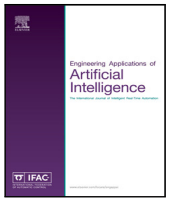
Engineering Applications of Artificial Intelligence 116 (2022) 105487

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Local-based k values for multi-label k-nearest neighbors rule ☆

J.A. Romero-del-Castillo *, Manuel Mendoza-Hurtado, Domingo Ortiz-Boyer, Nicolás García-Pedrajas

Department of Computing and Numerical Analysis, University of Córdoba, Campus de Rabanales, 14011 Córdoba, Spain

ARTICLE INFO

Keywords:
Multi-label k-nearest neighbors
Locally optimized k neighbors

ABSTRACT

Multi-label learning is a growing field in machine learning research. Many applications address instances that simultaneously belong to many categories, which cannot be disregarded if optimal results are desired. Among the many algorithms developed for multi-label learning, the multi-label k-nearest neighbor method is among the most successful. However, in a difficult classification task, such as multi-label learning, a challenge that arises in the k-nearest neighbor approach is the assignment of the appropriate value of k. Although a suitable value might be obtained using cross-validation, it is unlikely that the same value will be optimal for the whole space spanned by the training set. It is evident that different regions of the feature space would have different distributions of instances and labels that would require different values of k. The very complex boundaries among the many present labels make the necessity of local k values even more important than in the case with a single-label k-nearest neighbor. We present a simple yet powerful approach for setting a local value of k. We associate a potentially different k with every prototype and obtain the best value of that k by optimizing the criterion consisting of the local effect of the different k values in the neighborhood of the prototype. The proposed method has a fast training stage, as it only uses the neighborhood of each training instance to set the local k value. The complexity of the proposed method in terms of the testing time is similar to that of the standard multi-label k-nearest neighbor approach. Experiments performed on a set of 20 problems show that not only does our proposed method significantly outperform the standard multi-label k-nearest neighbor rule but also the locally adaptive multi-label k-nearest neighbor method can benefit from a local k value.

1. Introduction

Many modern applications use increasingly complex categorization schemes for data classification, where one instance may simultaneously belong to several topics. This task is typically termed multi-label learning (Zhang and Zhou, 2014).

In contrast with single-label classification where one instance is associated with only one class, multi-label classification is concerned with learning from instances that can be associated with multiple labels. Multi-label problems are complex, and therefore, are more difficult to solve than their single-label counterparts.

We cannot overstate the importance of multi-label learning, which has been developed and applied in several highly active research areas: text categorization (Schapire and Singer, 2000; Chen et al., 2007; Zhang et al., 2020a), automatic image annotation (Yu et al., 2013; Zhang et al., 2012; Wang and Zhang, 2020; Zhang et al., 2020b), web mining (Tang et al., 2009; Read et al., 2012), rule mining (Thabtah et al., 2004; Veloso et al., 2007), cheminformatics (Afzal et al., 2015; Toledano et al., 2019), data stream (Zheng et al., 2020; Alberghini et al., 2022), bioinformatics (Clare and King, 2001; Elisseeff and Weston, 2001) or information retrieval (Lai et al., 2016). In some of these fields of application, such as image classification, Deep learning (Sumbul and Demir, 0000) has proven its good performance.

Our method is aimed at tabular datasets, where classic methods such as ML-kNN have proven their good performance in spite of their simplicity. Tabular datasets as the ones used in this paper are common in many real-world applications of multi-label learning. For instance, in Cheminformatics molecular activity is usually predicted using either molecular fingerprints or descriptors. Both of them represent that dataset in a tabular way. These representations have been used for estimating molecular activities in a multi-label approach (Toledano et al., 2019). Web page categorization using a Bag-of-Words (BOW) for representing each document is also a common scenario for multi-label classification tasks (Ueda and Saito, 2003). This BOW representation is also a widely used method for describing documents in sentiment and mood analysis from a multi-label perspective (Liu and Chen, 2015).

☆ This work was supported in part by Project TIN2015-66108-P of the Spanish Ministry of Science and Innovation.

* Corresponding author.
E-mail addresses: aromero@uco.es (J.A. Romero-del-Castillo), i52mehum@uco.es (M. Mendoza-Hurtado), dortiz@uco.es (D. Ortiz-Boyer), npedrajas@uco.es (N. García-Pedrajas).

https://doi.org/10.1016/j.engappai.2022.105487
Received 15 June 2022; Received in revised form 6 September 2022; Accepted 26 September 2022
Available online 11 October 2022

