedge y_j \in h(\mathbf{x}_i), 1 \leq i \leq p\}|, \\ FP_j &= |\{\mathbf{x}_i | y_j \notin \mathcal{Y}_i \wedge y_j \in h(\mathbf{x}_i), 1 \leq i \leq p\}|, \\ TN_j &= |\{\mathbf{x}_i | y_j \notin \mathcal{Y}_i \wedge y_j \notin h(\mathbf{x}_i), 1 \leq i \leq p\}|, \\ FN_j &= |\{\mathbf{x}_i | y_j \in \mathcal{Y}_i \wedge y_j \notin h(\mathbf{x}_i), 1 \leq i \leq p\}|. \end{aligned} \tag{33}$$

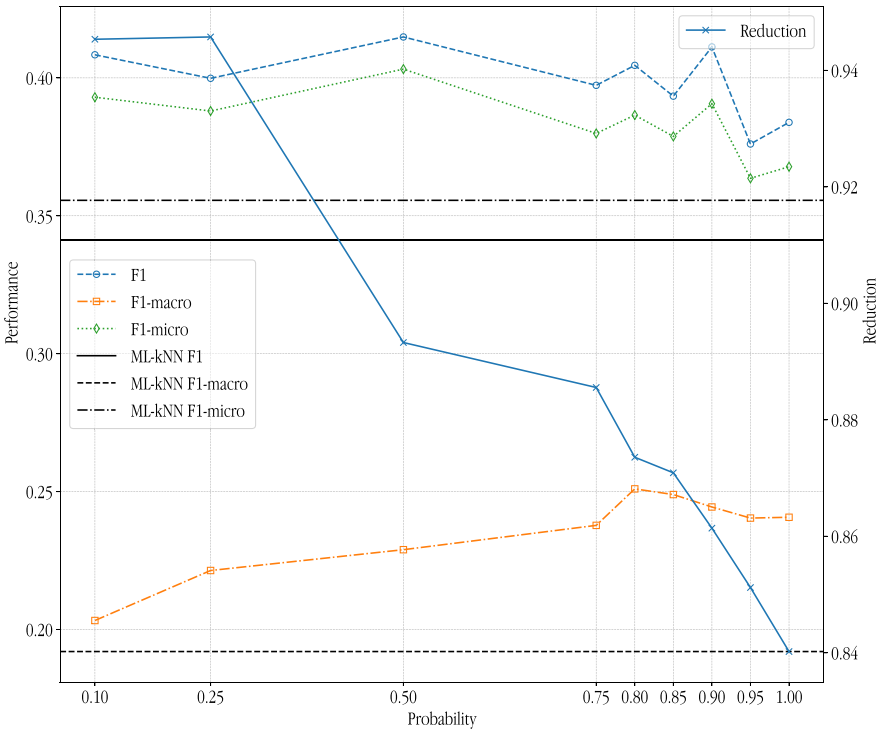


Fig. 2. The behavior of the reduction and classification performance (using F1, F1-macro, and F1-micro metrics) of PARIS(US) in terms of the initialization probability using undersampling.

With these four quantities, most of the binary classification metrics can be adapted to multi-label problems. Considering any measure $B(TP_j, FP_j, TN_j, FN_j)$, the label-based classification metric can be obtained either by micro-averaging or macro-averaging [36]:

- Macro-averaging:

$$B_{\text{macro}} = \frac{1}{q}\sum_{j=1}^{q} B(TP_j, FP_j, TN_j, FN_j). \tag{34}$$

- Micro-averaging:

$$B_{\text{micro}} = B\left(\sum_{j=1}^{q} TP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} TN_j, \sum_{j=1}^{q} FN_j\right). \tag{35}$$

It is evident that accuracy metric is the same macro- and micro-averaged. The use of different metrics is justified because they represent the performance of the models from different points of view. However, due to the multiplicity of metrics, we restrict ourselves to some of them. First, we discarded the subset accuracy because it is too restrictive, and we discarded the Hamming loss because in many datasets, due to the low label density, it is too optimistic. Recall and precision should not be considered alone, and accuracy is not a good metric for imbalanced datasets. Thus, we used F1 for the example-based, and the macro-averaged, F1_{macro}, and micro-averaged, F1_{micro}, for the label-based versions.

5. Experimental results and discussion

We performed experiments with the standard ML-kNN, BR, CC, and our approach. First, we carried out a set of experiments to study the behavior of our proposal for a better understanding of how it worked. This part of the study only used ML-kNN due to the large number of experiments involved. Then, we performed experiments comparing PARIS for ML-kNN, BR, and CC with their standard counterparts. In the following sections, we report the results of the experiments, the results of the statistical tests, and the discussion of the results. The study of the behavior of our method only used the training set to avoid contaminating the comparison experiments.

5.1. Hyper-parameters

First, we carried out experiments to test the influence on PARIS of the initialization for the population. We considered two aspects, as we developed two ways of initializing the population (see Section 3.3), the standard one considering a initialization probability common for all the labels and the undersampling strategy. The first aspect to study was whether there were differences in the performance between the two approaches. A second aspect was studying the performance of the algorithm depending on the initialization probability for both cases. Figs. 2 and 3 show the average value of the three classification metrics as the initialization probability was changed. The values of the probabilities tested for the two methods were not the same, as the initialization probability did not have the same meaning in both cases.

As expected, the behavior of the reduction was different in both cases. For the standard approach, the dependence of the final reduction was almost linear with respect to the initialization of the population. However, for the undersampling initialization, that was not the case. We must bear in mind that for the undersampling initialization, the actual initialization probability for each label also depended on the imbalance ratio of the label; thus, the correlation between the initialization probability and the actual number of 1's in the population was not as simple as in the standard initialization case.

Comparing both methods, the undersampling initialization outperformed the standard method in terms of reduction. That was an expected result as the former always initially selected fewer instances than the latter.

Regarding the performance, in both cases, retaining a larger percentage of instances did not guarantee a better classification performance. It seems that there were many useless, noisy, or redundant instances. In fact, for F1 and F1-micro, the average performance tended to improve with a larger reduction. On the other hand, F1-macro

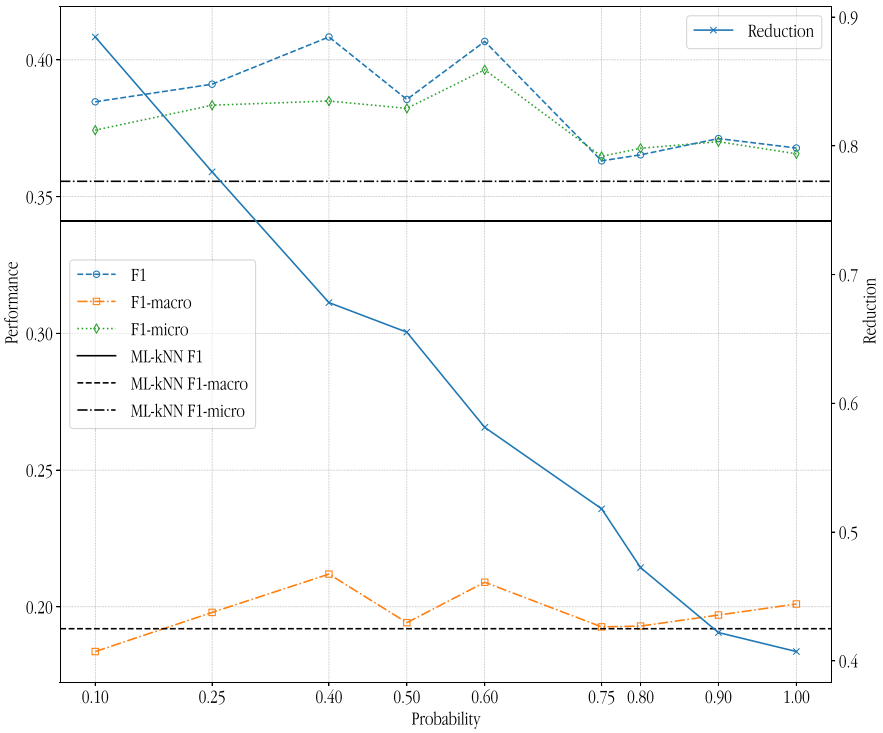


Fig. 3. The behavior of the reduction and classification performance (using F1, F1-macro, and F1-micro metrics) of PARIS in terms of the initialization probability.

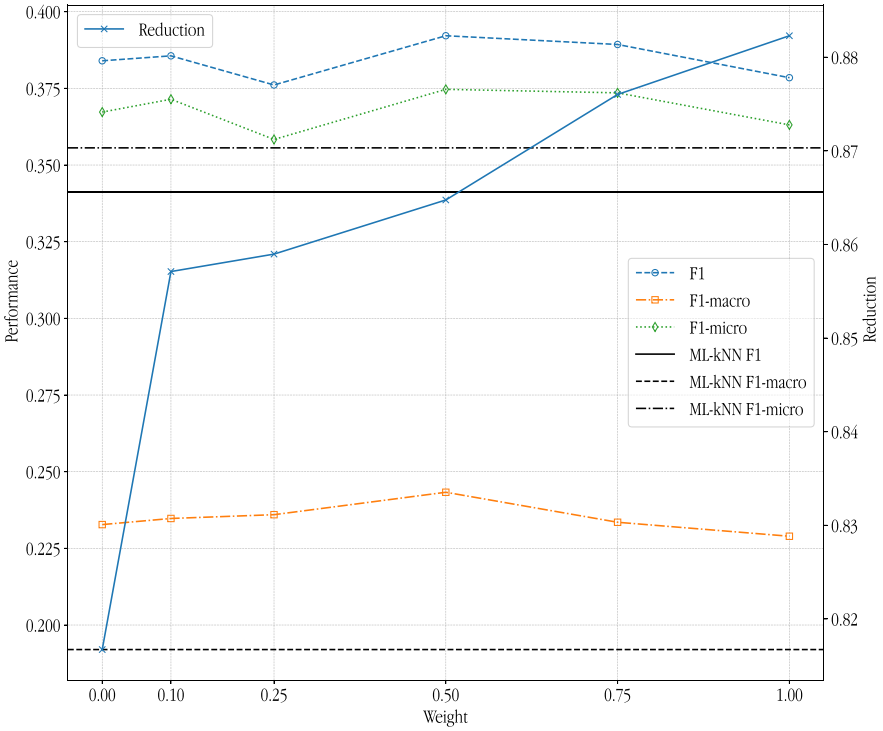


Fig. 4. The behavior of the reduction and classification performance (using F1, F1-macro, and F1-micro metrics) of PARIS in terms of the reduction weight in the fitness function.

showed a better performance for smaller reduction percentages. Both methods achieved similar classification performance values, but as undersampling initialization obtained better reduction, it was chosen for the remaining experiments. In the remaining we dropped the US for simplicity, so when PARIS is used we are actually referring to PARIS with undersampling initialization.

Then, we carried out experiments to test the optimal α value for our approach. Fig. 4 shows the behavior of the partial instance selection in terms of α. We tested values from $\alpha = 0$, meaning that the reduction was not considered for the fitness, to $\alpha = 1$, meaning that the reduction and classification performance had the same relative weight. However, we must bear in mind that both terms of the fitness function, the

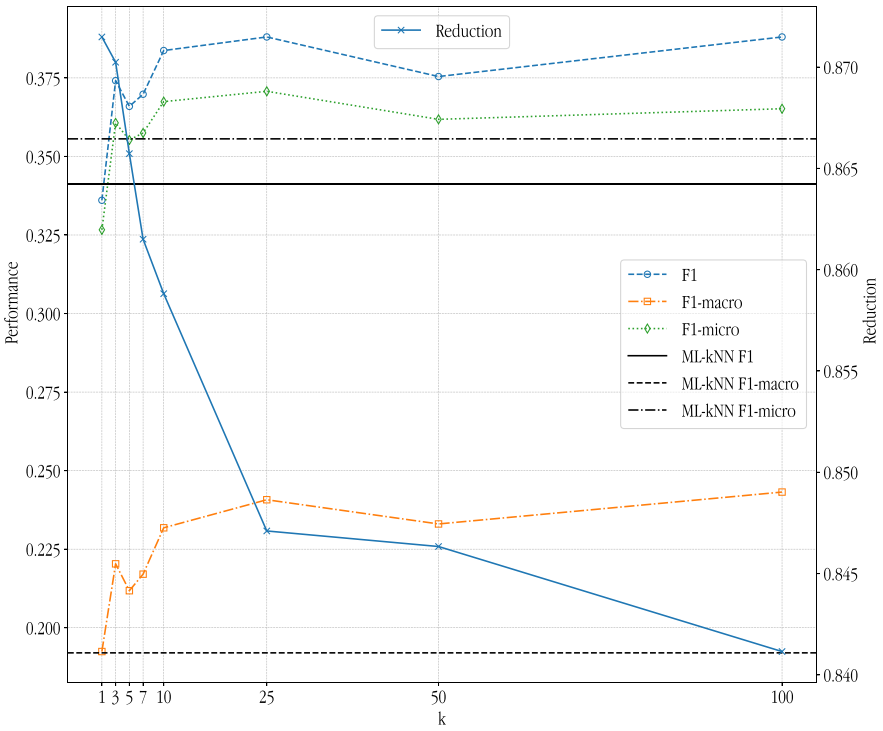


Fig. 5. The behavior of the reduction and classification performance (using F1, F1-macro, and F1-micro metrics) of PARIS in terms of the maximum k value.

classification performance and the reduction, did not have the same range. The reduction had the range $[0, 1]$ in the actual results of the experiments. On the other hand, the classification performance was the sum of three metrics, the F1, F1-macro, and F1-micro. In the actual values of the experiments, this factor took values from $[0, 3]$. This meant that even for $\alpha = 1$, the reduction was less relevant than the classification performance for the fitness function. As the reduction ability of our method was high, we did not pursue larger reductions.

As our target was the classification performance and not the reduction, we chose, for the remaining experiments, the value that achieved the best classification performance value for each algorithm. The results of this experiment showed that the values in the interval $\alpha \in [0.4, 0.6]$ were suitable for most datasets.

Our final experiment in this stage was aimed at studying the effect of the maximum value of k on the performance of PARIS. Fig. 5 shows the average reduction and classification performance when different values of k were chosen within the set $k = \{1, 3, 5, 7, 25, 50, 100\}$. The behavior in terms of the classification performance was quite stable for values above 10 neighbors. In terms of reduction, the ability to remove instances decreased as the number of neighbors increased (see Fig. 5).

5.2. Comparison with standard methods

Our next step was the comparison with the standard version of the tested multi-label methods. For ML-kNN, we tested ML-kNN with no instance selection, (ML-kNN in the plots), ML-kNN with standard instance selection (ML-kNN(IS)), ML-kNN with our approach (ML-kNN(PARIS)), that with the local k interpretation (ML-kNN(PARIS+LK)), the standard BR [32] (BR), that with our approach (BR(PARIS)), CC [37], and that with our approach (CC(PARIS)). We added three additional standard methods, RAndom k-labELsets [36] (RAkEL) with overlapping (RAkELo) and non-overlapping (RAkELd) labelsets, label powerset [36] (LP), and MLARAM [38], for a more thorough comparison. For all cases, a decision tree trained with C4.5 algorithm was used as the binary classifier. Decision trees were used because they are fast, accurate, and do not need to finetune hyper-parameters for a good performance. For the standard methods we used scikit-multilearn implementation [39].

In order to retain a fair comparison, we also cross-validated the optimal values of the initialization probability and α for the standard instance selection approach. We obtained the best values for an initialization probability of 0.25 and $\alpha = 1$. The detailed results are shown in Tables 2, 3, and 4 for F1, F1-macro, and F1-micro metrics, respectively, and in Table 5 for reduction.

Our first comparison was carried out using the Iman–Davenport and Nemenyi tests. The Iman–Davenport showed significant differences for all the experiments. The Nemenyi test is illustrated in Fig. 6 for F1, F1-macro, and F1-micro metrics, reduction, and all the tested methods. For F1, the Nemenyi test showed that PARIS always improved the results of the standard method. ML-kNN(PARIS) and BR(PARIS) were significantly better than ML-kNN and BR. CC(PARIS) achieved a better Friedman rank than CC, although below the critical difference. It is also worth mentioning that CC(PARIS) and ML-kNN(PARIS) obtained the best average ranking, above all the remaining standard methods. ML-kNN(PARIS+LK) was worse than ML-kNN(PARIS), although it was significantly better than ML-kNN. The standard instance selection was improved over ML-kNN, but it was worse than our approach.

For F1-macro metrics, the results showed a similar trend with an even better behavior of PARIS. CC(PARIS), BR(PARIS), and ML-kNN(PARIS) achieved the best average ranks. The large improvements in BR(PARIS) and ML-kNN(PARIS) over BR and ML-kNN, respectively, are noteworthy. Finally, for F1-micro, the results were somewhat different, with RAkEL among the top performing methods. However, the improvement with PARIS over the standard version of each method was still present. The reduction comparison is also shown in Fig. 6. The reduction results showed that PARIS not only performed better than the standard IS in terms of classification metrics, but it also achieved a better reduction ability.

A further comparison was carried out with the Holm procedure for the best performing method, CC(PARIS) for all metrics, and the remaining models. Fig. 7 shows this comparison. For the three metrics, the general advantage of our method was clear, with significant differences in most cases.

Table 2
The results for the standard methods and our approach for F1 metric.

Dataset	ML-kNN(PARIS)	ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
3s-bbc1000	0.1731	0.2265	0.0556	0.2366	0.1487	0.4026	0.1594	0.0000	0.3432	0.2404	0.3299
3s-guardian1000	0.4219	0.1042	0.0000	0.2661	0.2314	0.4285	0.1659	0.0215	0.4216	0.2269	0.3275
3s-inter3000	0.3866	0.0000	0.0000	0.0889	0.0392	0.4457	0.0882	0.0000	0.3824	0.2960	0.2588
3s-reuters1000	0.2353	0.0909	0.0333	0.2300	0.1667	0.1897	0.0667	0.0000	0.2728	0.2096	0.2514
bibtex	0.2533	0.2485	0.2522	0.3848	0.2729	0.4432	0.4069	0.1733	0.1913	0.1303	0.4053
corel16k005	0.1924	0.0321	0.0868	0.0333	0.1275	0.2099	0.0959	0.0035	0.1422	0.0550	0.0702
Corel5k	0.1372	0.0540	0.1441	0.0736	0.1185	0.2314	0.1416	0.0024	0.1957	0.0319	0.1350
delicious	0.2345	0.2223	0.2061	0.0726	0.1461	0.2253	0.1280	0.0986	0.2345	0.0702	0.1208
emotions	0.7174	0.6621	0.6541	0.5903	0.6496	0.3556	0.5777	0.6458	0.6796	0.5301	0.5612
enron	0.4994	0.4741	0.5413	0.4309	0.4838	0.5196	0.5186	0.4463	0.5698	0.3568	0.4193
flags	0.7439	0.7400	0.6898	0.6886	0.7429	0.6761	0.7250	0.7487	0.6834	0.7547	0.6459
foodtruck	0.5132	0.6004	0.5599	0.4638	0.4903	0.5862	0.6128	0.5986	0.6313	0.5188	0.4952
genbase	0.9850	0.9540	0.9789	0.9937	0.9938	0.9888	0.9913	0.3212	0.9850	0.9900	0.9899
GnegativePseAAC	0.7216	0.6589	0.6785	0.4855	0.6999	0.7523	0.6017	0.4839	0.7559	0.5979	0.4595
IMDB-ECC-F	0.2448	0.0199	0.1061	0.2805	0.2737	0.3578	0.0596	0.0001	0.2448	0.1778	0.2821
LLOG-F	0.1842	0.0615	0.1332	0.2145	0.2505	0.2278	0.1507	0.0082	0.1714	0.0950	0.2174
mediamill	0.6212	0.6527	0.5743	0.5681	0.5591	0.5534	0.6062	0.6094	0.2672	0.1829	0.5608
medical	0.6688	0.7080	0.6801	0.7381	0.6997	0.7273	0.7347	0.1727	0.6416	0.7974	0.7583
OHSUMED-F	0.3084	0.2062	0.2638	0.4527	0.3739	0.5240	0.4001	0.1531	0.3139	0.4196	0.4958
PlantPseAAC	0.3524	0.1143	0.2463	0.2692	0.3607	0.3266	0.1403	0.0408	0.3511	0.2323	0.2076
rcv1subset1	0.3618	0.2601	0.3001	0.2680	0.1852	0.3229	0.3394	0.0075	0.3134	0.3517	0.3795
rcv1subset2	0.3760	0.3110	0.2846	0.3304	0.2429	0.2369	0.3977	0.0000	0.3391	0.3490	0.3900
rcv1subset3	0.3401	0.2006	0.2555	0.3157	0.2589	0.2506	0.4084	0.0008	0.3055	0.4004	0.3943
rcv1subset4	0.3753	0.2886	0.3279	0.4178	0.3131	0.2472	0.4461	0.0024	0.3433	0.4518	0.4422
rcv1subset5	0.3411	0.2623	0.2389	0.3416	0.3148	0.2597	0.3819	0.0052	0.3219	0.3781	0.3904
SLASHDOT-F	0.3941	0.2627	0.3410	0.3929	0.2295	0.2841	0.1637	0.0132	0.5064	0.4583	0.4213
Stackex_chemistry	0.0949	0.0435	0.0939	0.1586	0.1310	0.1403	0.0633	0.0049	0.1809	0.0648	0.2036
Stackex_chess	0.1855	0.0923	0.1676	0.2467	0.1742	0.1778	0.2738	0.0646	0.2269	0.0713	0.2980
Stackex_coffee	0.0911	0.1477	0.0000	0.2324	0.2210	0.1276	0.2099	0.0000	0.2898	0.3037	0.2197
Stackex_cooking	0.0517	0.0550	0.0773	0.3101	0.2065	0.0825	0.3115	0.0040	0.1456	0.0474	0.3352
Stackex_cs	0.1135	0.0862	0.1654	0.2766	0.1728	0.1822	0.1347	0.0111	0.2048	0.0902	0.3352
Stackex_philosophy	0.1101	0.0521	0.1666	0.2824	0.2223	0.1800	0.2874	0.0171	0.1958	0.0538	0.3182
tmc2007	0.5803	0.5818	0.6173	0.5959	0.5271	0.6841	0.5967	0.3796	0.5125	0.6191	0.6287
Water-quality	0.6087	0.5613	0.5581	0.5300	0.4573	0.3497	0.5728	0.5682	0.5957	0.5518	0.5781
Yahoo_Arts	0.3092	0.0902	0.2383	0.3866	0.3464	0.2937	0.3179	0.0535	0.2706	0.3623	0.4364
Yahoo_Computers	0.5882	0.3879	0.3880	0.4814	0.5013	0.4523	0.4900	0.4364	0.2957	0.4951	0.5225
Yahoo_Education	0.3144	0.0607	0.1988	0.3170	0.3793	0.2837	0.2652	0.0289	0.2624	0.2968	0.3847
Yahoo_Entertainment	0.3690	0.3104	0.2798	0.4399	0.4316	0.2556	0.4233	0.1373	0.3642	0.4103	0.5101
Yahoo_Health	0.7516	0.3438	0.3686	0.5972	0.5833	0.4233	0.6487	0.4007	0.3865	0.6192	0.6178
Yahoo_Recreation	0.5990	0.1249	0.2452	0.4305	0.3647	0.3774	0.3996	0.0842	0.2322	0.4273	0.4598
Yahoo_Reference	0.3319	0.3890	0.4979	0.4414	0.5459	0.4910	0.4248	0.3043	0.2957	0.4435	0.4351
Yahoo_Science	0.5540	0.1308	0.1779	0.3818	0.3139	0.2357	0.3539	0.0400	0.2075	0.3869	0.4018
Yahoo_Social	0.6470	0.3603	0.5047	0.5423	0.6058	0.4705	0.5485	0.2174	0.3605	0.5610	0.5704
Yahoo_Society	0.4658	0.2906	0.4137	0.5562	0.4706	0.4258	0.4020	0.1926	0.2387	0.4260	0.5605
yeast	0.6655	0.6554	0.6686	0.5393	0.6300	0.6640	0.6350	0.6216	0.6576	0.4841	0.5310
Mean	0.4048	0.2929	0.3213	0.3861	0.3702	0.3793	0.3748	0.1805	0.3674	0.3559	0.4168

Table 3
The results for the standard methods and our approach for F1-macro metric.