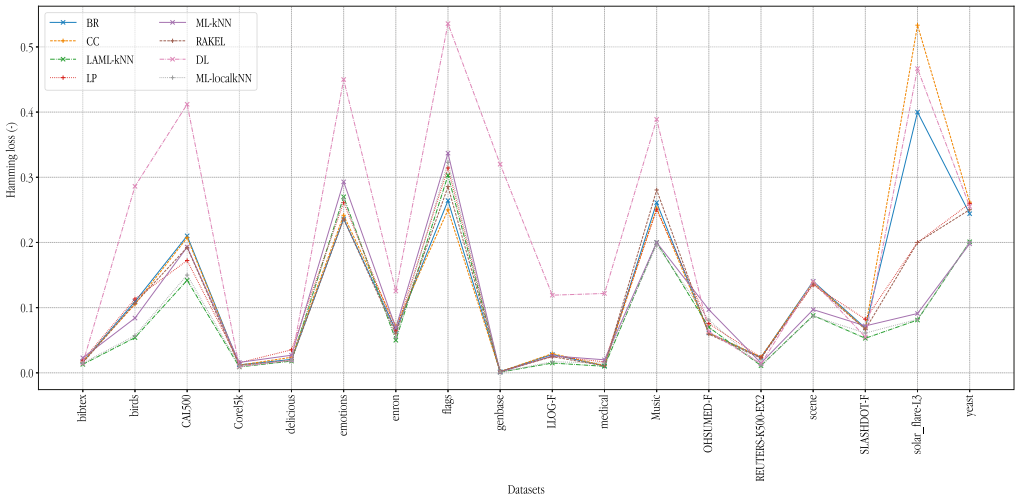


Fig. 1. *Hamming Loss* metric results for ML-kNN, LAML-kNN, Binary relevance, Classifier chains, Label powerset, RAkEL, Deep learning and our proposal.

Table 1
Description of the datasets.

Name	Instances	Features	Labels	LCard	LDens
bibtex	7395	1836	159	2.402	0.015
birds	645	260	19	1.014	0.053
CAL500	502	68	174	26.044	0.150
Corel5k	5000	499	374	3.522	0.009
delicious	16105	500	983	19.020	0.019
emotions	593	72	6	1.868	0.311
enron	1702	1001	53	3.378	0.064
flags	194	19	7	3.392	0.485
genbase	662	1185	27	1.252	0.046
LLOG-F	1460	1004	75	1.180	0.016
medical	978	1449	45	1.245	0.028
Music	592	71	6	1.870	0.312
OHSUMED-F	13929	1002	23	1.663	0.072
REUTERS	6000	500	103	1.462	0.014
scene	2407	294	6	1.074	0.179
SLASHDOT-F	3782	1079	22	1.181	0.054
solar_flare-L3	323	10	3	0.232	0.077
yeast	2417	103	14	4.237	0.303

BOW representation is also utilized by some of the datasets used in this paper (see Table 1). Datasets in Bioinformatics are usually represented as protein sequences (such as genbase dataset used in our experiments) or microarray information (yeast dataset). Both representations form tabular datasets.

The key challenge of multi-label learning is to take advantage of the correlations among labels to address the exponential growth of the label space with the number of distinct labels, 2^q, for q different labels. Methods that address multi-label datasets without considering the relationships among labels simply solve a group of independent binary problems. Learning and using the complex relationships among the labels is important for the success of any multi-label classification model.

Two broad categories of methods are employed to solve multi-label problems (Zhang and Zhou, 2014): problem transformation methods and algorithm adaptation methods. Problem transformation methods solve the multi-label learning problem by transforming it into other established learning scenarios. Algorithm adaptation methods work by modifying the known learning techniques to directly analyze multi-label data. The success of any approach depends on taking advantage of the correlations among labels to develop novel methods. These relationships can be complex and hierarchically structured. The performance of any method is closely related to its ability to address higher-order relationships among labels.

Among the adapted algorithms for multi-label learning, the multi-label k-nearest neighbor (ML-kNN) method (Zhang and Zhou, 2007) is among the top performing methods. ML-kNN is an adaptation of the well-known k-nearest neighbor method for multiclass problems. ML-kNN consists of storing a set of prototypes that represent the knowledge of the problem and using those prototypes to predict the *a posteriori* probability of each label to belong to the relevant set of labels of a given query instance.

ML-kNN is a fast method that matches the performance of more complex methods (Aldrees et al., 2016). Zufferey et al. (2015) carried out a comparison of many different multi-label methods in an experiment that used clinical data from chronic diseases and discovered that ML-kNN outperformed even ensemble methods such as AdaBoost.MH (Schapire and Singer, 2000), RAkEL (Tsoumakas et al., 2011a), HOMER (Tsoumakas et al., 2008) and Classifier chains (CC) (Read et al., 2011) for several multi-label metrics. In an extensive comparison, Madjarov et al. (2012) revealed that the performance of ML-kNN was similar to other more complex methods for most of the metrics, while being significantly faster.

Thus, ML-kNN, although simple, can usually match and even outperform more sophisticated and complex methods in terms of generalization error. A disadvantage of this classifier, however, is encountered in assigning an appropriate value of k. Although a good value might be obtained using cross-validation, the same value is unlikely to be optimal for the whole space spanned by the training set.

In this work, we propose a method that obtains local values from the training set for the k parameter, which will be utilized during the testing time. Our approach is based on directly learning the local value of k from the training set, evaluating the effect of every k value and choosing the best performing value. The proposed method is fast and accurate, showing better generalization capabilities than the original method with the same testing complexity.

The selection of the optimal value for k in a certain dataset is always problematic, as only a finite amount of training data are available. The standard approach assumes that there exists a unique k value that is optimal for all the regions of the input space. In practice, the situations of different prototypes differ. The appropriate values of k are very different for a prototype surrounded by other instances with similar labels, near the boundaries of the labels or with many noisy instances in its neighborhood. Thus, the use of different k values for the different prototypes would likely have a positive effect on the classification accuracy of the k-nearest neighbor rule.

We formally define a multi-label problem as follows: Let T be a multi-label dataset consisting of p multi-label instances $\mathbf{x}_i$ and its

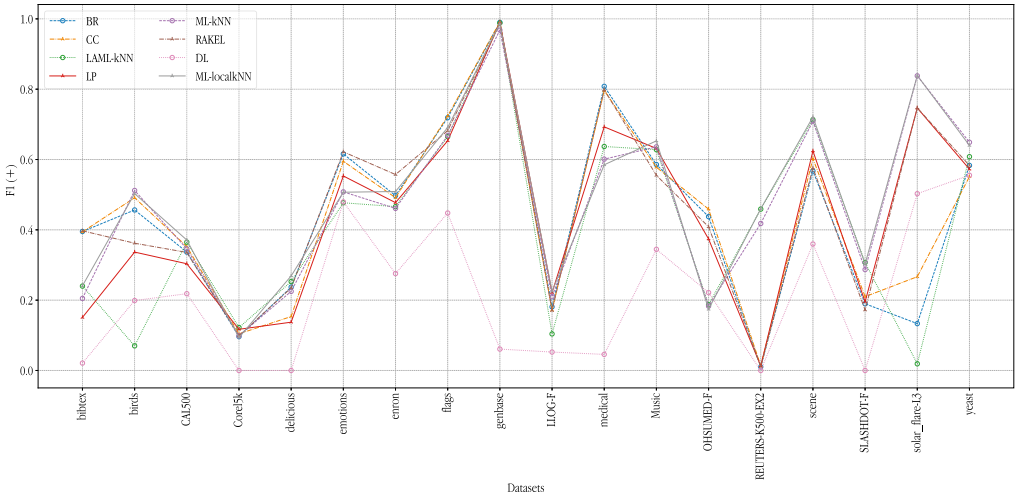


Fig. 2. *F1* metric results for ML-kNN, LAML-kNN, Binary relevance, Classifier chains, Label powerset, RAkEL, Deep learning and our proposal.

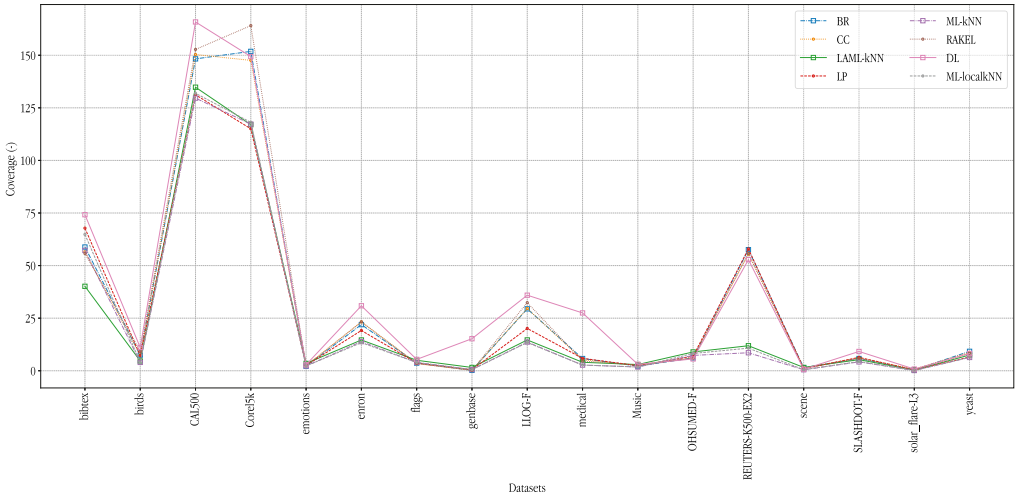


Fig. 3. *Coverage* metric results for ML-kNN, LAML-kNN, Binary relevance, Classifier chains, Label powerset, RAkEL, Deep learning and our proposal. Delicious dataset is not shown for a better presentation.