Dataset	ML-kNN(PARIS)	ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
3s-bbc1000	0.1970	0.2409	0.0926	0.2325	0.1018	0.3761	0.0784	0.0000	0.2171	0.2792	0.2898
3s-guardian1000	0.3150	0.1494	0.0000	0.2622	0.1405	0.2938	0.0826	0.0556	0.1787	0.2870	0.2987
3s-inter3000	0.2592	0.0000	0.0000	0.1068	0.0333	0.2275	0.0704	0.0000	0.1981	0.2263	0.2347
3s-reuters1000	0.2526	0.1273	0.0476	0.1368	0.0764	0.1672	0.0392	0.0000	0.1511	0.2674	0.2414
bibtex	0.1933	0.1582	0.1272	0.2565	0.1076	0.1921	0.2976	0.0290	0.1067	0.1545	0.2917
corel16k005	0.0791	0.0139	0.0182	0.0167	0.0248	0.0793	0.0404	0.0009	0.0604	0.0588	0.0495
Corel5k	0.0411	0.0132	0.0234	0.0143	0.0148	0.0413	0.0349	0.0004	0.0249	0.0335	0.0343
delicious	0.1249	0.0660	0.0503	0.0288	0.0441	0.0683	0.0270	0.0316	0.1249	0.0851	0.0580
emotions	0.7007	0.6402	0.6473	0.5699	0.6139	0.2069	0.5684	0.6714	0.6694	0.5502	0.5616
enron	0.2036	0.0818	0.1021	0.1728	0.1365	0.0989	0.1679	0.1016	0.1170	0.1620	0.1358
flags	0.5786	0.5799	0.5593	0.5752	0.5962	0.4744	0.5891	0.6171	0.4767	0.6233	0.5693
foodtruck	0.2827	0.0999	0.1628	0.1650	0.1815	0.2399	0.1558	0.1867	0.1580	0.2368	0.2313
genbase	0.6067	0.6089	0.6125	0.6543	0.6296	0.6250	0.6222	0.1816	0.6132	0.5926	0.6543
GnegativePseAAC	0.3706	0.3755	0.3550	0.3012	0.3199	0.4253	0.2178	0.2044	0.4095	0.3242	0.3132
IMDB-ECC-F	0.0948	0.0160	0.0658	0.0592	0.0245	0.0591	0.0573	0.0000	0.0948	0.0545	0.0873
LLOG-F	0.0664	0.0131	0.0237	0.0795	0.0346	0.0497	0.0581	0.0033	0.0293	0.0550	0.0937
mediamill	0.3540	0.3594	0.2495	0.1521	0.0823	0.0691	0.0699	0.0970	0.1079	0.1081	0.1598
medical	0.2373	0.2460	0.2109	0.3104	0.2206	0.2454	0.3256	0.0424	0.1213	0.3125	0.3396
OHSUMED-F	0.1597	0.1233	0.1396	0.3852	0.1846	0.2824	0.3115	0.0936	0.2368	0.3585	0.4034
PlantPseAAC	0.2007	0.0674	0.0769	0.1921	0.1281	0.1323	0.0518	0.0271	0.1072	0.1962	0.1429

(*continued on next page*)

Table 3 (*continued*).

Dataset	ML-kNN(PARIS)	ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
rcv1subset1	0.1649	0.1489	0.1275	0.1170	0.0660	0.0801	0.1830	0.0029	0.1245	0.2203	0.2148
rcv1subset2	0.1750	0.1733	0.1265	0.1184	0.0456	0.0142	0.2037	0.0000	0.1227	0.2058	0.2033
rcv1subset3	0.1780	0.0703	0.1160	0.1113	0.0647	0.0473	0.2195	0.0003	0.1229	0.2227	0.1997
rcv1subset4	0.1779	0.1104	0.0945	0.1301	0.0508	0.0135	0.1851	0.0003	0.1179	0.2345	0.2090
rcv1subset5	0.1524	0.0944	0.0911	0.1194	0.0695	0.0170	0.1905	0.0005	0.1324	0.1887	0.2181
SLASHDOT-F	0.2232	0.1289	0.1740	0.2997	0.0820	0.1490	0.1059	0.0152	0.3031	0.3015	0.3475
Stackex_chemistry	0.0445	0.0119	0.0114	0.0820	0.0080	0.0057	0.0189	0.0008	0.0358	0.0723	0.1170
Stackex_chess	0.0505	0.0073	0.0157	0.0770	0.0259	0.0074	0.0966	0.0087	0.0182	0.0755	0.0982
Stackex_coffee	0.0310	0.0226	0.0000	0.0691	0.0290	0.0140	0.0375	0.0000	0.0288	0.0492	0.0683
Stackex_cooking	0.0257	0.0147	0.0146	0.1304	0.0365	0.0019	0.1261	0.0025	0.0234	0.0705	0.1497
Stackex_cs	0.0547	0.0178	0.0229	0.1173	0.0175	0.0174	0.0504	0.0023	0.0430	0.0796	0.1557
Stackex_philosophy	0.0423	0.0030	0.0082	0.0857	0.0187	0.0041	0.0851	0.0020	0.0171	0.0698	0.1233
tmc2007	0.4101	0.3893	0.4102	0.4585	0.2671	0.4911	0.4158	0.0451	0.3788	0.4941	0.5007
Water-quality	0.5487	0.4565	0.5213	0.4811	0.3441	0.2853	0.4835	0.4997	0.5542	0.5176	0.5544
Yahoo_Arts	0.1364	0.0442	0.0792	0.2369	0.1187	0.0310	0.2460	0.0242	0.1349	0.3141	0.2670
Yahoo_Computers	0.4174	0.1386	0.1812	0.2031	0.1326	0.0261	0.2373	0.1056	0.1142	0.3103	0.2206
Yahoo_Education	0.1383	0.0653	0.0969	0.2087	0.0881	0.0144	0.2173	0.0258	0.0853	0.2701	0.1930
Yahoo_Entertainment	0.2048	0.1693	0.1647	0.3083	0.1508	0.0210	0.2847	0.0941	0.1770	0.3628	0.3603
Yahoo_Health	0.3807	0.1594	0.1518	0.1980	0.1600	0.0215	0.2870	0.1222	0.1548	0.3229	0.2772
Yahoo_Recreation	0.4275	0.1441	0.2008	0.3539	0.1512	0.1198	0.3971	0.1123	0.1513	0.4207	0.3895
Yahoo_Reference	0.1021	0.0916	0.0757	0.2207	0.0988	0.0392	0.2046	0.0450	0.0736	0.2463	0.2027
Yahoo_Science	0.3196	0.0514	0.0577	0.2647	0.0684	0.0160	0.2756	0.0131	0.0806	0.2862	0.2826
Yahoo_Social	0.3513	0.0792	0.1110	0.2396	0.0992	0.0396	0.2730	0.0281	0.0848	0.3163	0.2879
Yahoo_Society	0.1379	0.0642	0.0866	0.2343	0.0928	0.0248	0.2709	0.0361	0.1262	0.3028	0.2829
yeast	0.4436	0.4121	0.4167	0.3883	0.3464	0.4244	0.3382	0.3268	0.4467	0.4019	0.4010
Mean	0.2368	0.1566	0.1538	0.2206	0.1406	0.1396	0.2066	0.0857	0.1746	0.2516	0.2559

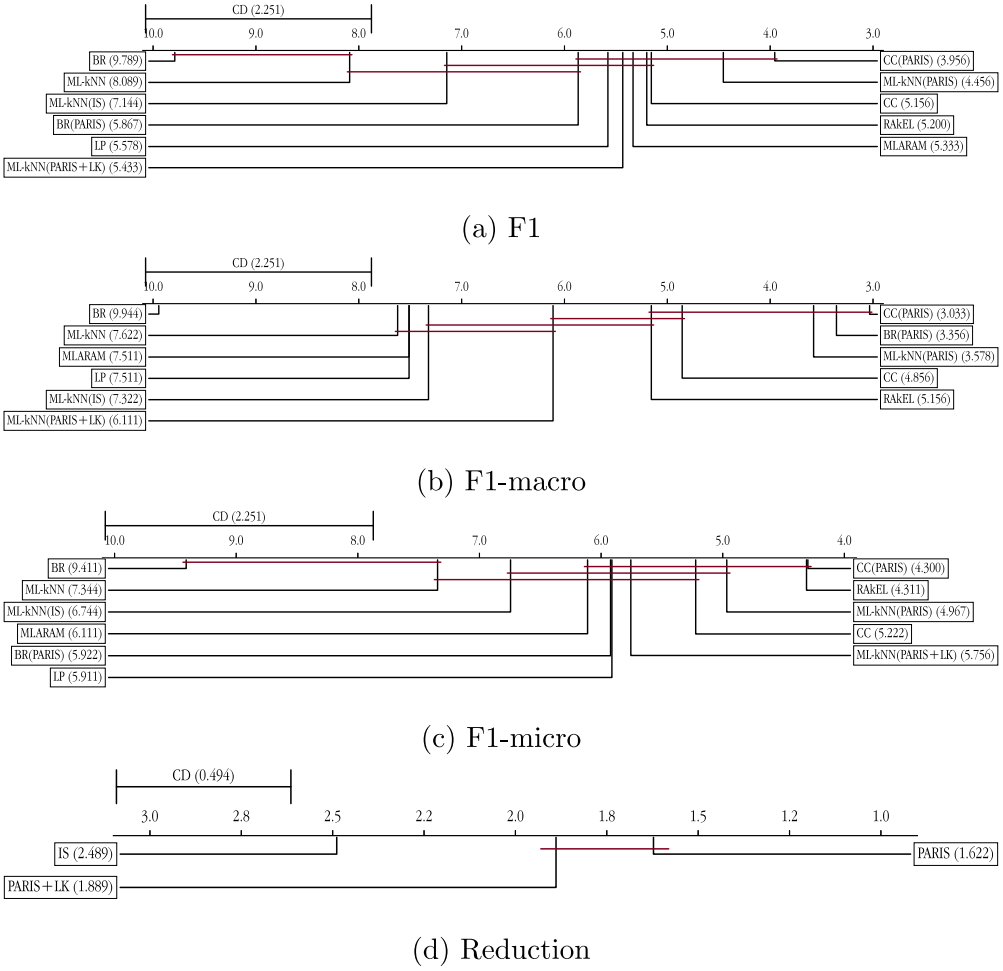


Fig. 6. The Nemenyi test for the comparison of ML-kNN, IS, and PARIS with the standard and undersampled initialization for F1, F1-macro, and F1-micro metrics and the reduction.

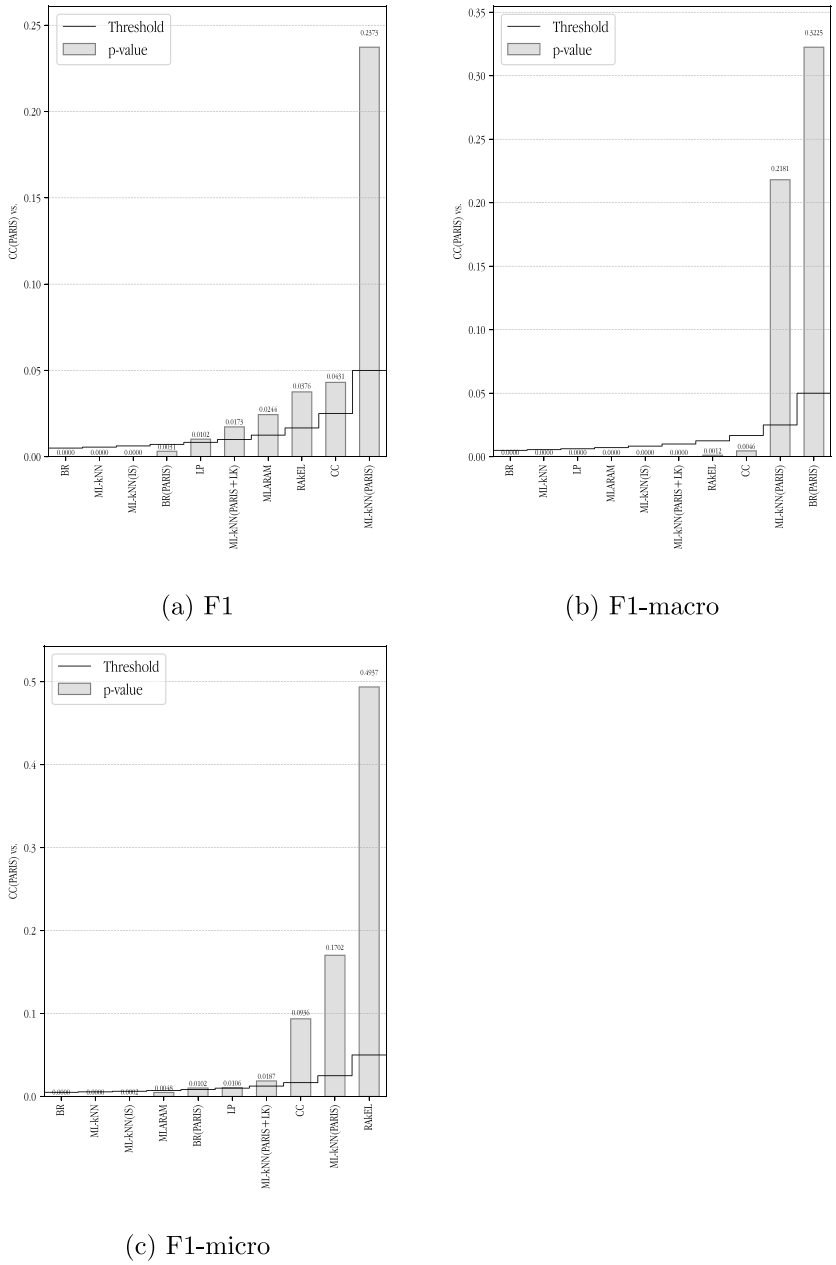


Fig. 7. The Holm test for the comparison of ML-kNN, IS, CC, MLARAM, BR, RAkEL, LP, and PARIS with the standard and undersampled initialization for F1, F1-macro, and F1-micro metrics and the reduction.