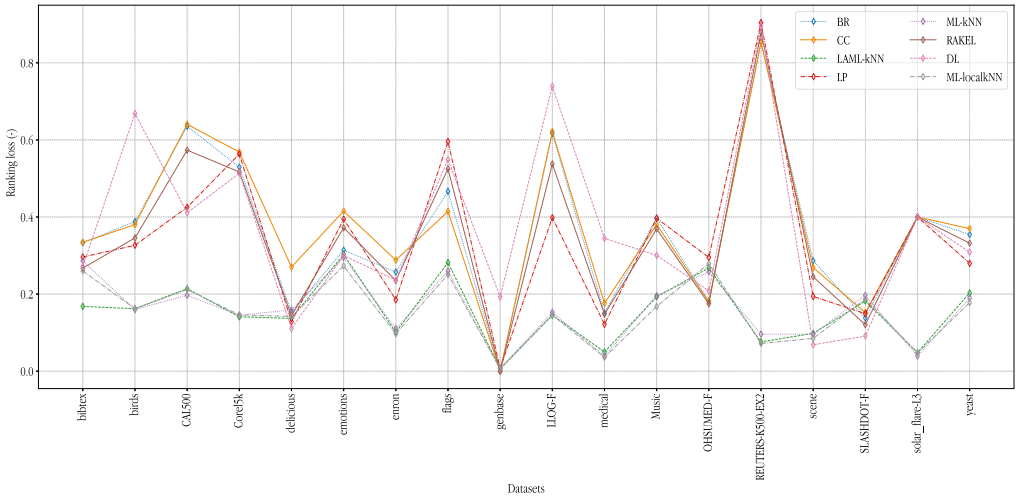


Fig. 4. *Ranking Loss* metric results for ML-kNN, LAML-kNN, Binary relevance, Classifier chains, Label powerset, RAkEL, Deep learning and our proposal.

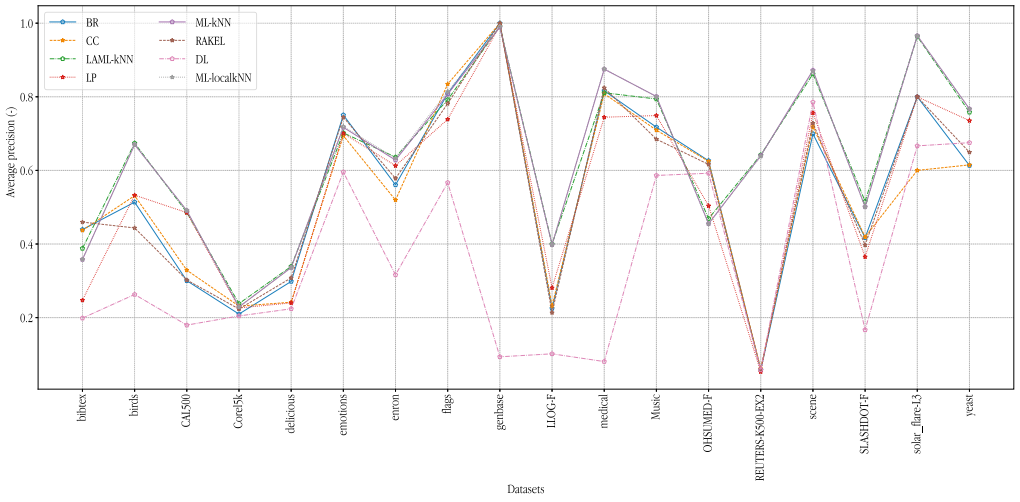


Fig. 5. *Average Precision* metric results for ML-kNN, LAML-kNN, Binary relevance, Classifier chains, Label powerset, RAkEL, Deep learning and our proposal.