The relative performance of PARIS and the standard methods is illustrated in Fig. 8. This figure shows the performance of the standard method in the x-axis and that of PARIS in the y-axis for every dataset. Values above the main diagonal show a better performance of PARIS. The figure shows the comparison between PARIS and ML-kNN, Fig. 8(a), PARIS and ML-kNN(IS), Fig. 8(b), BR, Fig. 8(c), and CC, Fig. 8(d). The three classification metrics are shown. For the four comparisons, a clear advantage of PARIS is shown, with the vast majority of the points above the main diagonal.

In terms of F1, F1-macro, and F1-micro, the comparison with ML-kNN shows most of the points above the main diagonal, which is a remarkable achievement taking into account that this performance is coupled with a large reduction. The comparison of IS versus PARIS was even better, with a better performance of PARIS for most of the datasets. The same results are shown for the comparison of CC against CC(PARIS) and BR against BR(PARIS).

A plot was created to show the comparison of the average value of reduction and classification performance for PARIS and IS. This plot is shown in Fig. 9. The plot illustrated that PARIS was also better for the combination of the reduction and classification performance. This aspect was also tested using the Holm procedure. Fig. 10 shows the Holm test for this combined measure of the reduction and classification performance. The plot shows the comparison of the best performing algorithm, ML-kNN(PARIS), against ML-kNN(PARIS+LK), and ML-kNN(IS). The Holm test found that ML-kNN(PARIS) was significantly better than the other two methods for the three metrics combined with the reduction.

As a final step, we carried out pairwise comparisons using the Wilcoxon test. Tables 6, 7, and 4 show the comparison in terms of the Wilcoxon test for F1, F1-macro, and F1-micro metrics, respectively. The table shows the win/loss record of the method in the column against the method in the row, the p-value of the Wilcoxon test, and the R^+/R^- values of the Wilcoxon test. The Iman–Davenport p-value is also shown.

Table 4
The results for the standard methods and our approach for F1-micro metric.

Dataset	ML-kNN(PARIS)	ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
3s-bbc1000	0.2200	0.2778	0.1200	0.2449	0.1481	0.3711	0.1446	0.0000	0.3471	0.2778	0.3133
3s-guardian1000	0.4040	0.1600	0.0000	0.2807	0.2319	0.3918	0.1791	0.0513	0.3951	0.2889	0.3111
3s-inter3000	0.3235	0.0000	0.0000	0.1200	0.0556	0.4390	0.1429	0.0000	0.3889	0.2745	0.2571
3s-reuters1000	0.2549	0.1429	0.0571	0.2128	0.1613	0.2051	0.0851	0.0000	0.2737	0.2667	0.2737
bibtex	0.2154	0.2957	0.2859	0.3806	0.2474	0.3205	0.4114	0.1989	0.1750	0.1273	0.3717
corel16k005	0.1838	0.0421	0.1040	0.0367	0.1177	0.1833	0.1054	0.0035	0.1390	0.0523	0.0853
Corel5k	0.1252	0.0734	0.1664	0.0852	0.1223	0.1888	0.1447	0.0034	0.1885	0.0314	0.1462
delicious	0.2188	0.2250	0.2089	0.0610	0.1418	0.2224	0.1338	0.0900	0.2188	0.0694	0.1125
emotions	0.7040	0.6698	0.6605	0.5841	0.6311	0.3406	0.5784	0.6857	0.6783	0.5614	0.5630
enron	0.4869	0.4813	0.5254	0.4010	0.4462	0.4782	0.5034	0.5022	0.5432	0.2870	0.3943
flags	0.7407	0.7344	0.7015	0.6901	0.7302	0.6613	0.7328	0.7460	0.6818	0.7463	0.6667
foodtruck	0.4696	0.4748	0.4918	0.3924	0.4238	0.5412	0.5106	0.5333	0.5422	0.4677	0.4286
genbase	0.9697	0.9565	0.9693	0.9878	0.9877	0.9814	0.9814	0.4571	0.9697	0.9814	0.9818
GnegativePseAAC	0.6881	0.7550	0.6713	0.4651	0.6923	0.6725	0.6538	0.6018	0.7417	0.5379	0.4575
IMDB-ECC-F	0.2212	0.0261	0.1217	0.2554	0.2460	0.3441	0.0808	0.0001	0.2212	0.1834	0.2579
LLOG-F	0.1680	0.0870	0.1538	0.2280	0.2353	0.1818	0.1781	0.0120	0.1667	0.0918	0.2197
mediamill	0.5875	0.6168	0.5348	0.5301	0.5129	0.5009	0.5643	0.5689	0.2532	0.1778	0.5161
medical	0.6178	0.6960	0.6417	0.7200	0.6992	0.6642	0.7797	0.2685	0.6513	0.7129	0.7372
OHSUMED-F	0.3075	0.2647	0.3219	0.4904	0.3622	0.4510	0.4471	0.1973	0.3032	0.3915	0.5032
PlantPseAAC	0.3370	0.1778	0.2500	0.2912	0.3575	0.3146	0.1986	0.0678	0.3574	0.2311	0.2036
rcv1subset1	0.3457	0.2875	0.3108	0.2535	0.1830	0.2697	0.3519	0.0086	0.2945	0.3469	0.3449
rcv1subset2	0.3459	0.3258	0.3045	0.2819	0.1878	0.2190	0.3872	0.0000	0.3113	0.3226	0.3607
rcv1subset3	0.3142	0.2117	0.2745	0.2711	0.1966	0.2083	0.3950	0.0013	0.2763	0.3650	0.3429
rcv1subset4	0.3340	0.2936	0.3204	0.3578	0.2432	0.1811	0.4161	0.0027	0.3007	0.4185	0.3756
rcv1subset5	0.3154	0.2632	0.2528	0.2935	0.2593	0.2508	0.3789	0.0050	0.2970	0.3548	0.3275
SLASHDOT-F	0.3745	0.3281	0.3847	0.4654	0.2177	0.2493	0.2003	0.0212	0.4836	0.4027	0.4221
Stackex_chemistry	0.0894	0.0577	0.1118	0.1840	0.1208	0.1256	0.0764	0.0080	0.1604	0.0641	0.2141
Stackex_chess	0.1703	0.1121	0.1761	0.2724	0.1783	0.1654	0.2969	0.1065	0.2261	0.0723	0.2913
Stackex_coffee	0.1059	0.1707	0.0000	0.2444	0.2143	0.1356	0.1875	0.0000	0.2286	0.2549	0.2385
Stackex_cooking	0.0504	0.0745	0.1015	0.3140	0.1953	0.0783	0.3297	0.0050	0.1298	0.0450	0.3033
Stackex_cs	0.1077	0.1163	0.1965	0.3004	0.1706	0.1733	0.1477	0.0152	0.1946	0.0809	0.3286
Stackex_philosophy	0.1039	0.0594	0.1538	0.2937	0.1932	0.1518	0.3057	0.0239	0.1793	0.0534	0.2882
tmc2007	0.5535	0.5804	0.6052	0.5788	0.5156	0.6358	0.5893	0.3765	0.4810	0.6061	0.5905
Water-quality	0.5929	0.5531	0.5734	0.5308	0.4449	0.3568	0.5706	0.5680	0.6045	0.5470	0.5756
Yahoo_Arts	0.2730	0.1113	0.2321	0.3569	0.3172	0.2604	0.3725	0.0687	0.2505	0.3669	0.3926
Yahoo_Computers	0.4884	0.4142	0.3522	0.4499	0.4605	0.4088	0.5096	0.4480	0.2478	0.5116	0.4587
Yahoo_Education	0.2935	0.1027	0.2318	0.3262	0.3645	0.2550	0.3565	0.0473	0.2520	0.3537	0.3701
Yahoo_Entertainment	0.3502	0.3250	0.2982	0.4679	0.3843	0.2272	0.4924	0.1911	0.3001	0.4258	0.4922
Yahoo_Health	0.6551	0.3794	0.3920	0.5326	0.5570	0.3702	0.6147	0.4043	0.3580	0.5874	0.5553
Yahoo_Recreation	0.5830	0.1776	0.2454	0.4743	0.3328	0.2958	0.4759	0.1365	0.2173	0.4557	0.4912
Yahoo_Reference	0.3685	0.4277	0.4738	0.5055	0.5229	0.4741	0.5184	0.3916	0.2575	0.5169	0.4716
Yahoo_Science	0.5190	0.1552	0.1887	0.4113	0.2765	0.2119	0.4038	0.0602	0.1871	0.3930	0.4024
Yahoo_Social	0.6786	0.4047	0.4557	0.5896	0.5616	0.4353	0.6053	0.2918	0.2908	0.5971	0.5241
Yahoo_Society	0.4044	0.2996	0.3119	0.4935	0.4110	0.3766	0.4284	0.2221	0.2135	0.4056	0.4870
yeast	0.6453	0.6496	0.6586	0.5428	0.6197	0.6422	0.6196	0.6148	0.6371	0.4718	0.5248
Mean	0.3846	0.3120	0.3243	0.3833	0.3484	0.3469	0.3908	0.2001	0.3470	0.3506	0.3994

Table 5
The results for the standard method instance selection and our approach for the reduction ability.

Dataset	IS	PARIS	PARIS+LK
3s-bbc1000	0.8766	0.8370	0.7300
3s-guardian1000	0.9004	0.8807	0.7263
3s-inter3000	0.8158	0.8257	0.6765
3s-reuters1000	0.8826	0.8580	0.7197
bibtex	0.8341	0.9767	0.9759
corel16k005	0.7747	0.9750	0.9714
Corel5k	0.8549	0.9848	0.9848
delicious	0.7560	0.9688	0.9688
emotions	0.8668	0.6248	0.6357
enron	0.8739	0.9107	0.8703
flags	0.8448	0.7471	0.5057
foodtruck	0.9044	0.8427	0.8345
genbase	0.9193	0.9582	0.9487
GnegativePseAAC	0.9497	0.8679	0.7976
IMDB-ECC-F	0.7674	0.8915	0.8915
LLOG-F	0.8789	0.9689	0.9684
mediamill	0.7618	0.9442	0.9427
medical	0.8682	0.9704	0.9495
OHSUMED-F	0.7976	0.9076	0.8933

(*continued on next page*)

Table 5 (*continued*).