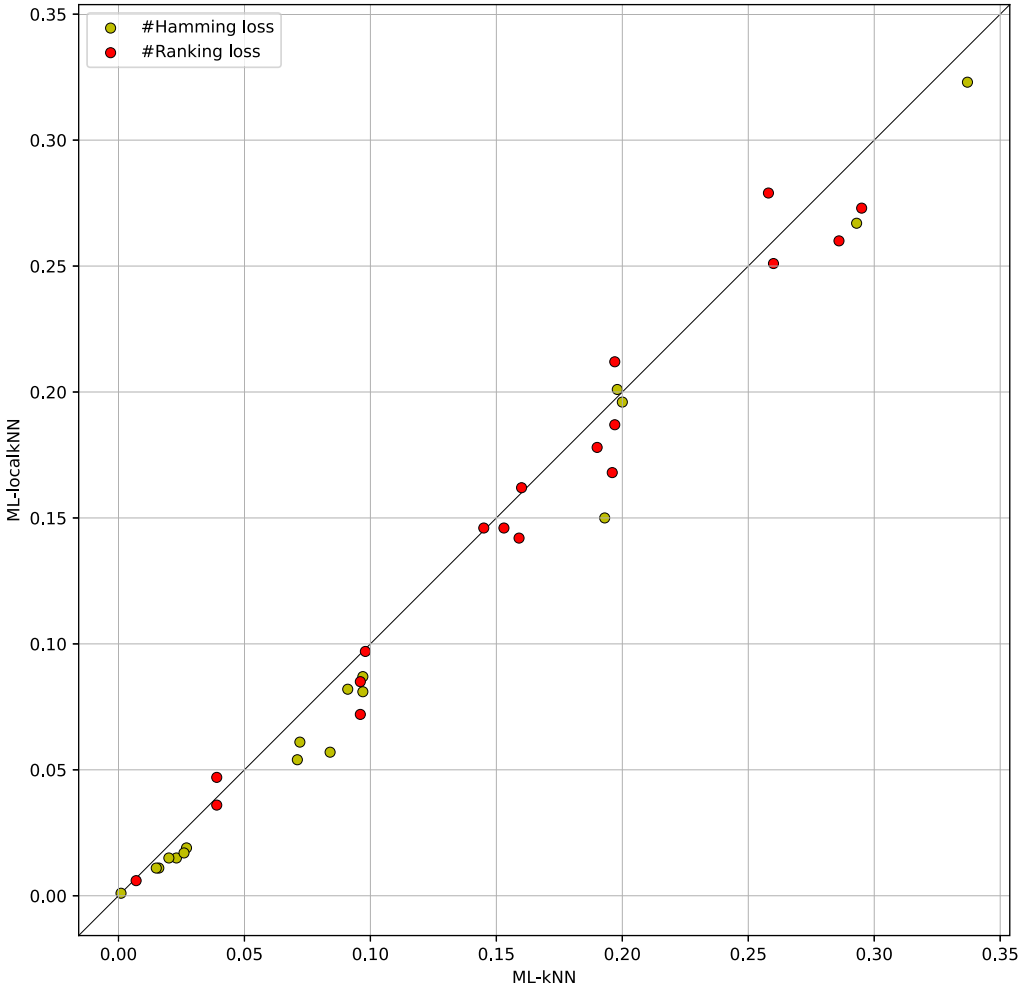


Fig. 6. Comparison of ML-localkNN vs ML-kNN using *Hamming Loss* and *Ranking Loss* metrics.

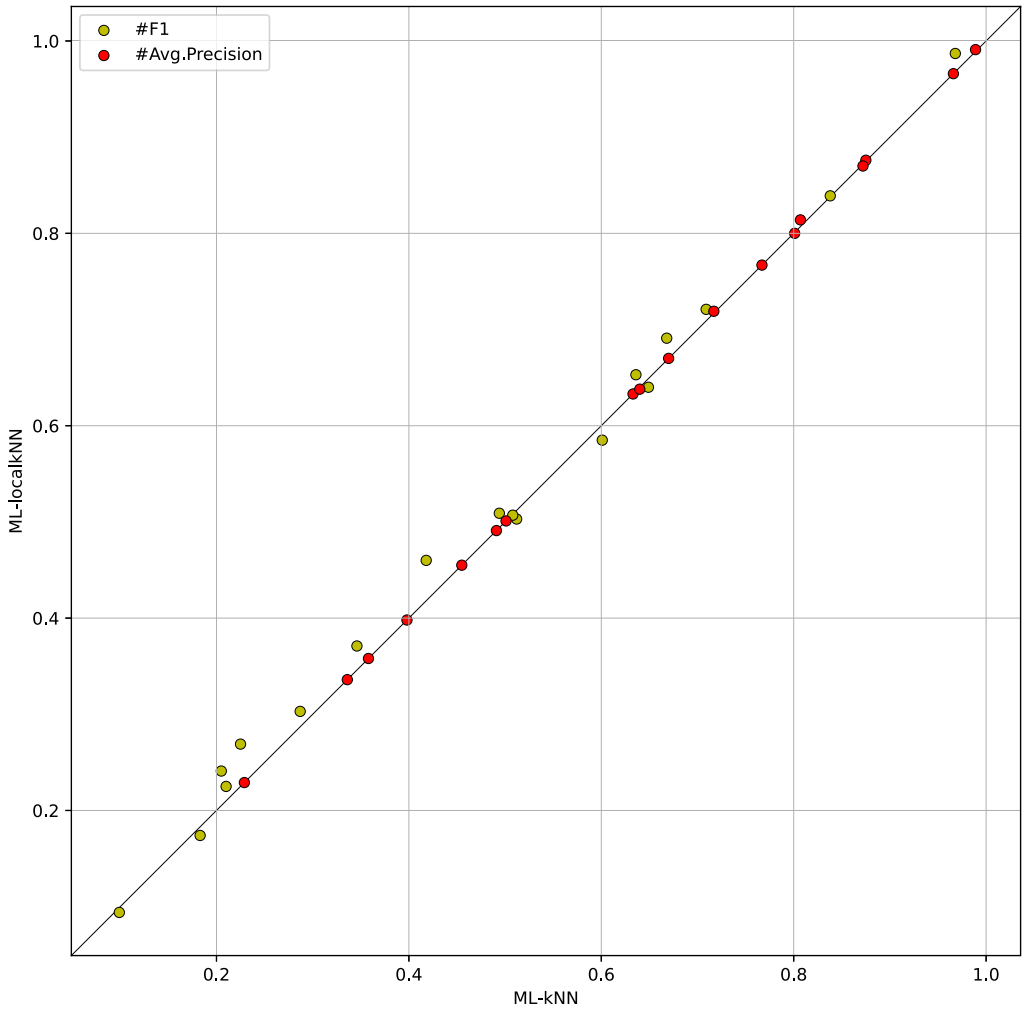


Fig. 7. Comparison of ML-LOCALkNN vs ML-kNN using F_1 and *Average Precision* metrics.

associated labelset Y_i, i.e., $T = \{(\mathbf{x}_i, Y_i)\}$, $1 \leq i \leq p$, $\mathbf{x}_i \in X$, $Y_i \in Y$, $X = \mathbb{R}^p$ and $Y = \{0|1\}^q$ is the label space with q possible binary labels. Let h be a multi-label classifier and $h(\mathbf{x}_i) \in \{0|1\}^q$ be the set of labels predicted by h for the instance $\mathbf{x}_i$. Let $f(\mathbf{x}_i, y)$, $\mathbf{x}_i \in X$, $y \in Y_i$, $Y_i \in Y$ be a real-valued function $f : X \times Y \to \mathbb{R}$. A successful learning system would tend to output larger values of f for labels in Y_i than for the labels that are not in Y_i. The real-valued function f can be easily transformed into a ranking function $\text{rank}_f(\mathbf{x}_i, y)$ to predict the rank of the label y for instance $\mathbf{x}_i$. The classifier $h(\mathbf{x}_i)$ can be obtained from $f(\mathbf{x}_i, y)$ when an appropriate threshold is set. Our approach is based on two main ideas. First, we assign to each prototype a value of k that is used to classify any query instance whose nearest neighbor is that prototype. Thus, instead of the standard set formed by prototypes of the form $(\mathbf{x}_i, Y_i)$, where $\mathbf{x}_i$ is the prototype and Y_i is its set of relevant labels, we use augmented prototypes of the form $(\mathbf{x}_i, Y_i, k_i)$, where k_i is the associated k value for prototype $\mathbf{x}_i$. To classify any query instance $\mathbf{x}$, we obtain its nearest neighbor $\mathbf{x}_{\text{nearest}}$ and then we proceed with standard ML-kNN using $k = k_{\text{nearest}}$. In our training process, we must obtain the optimal value of k_i values associated with every prototype $\mathbf{x}_i$. This optimal value is obtained by evaluating all the possible k values in a given interval with an evaluation function that considers the classification performance of each value.

The main contribution of this paper is that we present the first method for developing a k nearest neighbor based learning algorithm for multi-label datasets which learns local k values for every prototype. Our approach has the advantage of allowing the selection this local k value with a very simple and fast procedure. During the training time, the method is simple and has linear complexity; during the testing time, the algorithm has the same workload as the standard version of the k-nearest neighbor rule. Furthermore, the whole process can be run in parallel, which means that the size of the dataset is not a constraint in our approach, as the process of obtaining the optimal k value for each prototype is independent of the remaining prototypes.

The experiments show that our proposal can significantly outperform the standard ML-kNN rule in a set of 20 problems, with different numbers of labels, features and instances, and it is evaluated using five different metrics.

To validate the proposed approach, we also compare the results with another method that is also an adaptation of ML-kNN but that takes into account the local difference among samples. The locally adaptive multi-label k-nearest neighbor (LAML-kNN) method (Wang et al., 2018a) considers local differences in the number of neighbors of each instance having each label. The method adapts ML-kNN by calculating different probabilities to be assigned to each label depending on the region in which the test instance is located. For this reason, we have also considered it interesting to validate this model against the model that we propose.