Dataset	IS	PARIS	PARIS+LK
PlantPseAAC	0.9409	0.9094	0.9169
rcv1subset1	0.8200	0.9454	0.9517
rcv1subset2	0.8244	0.9505	0.9583
rcv1subset3	0.8115	0.9502	0.9582
rcv1subset4	0.7878	0.9525	0.9592
rcv1subset5	0.7959	0.9527	0.9579
SLASHDOT-F	0.8419	0.9510	0.9170
Stackex_chemistry	0.7891	0.9806	0.9807
Stackex_chess	0.8890	0.9866	0.9831
Stackex_coffee	0.8663	0.9704	0.9297
Stackex_cooking	0.7933	0.9908	0.9910
Stackex_cs	0.7643	0.9850	0.9850
Stackex_philosophy	0.8671	0.9838	0.9843
tmc2007	0.7573	0.8538	0.8538
Water-quality	0.8470	0.5311	0.5084
Yahoo_Arts	0.8374	0.9101	0.8978
Yahoo_Computers	0.9204	0.9311	0.9300
Yahoo_Education	0.7969	0.9293	0.9297
Yahoo_Entertainment	0.8510	0.8981	0.8941
Yahoo_Health	0.8917	0.9194	0.9195
Yahoo_Recreation	0.8508	0.9031	0.8970
Yahoo_Reference	0.8992	0.9401	0.9426
Yahoo_Science	0.8434	0.9448	0.9437
Yahoo_Social	0.8931	0.9418	0.9476
Yahoo_Society	0.9358	0.9404	0.9009
yeast	0.8676	0.7565	0.6586
Mean	0.8469	0.9078	0.8820

Table 6
Comparison of PARIS and the standard methods for F1 metric.

		ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS)	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
Mean		0.2929	0.3213	0.3861	0.3702	0.3793	0.3748	0.1805	0.4048	0.3674	0.3559	0.4168
Ranks		8.0889	7.1444	5.1556	5.5778	5.3333	5.2000	9.7889	4.4556	5.4333	5.8667	3.9556
ML-kNN	w/l		30/14	36/9	33/12	35/10	36/9	4/40	37/8	37/8	33/12	37/8
	p		0.0091✓	0.0001✓	0.0002✓	0.0001✓	0.0000✓	0.0000✗	0.0000✓	0.0001✓	0.0018✓	0.0000✓
	R^+/R^-		748.5/286.5	870.0/165.0	846.0/189.0	867.0/168.0	891.0/144.0	46.5/988.5	939.0/96.0	873.0/162.0	794.0/241.0	910.0/125.0
ML-kNN(IS)	w/l			31/14	32/13	32/13	31/14	7/36	35/10	34/11	26/19	33/12
	p			0.0009✓	0.0015✓	0.0013✓	0.0011✓	0.0000✗	0.0000✓	0.0054✓	0.0832	0.0000✓
	R^+/R^-			813.0/222.0	798.0/237.0	802.0/233.0	806.0/229.0	57.5/977.5	898.0/137.0	764.0/271.0	671.0/364.0	877.0/158.0
CC	w/l				16/29	18/27	23/22	8/37	24/21	19/26	21/24	35/10
	p				0.0982	0.4947	0.7477	0.0000✗	0.3402	0.3635	0.2165	0.0000✓
	R^+/R^-				371.0/664.0	457.0/578.0	489.0/546.0	57.0/978.0	602.0/433.0	437.0/598.0	408.0/627.0	894.0/141.0
LP	w/l					22/23	23/22	4/41	28/17	23/22	20/25	32/13
	p					0.8611	0.6234	0.0000✗	0.0489✓	0.9595	0.3756	0.0028✓
	R^+/R^-					533.0/502.0	561.0/474.0	31.0/1004.0	692.0/343.0	513.0/522.0	439.0/596.0	782.0/253.0
MLARAM	w/l						23/22	5/40	24/21	21/24	22/23	27/18
	p						0.9595	0.0000✗	0.3879	0.5163	0.3879	0.0416✓
	R^+/R^-						513.0/522.0	67.0/968.0	594.0/441.0	460.0/575.0	441.0/594.0	698.0/337.0
RAkEL	w/l							3/42	25/20	20/25	23/22	29/16
	p							0.0000✗	0.1348	0.5763	0.6886	0.0047✓
	R^+/R^-							17.0/1018.0	650.0/385.0	468.0/567.0	482.0/553.0	768.0/267.0
BR	w/l								43/2	40/5	37/8	38/7
	p								0.0000✓	0.0000✓	0.0000✓	0.0000✓
	R^+/R^-								1024.0/11.0	970.0/65.0	917.0/118.0	986.0/49.0
ML-kNN(PARIS)	w/l									14/28	15/30	26/19
	p									0.1810	0.0066✗	0.5610
	R^+/R^-									399.0/636.0	277.0/758.0	569.0/466.0
ML-kNN(PARIS+LK)	w/l										21/24	28/17
	p										0.5236	0.0266✓
	R^+/R^-										461.0/574.0	714.0/321.0
BR(PARIS)	w/l											32/13
	p											0.0003✓
	R^+/R^-											837.0/198.0
Iman-Davenport $p-$ value: 0.0000												

Significant differences are marked with a ✓for the column being better than the row and with a ✗for the row being better than the column. The tables show that the p-value of the Iman–Davenport test, as stated above, was below the 0.05 threshold for all cases.

Regarding F1 metric, shown in Table 6, the first remarkable result is that all three models performed better with PARIS than with the standard version. Moreover, ML-kNN performed better than ML-kNN(IS). Furthermore, CC(PARIS) was better than all the remaining methods with the exception of ML-kNN(PARIS). For F1-macro, see Table 7; similar results were obtained with the difference that the performance of BR(PARIS) was much better than that for F1 metric. For F1-micro, PARIS was better all the remaining methods. CC(PARIS) was significantly better than all the methods with the exception of ML-kNN(PARIS). BR(PARIS) and ML-kNN(PARIS) achieved better performance than BR and ML-kNN, respectively (see Table 8).

5.3. Comparison with other instance selection methods

Our method is a new approach to instance selection for multi-label learning, so it should be compared with previous instance selection methods. However, as it was stated in Section 2, there are few methods for this task developed so far. In the previous section we compared PARIS with instance selection carried out by a genetic algorithm which is the algorithm reported in [16]. In this section we add the other methods developed so far. Arnaiz-González et al. [14] developed an instance selection methodology adapting the concept of local set, used

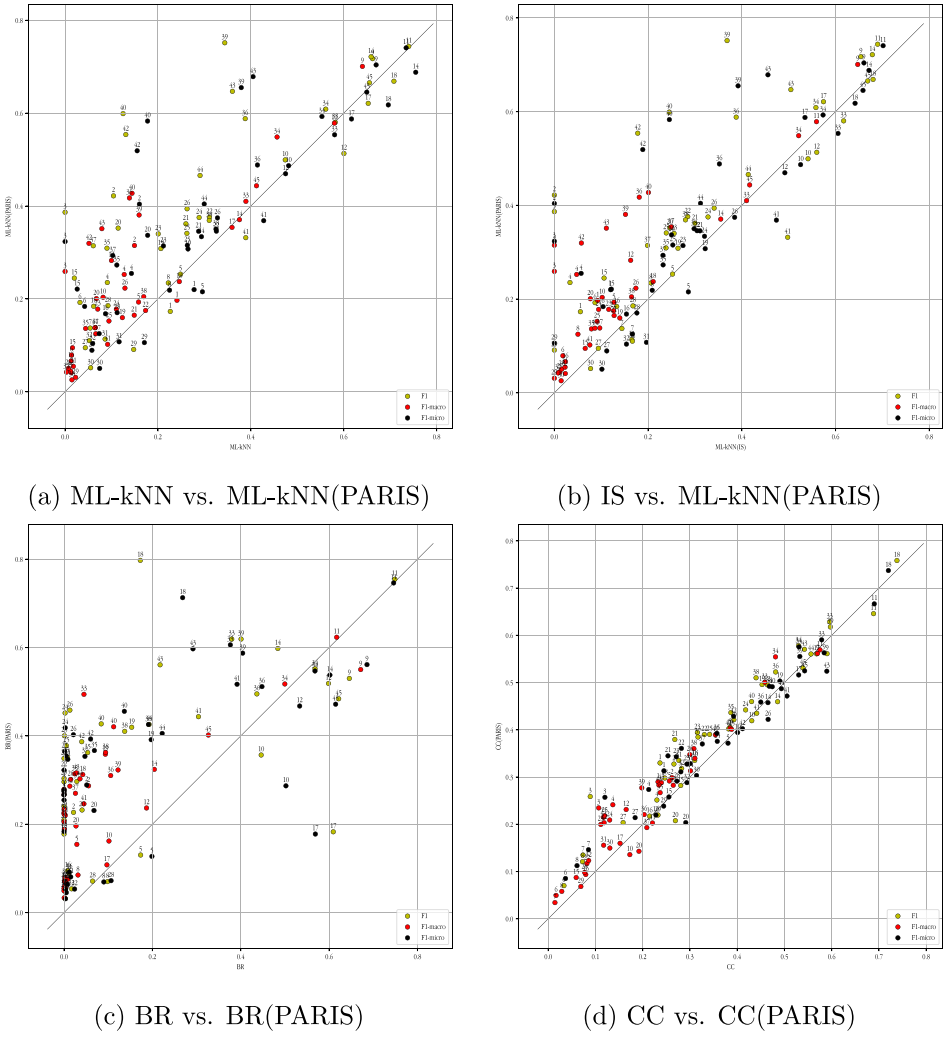


Fig. 8. Comparison between ML-kNN(PARIS) and ML-kNN, (a), ML-KNN(PARIS) and IS, (b), CC(PARIS) and BR, (c), and BR(PARIS) and BR, (d), for the three metrics of the classification performance.

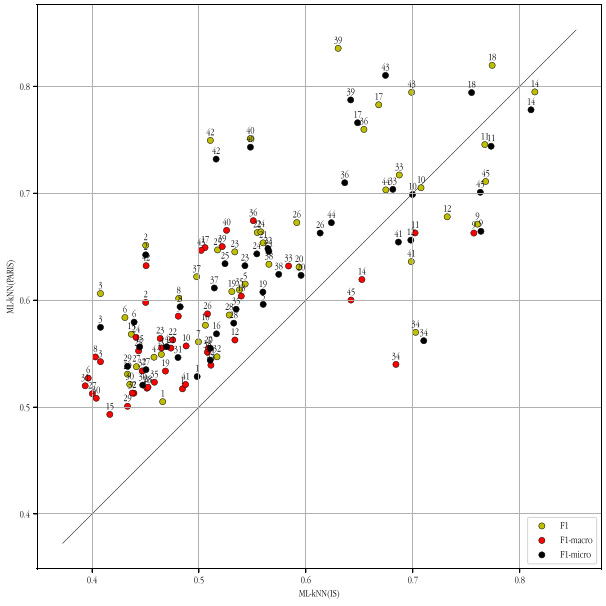


Fig. 9. Comparison between IS and ML-kNN(PARIS) for the average value of F1, F1-macro, and F1-micro metrics and the reduction.

in the Iterative Case Filtering (ICF) method [8] for single-label datasets, to the multi-label case. The author proposed two algorithms HDLSSm and HDLSBo that proved a good performance in terms of classification and ranking metrics. However, HDLSSm method was not included in the comparison because its average reduction rate was bellow 5% and thus cannot be fairly compared with the other methods. For the same reason we also excluded a multi-label version of edited k-NN [40]. Finally, Charte et al. [13,41] developed an adaptation of random undersampling for multi-label problems. Although it is not a purely instance selection method we include it in this comparison as its result is a subset of the instances in the dataset. For random undersampling we used the best sampling ratio of 10% as found in Charte et al. [13].

Our first comparison was carried out using the Iman–Davenport and Nemenyi tests. The Iman–Davenport showed significant differences for all the experiments. The Nemenyi test is illustrated in Fig. 11 for F1, F1-macro, and F1-micro metrics, reduction, and all the tested methods. The plot shows results for evolutionary instance selection (IS) [16], cooperative coevolutionary instance selection (COOP) [11], local set based instance selection (HDLSBo) [14] and random undersampling (RUS) [13].

For F1, the Nemenyi test showed that PARIS always improved the results of the other methods. ML-kNN(PARIS) was significantly better than COOP, IS, RUS and HDLSBo. CC(PARIS) and B(PARIS) achieved a better Friedman rank than the remaining methods, although below the critical difference. The results for F1-macro were even better for our

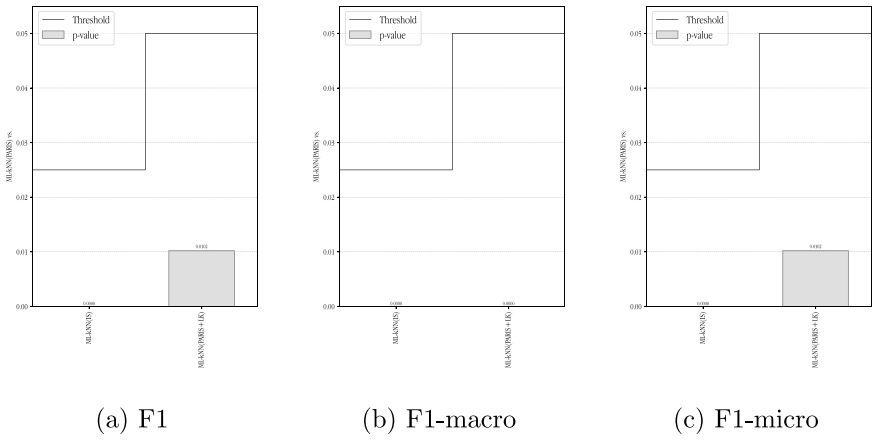


Fig. 10. The Holm test for the comparison of IS and PARIS for the average value of F1, F1-macro, and F1-micro metrics and reduction.