This paper is organized as follows: Section 2 revises the previous work on the topic of the paper: Section 3 explains our proposal; Section 4 describes our experimental setup; Section 5 shows the results

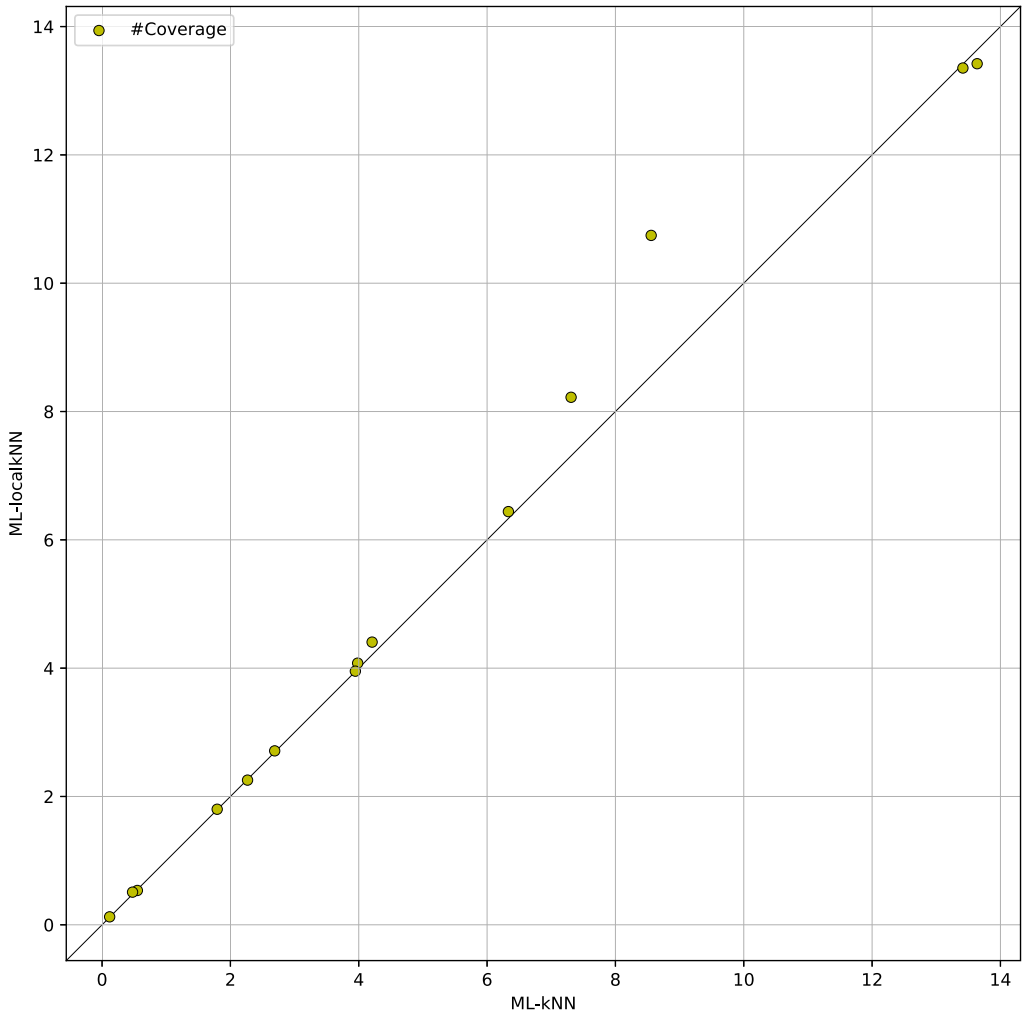


Fig. 8. Comparison of ML-LOCALkNN vs ML-kNN using *Coverage* metric.

of our experiments; and Section 6 states the conclusions of our work and describes new research lines.

2. Related work

Although local k values have been developed for single-label k-NN methods (García-Pedrajas et al., 2017), to the best of our knowledge, no other attempt has been made to develop local k values for multi-label instance-based methods. The method developed for standard single label is simpler, as the number of neighbors only affects the vicinity considered in testing time when a query instance is classified. For ML-kNN the situation is more complex as the number of neighbors affects the calculation of the prior and posterior probabilities and the final probabilities of every label in testing time. Furthermore, we must take into account how the number of local neighbors affect the ability to predict each label.

There are related attempts to try to improve the number of considered neighbors but none of them have addressed the problem of a local value. Wang et al. (2018b) used local information for improving the standard ML-kNN algorithm but using a unique global k for all instances and labels. Roseberry et al. (2021) developed an adaptive value of k for data streams that evolves as new data is obtained. The method was applied in a subsequent paper for the construction of ensembles of classifiers (Alberghini et al., 2022). In the context of instance selection for ML-kNN, García-Pedrajas and Cerruela-García (2021) developed a cooperative coevolutionary algorithm where the final value of k was evolved alongside the selection of instances.

3. Multi-label local k-nearest neighbors method

The basic idea of the proposed multi-label local k-nearest neighbors (ML-LOCALkNN) method is to use a local value of k that is adapted to every region spanned by the training set. To achieve this goal every prototype $\mathbf{x}_i$ of the training set is assigned a local k value k_i. The testing stage of ML-kNN is modified to use these k's. When a new query or test instance $\mathbf{x}_t$ must be classified, first, the nearest neighbor of $\mathbf{x}_t$ in the training set $\mathbf{x}_i$ is obtained, and second, the k value of $\mathbf{x}_i$ is used to classify query instance $\mathbf{x}_t$ as shown in the standard ML-kNN algorithm.