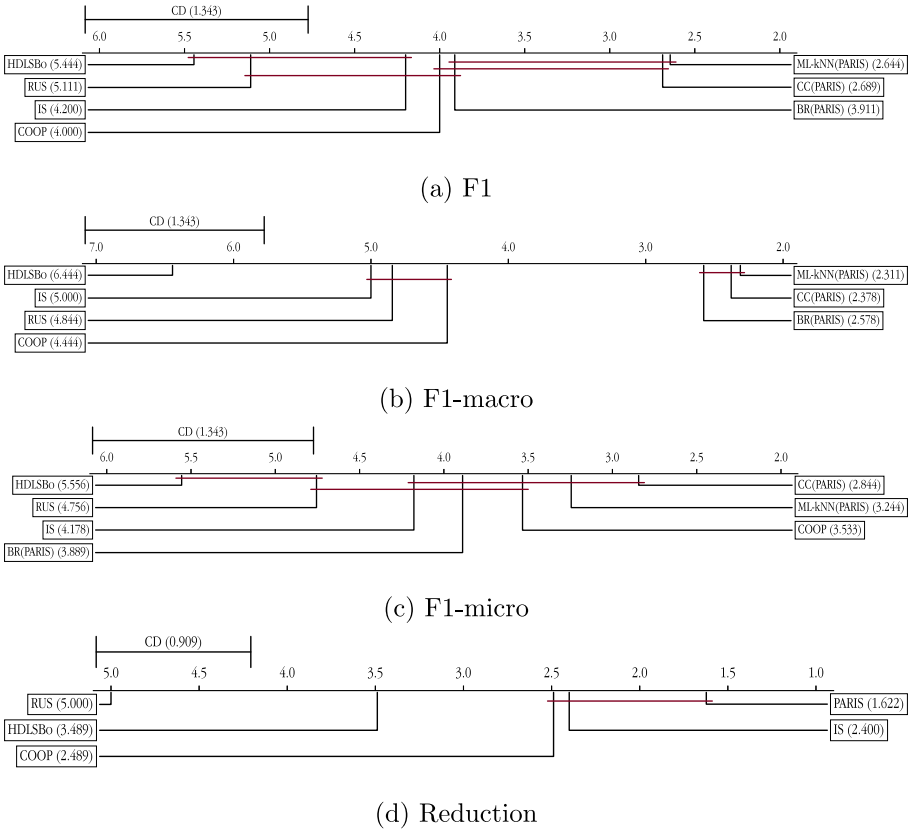


Fig. 11. The Nemenyi test for the comparison of other instance selection algorithms presented in the literature and PARIS for F1, F1-macro, and F1-micro metrics and reduction.

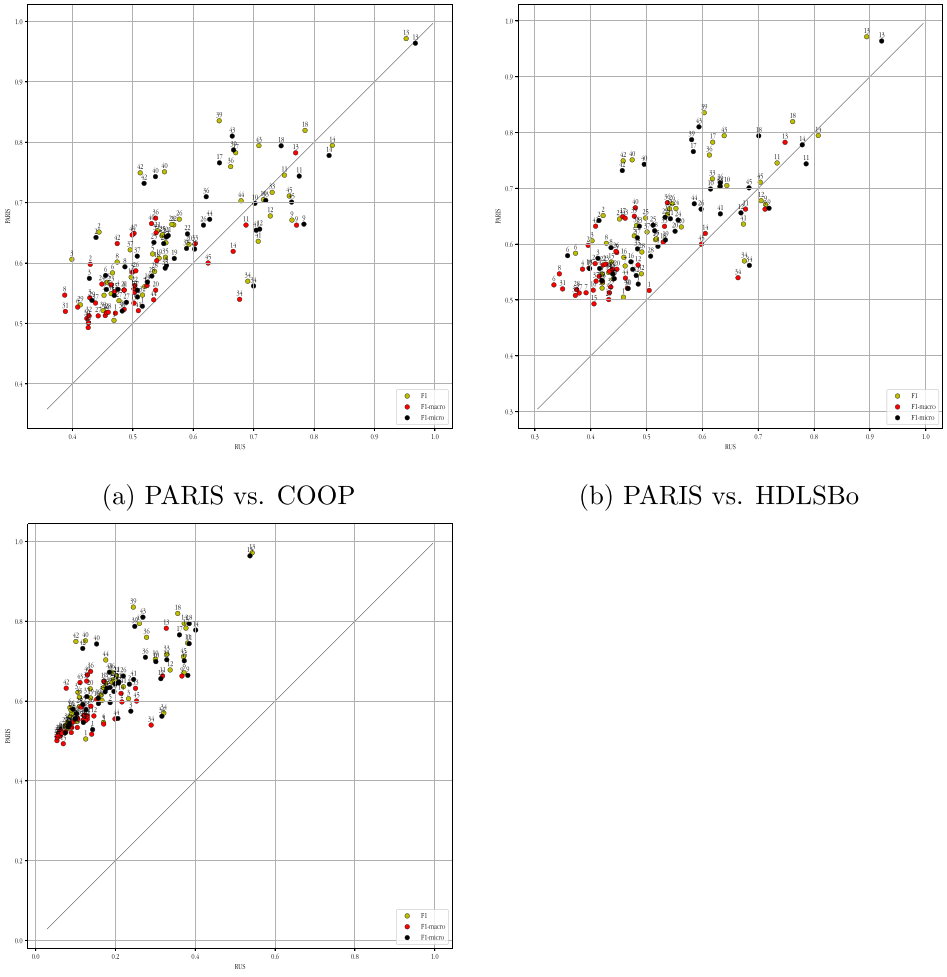
proposal. All three versions, ML-kNN(PARIS), CC(PARIS) and B(PARIS), outperformed significantly the remaining algorithms. For F1-micro the differences were less marked, although CC(PARIS) and ML-kNN(PARIS) were still the two best performing methods. In terms of reduction, see Fig. 11(d), the results showed that the better classification performance of PARIS was achieved while keeping the best reduction ability of the tested methods. The reduction ability of RUS is understandably low as it was not pretended as a instance selection method.

Fig. 12 shows the comparison of the averaged value of classification performance and reduction for the three studied metrics. We show the combined classification performance and reduction due to the fact that the very different reduction ratios of the methods made a direct comparison in terms of performance only unfair. The figure shows a clear advantage of our proposal with better combined performance for the vast majority of datasets.

Finally, a Holm test was carried out to compare the best method for the classification metrics and reduction. The results are shown in Fig. 13. For F1 metric, ML-kNN(PARIS) improved the performance of all other methods with the exception of CC(PARIS). Similar results were obtained for F1-macro, with ML-kNN(PARIS) achieving better performance than COOP, RUS, IS and HDLSBo. For F1-micro CC(PARIS) was the best method and it outperformed the remaining ones with the exception of COOP and ML-kNN(PARIS). For reduction, the Holm test corroborated the results of the Nemenyi test and showed that PARIS achieved significantly better results than the remaining methods.

5.4. Study of the results

An interesting aspect to study in any new method is its behavior as a function of the different characteristics of the datasets. In order to gain



(a) PARIS vs. COOP

(b) PARIS vs. HDLSBo

(c) PARIS vs. RUS

Fig. 12. Comparison between PARIS and the other instance selection algorithms presented in the literature for the three metrics of the classification performance combined with reduction.

Table 7
Comparison of PARIS and the standard methods for F1-macro metric.

		ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS)	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
Mean		0.1566	0.1538	0.2206	0.1406	0.1396	0.2066	0.0857	0.2368	0.1746	0.2516	0.2559
Ranks		7.6222	7.3222	4.8556	7.5111	7.5111	5.1556	9.9444	3.5778	6.1111	3.3556	3.0333
ML-kNN	w/l		25/19	36/9	23/22	21/24	35/10	6/38	39/6	34/11	39/6	39/6
	p		0.5200	0.0000✓	0.4196	0.4876	0.0012✓	0.0000✗	0.0000✓	0.0054✓	0.0000✓	0.0000✓
	R^+/R^-		574.5/460.5	890.0/145.0	446.0/589.0	456.0/579.0	804.0/231.0	107.5/927.5	994.0/41.0	764.0/271.0	950.0/85.0	966.0/69.0
ML-kNN(IS)	w/l			33/12	22/23	18/27	33/12	4/39	43/2	32/13	38/7	41/4
	p			0.0000✓	0.3233	0.2811	0.0004✓	0.0000✗	0.0000✓	0.0096✓	0.0000✓	0.0000✓
	R^+/R^-			904.0/131.0	430.0/605.0	422.0/613.0	831.0/204.0	79.5/955.5	1032.0/3.0	747.0/288.0	961.0/74.0	981.0/54.0
CC	w/l				7/38	11/34	20/25	5/40	26/19	15/30	31/14	37/7
	p				0.0000✗	0.0001✗	0.4196	0.0000✗	0.2964	0.0012✗	0.0009✓	0.0000✓
	R^+/R^-				34.0/1001.0	161.0/874.0	446.0/589.0	35.0/1000.0	610.0/425.0	231.0/804.0	811.0/224.0	944.5/90.5
LP	w/l					19/26	33/12	5/40	41/4	28/17	43/2	41/4
	p					0.7391	0.0000✓	0.0000✗	0.0000✓	0.0015✓	0.0000✓	0.0000✓
	R^+/R^-					488.0/547.0	887.0/148.0	85.0/950.0	1009.0/26.0	798.0/237.0	1005.0/30.0	1013.0/22.0
MLARAM	w/l						30/15	13/32	37/8	30/15	34/11	38/7
	p						0.0060✓	0.0027✗	0.0000✓	0.0210✓	0.0000✓	0.0000✓
	R^+/R^-						761.0/274.0	252.0/783.0	914.0/121.0	722.0/313.0	921.0/114.0	965.0/70.0
RAkEL	w/l							6/39	25/20	16/29	33/12	33/12
	p							0.0000✗	0.1290	0.0572	0.0000✓	0.0000✓
	R^+/R^-							56.0/979.0	652.0/383.0	349.0/686.0	883.0/152.0	908.0/127.0
BR	w/l								44/1	42/3	44/1	43/2
	p								0.0000✓	0.0000✓	0.0000✓	0.0000✓
	R^+/R^-								1031.0/4.0	986.0/49.0	1016.0/19.0	1015.0/20.0
ML-kNN(PARIS)	w/l									7/36	23/22	25/20
	p									0.0000✗	0.2861	0.1438
	R^+/R^-									121.5/913.5	612.0/423.0	647.0/388.0
ML-kNN(PARIS+LK)	w/l										36/9	39/6
	p										0.0000✓	0.0000✓
	R^+/R^-										918.0/117.0	948.0/87.0

(*continued on next page*)

Table 7 (*continued*).

		ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS)	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
BR(PARIS)	w/l											22/23
	p											0.6314
	R^+/R^-											560.0/475.0
Iman-Davenport $p-$ value: 0.0000												

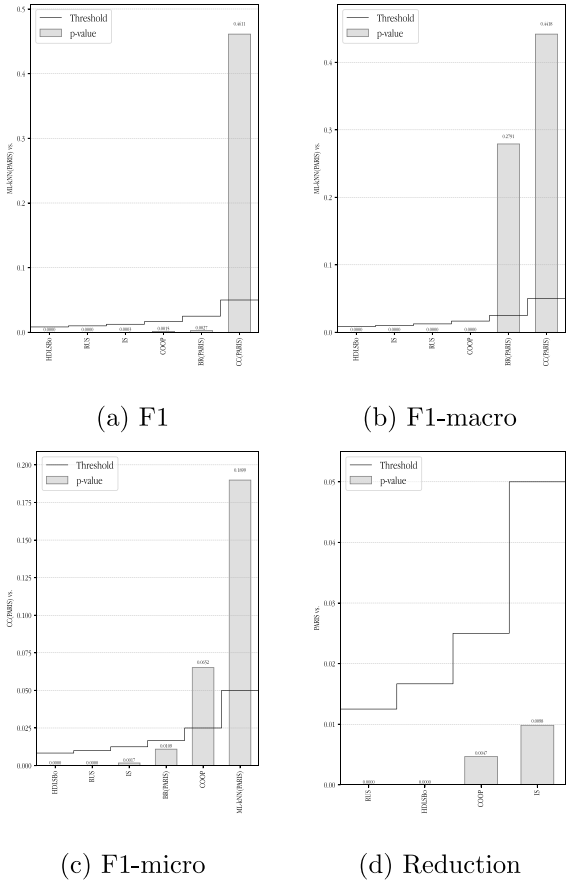


Fig. 13. The Holm test for the comparison of other instance selection algorithms presented in the literature and PARIS for F1, F1-macro, and F1-micro metrics and reduction.

a deeper understanding of when our method improved the standard approach, we studied the differences between PARIS and the standard methods depending on the number of instances of the datasets, the number of inputs, the number of labels, the label cardinality, the label density, the label diversity, the proportion of distinct labels, and the MeanIR and CVIR measures.

For most of these, we found no clear relationship between the characteristics and the differences in the performance of PARIS and its standard counterpart. However, for three cases, some trend was observed, namely, the number of inputs, the cardinality of the labels, and the CVIR measure. Fig. 14 shows the difference as the number of inputs of the datasets increases in terms of F1 metric between ML-kNN and ML-kNN(PARIS), Fig. 14(a), BR and BR(PARIS), Fig. 14(b), and CC and CC(PARIS), Fig. 14(c). A linear regression is also shown to illustrate the general trend in the differences. The figure shows that the relative performance of PARIS improved as the number of features of the datasets increased.

Fig. 15 shows the same information when the datasets were ordered in increasing values of the cardinality of the labels. For CC, no trend was found. However, for ML-kNN and BR our approach tended to perform better for smaller values of cardinality. With smaller values of cardinality the individual labels are more imbalanced. As our method is able to balance the individual labels with its partial selection strategy, it is reasonable to expect a better relative performance for smaller cardinality values. Fig. 16, showing the same information sorted by increasing CVIR values, corroborated these results. The datasets with a higher CVIR are more imbalanced; thus, our method should be able to improve their results.

6. Conclusions and future work

In this paper, we presented a novel approach for carrying out instance selection for multi-label datasets. Instead of selecting or removing an instance from the training set, we proposed the selection of partial instances, meaning that an instance can be used for certain labels only. This approach was adapted to three different multi-label methods: ML-kNN, BR, and CC. This novel approach was compared with the standard method of instance selection in a set of 45 problems and obtained better results. The method was also tested against other well-known multi-label classification models with very competitive performance. We also compared our approach with other state-of-the-art instance selection methods. That comparison showed the superior performance of PARIS.

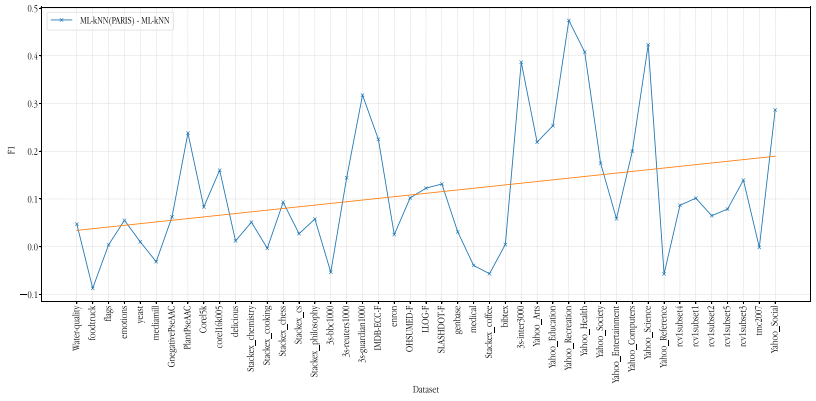
Table 8
Comparison of PARIS and the standard methods for F1-micro metric.

		ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS)	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
Mean		0.3120	0.3243	0.3833	0.3484	0.3469	0.3908	0.2001	0.3846	0.3470	0.3506	0.3994
Ranks		7.3444	6.7444	5.2222	5.9111	6.1111	4.3111	9.4111	4.9667	5.7556	5.9222	4.3000
ML-kNN	w/l		29/15	31/14	27/18	26/19	36/9	7/37	32/13	30/15	29/15	37/8
	p		0.0745	0.0010✓	0.0324✓	0.0650	0.0001✓	0.0000✗	0.0004✓	0.0210✓	0.0294✓	0.0001✓
	R^+/R^-		675.5/359.5	808.0/227.0	707.0/328.0	681.0/354.0	861.0/174.0	53.5/981.5	831.0/204.0	722.0/313.0	710.5/324.5	856.0/179.0
ML-kNN(IS)	w/l			30/15	29/16	26/19	32/13	7/36	30/15	29/16	27/18	32/13
	p			0.0018✓	0.0873	0.2662	0.0002✓	0.0000✗	0.0030✓	0.1290	0.2082	0.0003✓
	R^+/R^-			794.0/241.0	669.0/366.0	616.0/419.0	843.0/192.0	65.5/969.5	780.0/255.0	652.0/383.0	629.0/406.0	840.0/195.0
CC	w/l				15/30	14/31	30/15	9/36	24/21	19/26	23/22	27/18
	p				0.0038✗	0.0451✗	0.2041	0.0000✗	0.7735	0.1207	0.2662	0.0096✓
	R^+/R^-				261.0/774.0	340.0/695.0	630.0/405.0	72.0/963.0	543.0/492.0	380.0/655.0	419.0/616.0	747.0/288.0
LP	w/l					20/25	27/18	6/39	26/19	23/22	24/21	31/14
	p					0.6639	0.0045✓	0.0000✗	0.0334✓	0.9865	0.7908	0.0010✓
	R^+/R^-					479.0/556.0	769.0/266.0	54.0/981.0	706.0/329.0	519.0/516.0	541.0/494.0	809.0/226.0
MLARAM	w/l						26/18	7/38	27/18	22/23	22/22	31/14
	p						0.0320✓	0.0000✗	0.1407	0.8789	0.7264	0.0066✓
	R^+/R^-						707.5/327.5	104.0/931.0	648.0/387.0	531.0/504.0	548.5/486.5	758.0/277.0
RAkEL	w/l							4/41	20/25	19/26	12/32	20/25
	p							0.0000✗	0.6886	0.0700	0.0104✗	0.7477
	R^+/R^-							28.0/1007.0	482.0/553.0	357.0/678.0	290.5/744.5	489.0/546.0

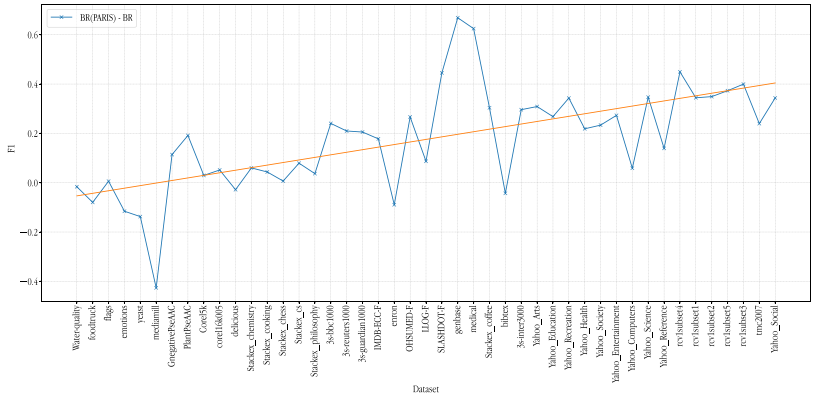
(*continued on next page*)

Table 8 (*continued*).

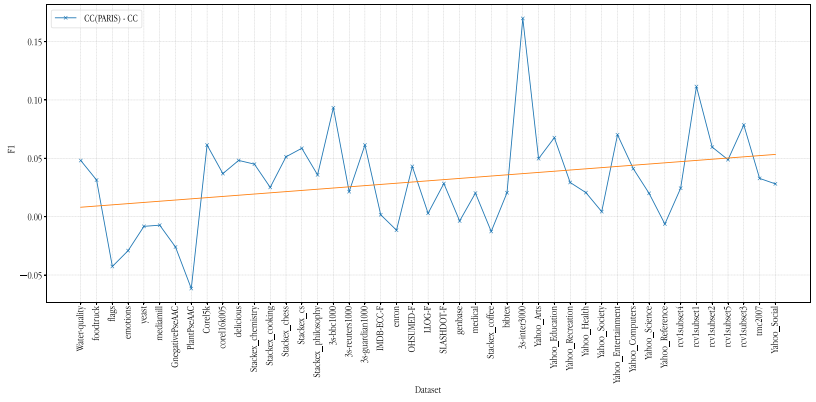
		ML-kNN	ML-kNN(IS)	CC	LP	MLARAM	RAkEL	BR	ML-kNN(PARIS)	ML-kNN(PARIS+LK)	BR(PARIS)	CC(PARIS)
BR	w/l								41/4	36/9	35/10	38/7
	p								0.0000✓	0.0000✓	0.0000✓	0.0000✓
	R^+/R^-								1015.0/20.0	918.0/117.0	887.0/148.0	974.0/61.0
ML-kNN(PARIS)	w/l									17/25	19/26	26/19
	p									0.4004	0.0572	0.3817
	R^+/R^-									443.0/592.0	349.0/686.0	595.0/440.0
ML-kNN(PARIS+LK)	w/l										21/24	29/15
	p										0.9236	0.0167✓
	R^+/R^-										509.0/526.0	729.5/305.5
BR(PARIS)	w/l											30/15
	p											0.0073✓
	R^+/R^-											755.0/280.0
Iman-Davenport $p-$ value: 0.0000												



(a) ML-kNN

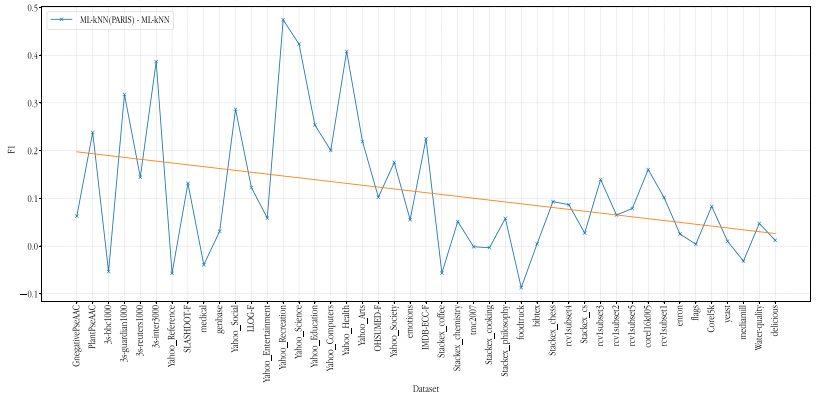


(b) BR

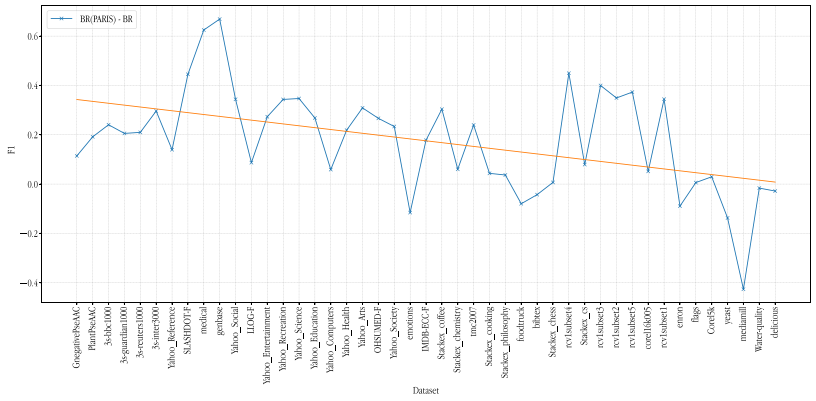


(c) CC

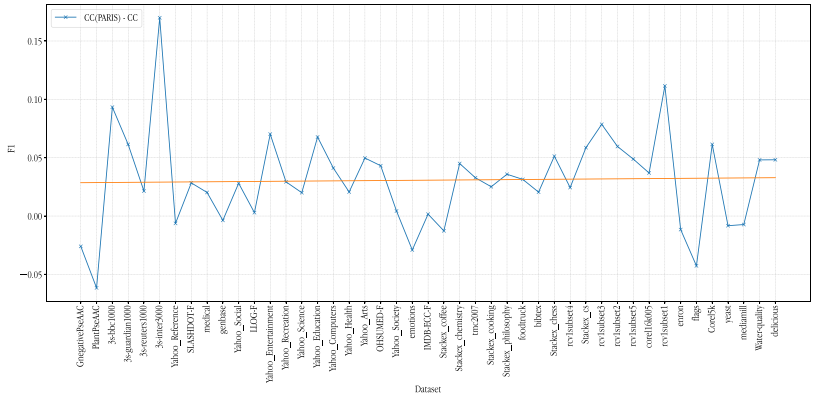
Fig. 14. Difference in terms of F1 metric between PARIS and the standard methods sorted by an increasing number of features.



(a) ML-kNN



(b) BR



(c) CC

Fig. 15. Difference in terms of F1 metric between PARIS and the standard methods sorted by increasing cardinality.

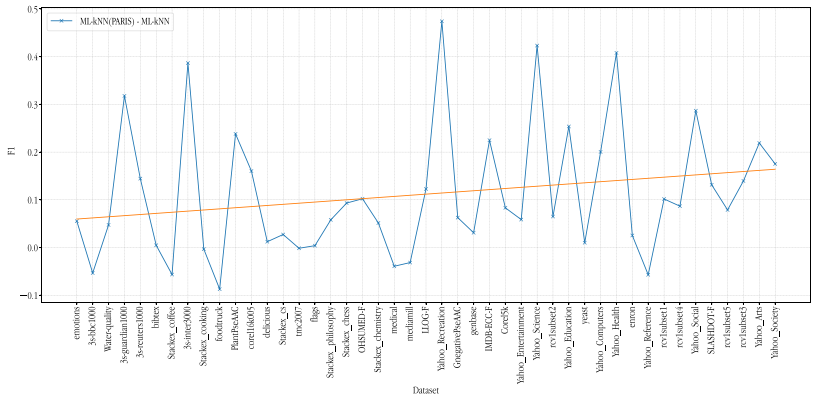
Our proposal opens new research lines. One of the first is the possibility of applying a cooperative coevolutionary approach to our method. The decomposition of the evolution for each label is a natural extension of the presented algorithm. Another interesting research line is extending the presented philosophy to other classification methods. For some multi-label methods, such as the classifier label ranking (CLR) [42] or instance-based logistic regression for multi-label data (IBLR-ML) [6], the adaptation is straightforward. Extensions to other methods used in this paper, such as the LP, RAkEL, or MLARAN, would be an interesting challenge.

Declaration of competing interest

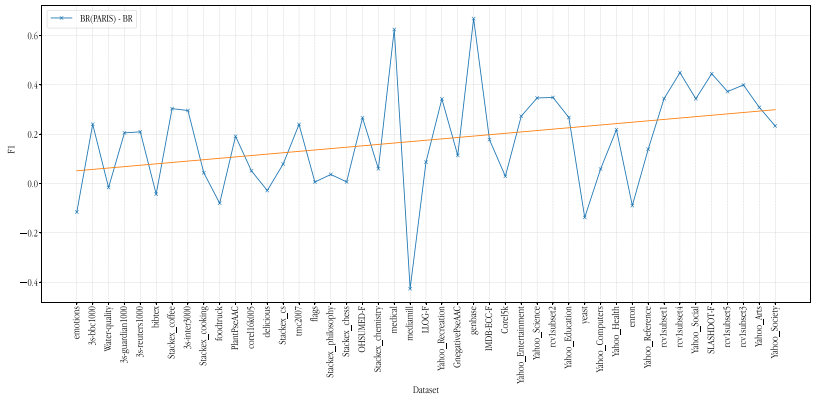
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

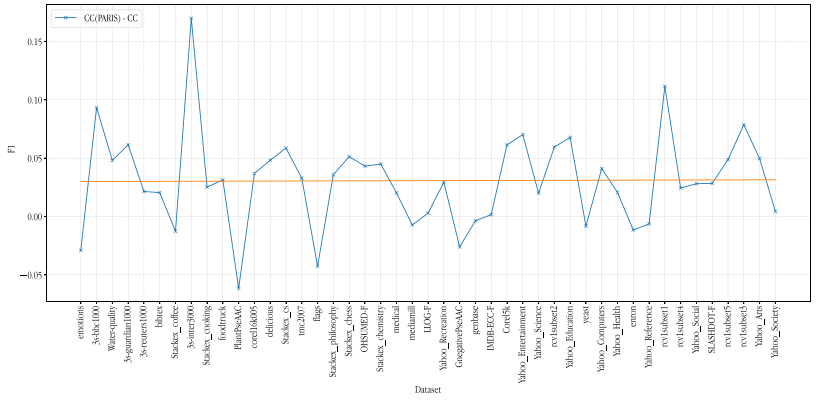
Data will be made available on request.



(a) ML-kNN



(b) BR



(c) CC

Fig. 16. Difference in terms of F1 metric between PARIS and the standard methods sorted by increasing cardinality.

References

[1] M.L. Zhang, Z.H. Zhou, A review on multi-label learning algorithms, IEEE Trans. Knowl. Data Eng. 26 (2014) 1819–1837.

[2] J. Read, L. Martino, P.M. Olmos, D. Luengo, Scalable multi-output label prediction: From classifier chains to classifier trellises, Pattern Recognit. 48 (2015) 2096–2109.

[3] M.-L. Zhang, Z.-H. Zhou, Ml-knn: A lazy learning approach to multi-label learning, Pattern Recognit. 40 (2007) 2038–2048.

[4] Z. Younes, F. Abdallah, T. Denoeux, Multi-label classification algorithm derived from k-nearest neighbor rule with label dependencies, in: 16th European Signal Processing Conference, 2008, pp. 1–5.

[5] A. Pakrashi, B.M. Namee, Stacked-mlknn: A stacking based improvement to multi-label k-nearest neighbours, Proc. Mach. Learn. Res. 74 (2017) 51–63.

[6] W. Cheng, E. Hüllermeier, Combining instance-based learning and logistic regression for multilabel classification, Mach. Learn. 76 (2009) 211–225.

[7] X. Lin, X.-W. Chen, Mr.knn: Soft relevance for multi-label classification, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, Association for Computing Machinery, New York, NY, USA, 2010, pp. 349–358.

[8] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Min. Knowl. Discov. 6 (2002) 153–172.

[9] N. García-Pedrajas, A. de Haro-García, J. Pérez-Rodríguez, A scalable memetic algorithm for simultaneous instance and feature selection, Evolut. Comput. 22 (2014) 1–45.

[10] N. García-Pedrajas, Evolutionary computation for training set selection, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 1 (6) (2011) 512–523.

[11] N. García-Pedrajas, G. Cerruela-García, Cooperative coevolutionary instance selection for multilabel problems, Knowl.-Based Syst. 234 (2021) 10756.

[12] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[13] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, Addressing imbalance in multilabel classification: Measures and random resampling algorithms, Neurocomputing 163 (2015) 3–16.
[14] A. Arnaiz-González, J.F. Díez-Pastor, J.J. Rodríguez, C. García-Osorio, Local sets for multi-label instance selection, Appl. Soft Comput. 68 (2018) 651–666.
[15] M. Kordos, A. Arnaiz-González, C. García-Osorio, Evolutionary prototype selection for multi-output regression, Neurocomputing 358 (2019) 309–320.
[16] J.R. del Castillo, D. Ortiz-Boyer, N. García-Pedrajas, Instance selection for multi-label learning based on a scalable evolutionary algorithm, in: 2021 International Conference on Data Mining Workshops, ICDMW, IEEE Computer Society, Los Alamitos, CA, USA, 2021, pp. 843–851, http://dx.doi.org/10.1109/ICDMW53433.2021.00108.
[17] J. Ma, T.W.S. Chow, Topic-based instance and feature selection in multilabel classification, IEEE Trans. Neural Netw. Learn. Syst. 33 (2022) 315–329.
[18] D.E.K. Mansouri, K. Benabdeslem, Towards multi-label feature selection by instance and label selections, in: K. Karlapalem, H. Cheng, N. Ramakrishnan, R.K. Agrawal, P.K. Reddy, J. Srivastava, T. Chakraborty (Eds.), Advances in Knowledge Discovery and Data Mining, Springer International Publishing, 2021, pp. 233–244.
[19] J.J. Valero-Mas, A.J. Gallego, P. Alonso-Jiménez, X. Serra, Multilabel prototype generation for data reduction in k-nearest neighbour classification, Pattern Recognit. 135 (2023) 109190.
[20] J. Pérez-Rodríguez, A. Arroyo-Peña, N. García-Pedrajas, Simultaneous instance and feature selection and weighting using evolutionary computation: Proposal and study, Appl. Soft Comput. 37 (2015) 416–443.
[21] L.J. Eshelman, The CHC Adaptive Search Algorithm: How to Have Safe Search when Engaging in Nontraditional Genetic Recombination, Morgan Kauffman, San Mateo, CA, 1990.
[22] S.J. Louis, G. Li, Combining robot control strategies using genetic algorithms with memory, in: Lecture Notes in Computer Science, Evolutionary Programming VI, vol.1213, 1997, pp. 431–442.
[23] C. García-Osorio, A. de Haro-García, N. García-Pedrajas, Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts, Artificial Intelligence 174 (2010) 410–441.
[24] A. de Haro-García, N. García-Pedrajas, A divide-and-conquer recursive approach for scaling up instance selection algorithms, Data Min. Knowl. Discov. 18 (3) (2009) 392–418.
[25] N. García-Pedrajas, A. de Haro-García, J. Pérez-Rodríguez, A scalable approach to simultaneous evolutionary instance and feature selection, Inform. Sci. 228 (2013) 150–174.
[26] K. Sechidis, G. Tsoumakas, I. Vlahavas, On the stratification of multi-label data, in: Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases, Part III, 2011, pp. 145–158.
[27] P. Szymański, T. Kajdanowicz, A network perspective on stratification of multi-label data, in: First International Workshop on Learning with Imbalanced Domains: Theory and Applications, 2017, pp. 22–35.
[28] A. de Haro-García, J. Pérez-Rodríguez, N. García-Pedrajas, Combining three strategies for evolutionary instance selection for instance-based learning, Swarm Evol. Comput. 42 (2018) 160–172.
[29] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[30] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–83.
[31] P.B. Nemenyi, Distribution-Free Multiple Comparisons, (Ph.D. thesis), Princeton University, 1963.
[32] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (2004) 1757–1771.
[33] M.S. Sorower, A literature survey on algorithms for multi-label learning, in: Ph. D Qualifying Review Paper. Major Professor: Thomas G. Dietterich, Ph.D, Computer Science, Oregon State University, 2010.
[34] R.E. Schapire, Y. Singer, Boostexter: A boosting-based system for text categorization, Mach. Learn. 39 (2000) 135–168.
[35] S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, in: Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2004, in: 3056 of Lecture Notes in Computer Science, Springer, 2004, pp. 22–30.
[36] G. Tsoumakas, I. Katakis, I. Vlahavas, Random k-labelsets for multi-label classification, IEEE Trans. Knowl. Data Eng. 23 (2011) 1079–1089.
[37] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. 85 (2011) 333–359.
[38] F. Benites, E. Sapozhnikova, Haram: A hierarchical aram neural network for large-scale text classification, in: 2015 IEEE International Conference on Data Mining Workshop, ICDMW, 2015, pp. 847–854, http://dx.doi.org/10.1109/ICDMW.2015.14.
[39] P. Szymański, T. Kajdanowicz, A scikit-based Python environment for performing multi-label classification, 2017, ArXiv e-prints arXiv:1702.01460.
[40] S. Kanj, F. Abdallah, T. Denoeux, K. Tout, Editing training data for multi-label classification with the k-nearest neighbor rule, Pattern Anal. Appl. 19 (1) (2016) 145–161.
[41] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, Mlenn: A first approach to heuristic multilabel undersampling, in: 15th International Conference Intelligent Data Engineering and Automated Learning, Springer International Publishing, Salamanca (Spain), 2014, pp. 1–9.
[42] J. Fürnkranz, E. Hüllermeier, E.L. Mencía, Multilabel classification via calibrated label ranking, Mach. Learn. 73 (2008) 133–153.