Instead of the standard set formed by prototypes of the form $(\mathbf{x}_i, Y_i)$, where $\mathbf{x}_i$ is the prototype and Y_i is its set of relevant labels, we use augmented prototypes of the form $(\mathbf{x}_i, Y_i, k_i)$, where k_i is the associated k value for prototype $\mathbf{x}_i$ that will be utilized in its neighborhood as the value for the k-nearest neighbor rule.

Once we have established this new modified version of ML-kNN, the optimal local value, k_i, needs to be obtained for every training instance $\mathbf{x}_i$. As our aim is to improve the classification ability of ML-kNN, we

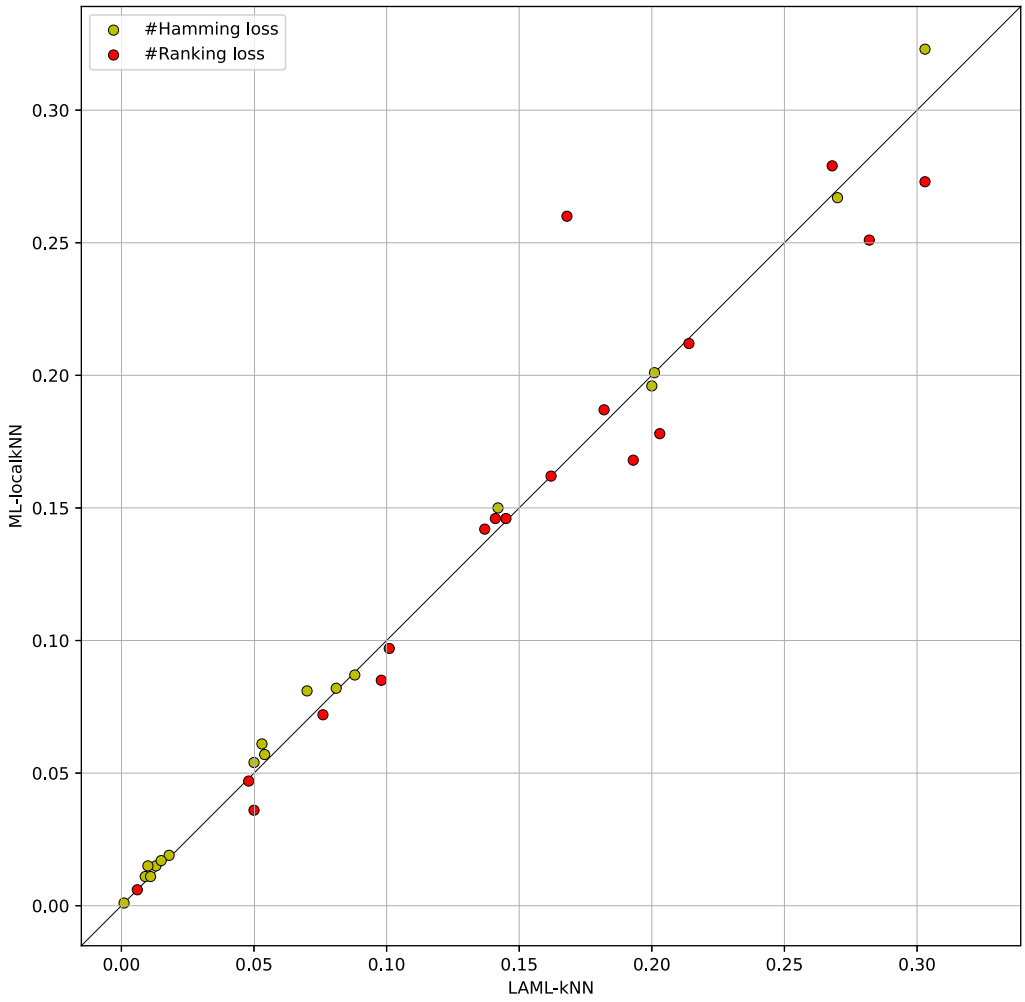


Fig. 9. Comparing ML-LOCALkNN vs LAML-kNN using *Hamming Loss* and *Ranking Loss* metrics.

must obtain the set of k values, $(k_1, k_2, \ldots, k_p)$, that would obtain an optimal classification performance. In this way, it is an optimization problem, were we must obtain the value for every $k_i \in [k_{\min}, k_{\max}]$ that obtains the best classification performance.

Although the optimization problem might appear very difficult, there is a fact that makes it easier. Every local k_i only affects the instances whose nearest neighbor is $\mathbf{x}_i$; thus, to obtain the optimal k_i value, our algorithm only needs to take into account the instances in the vicinity of $\mathbf{x}_i$. This fact simplifies the problem and accelerates the algorithm. The second relevant aspect is the evaluation of the set of possible candidate values of k for every training instance. Thus, we can use the straightforward approach of assigning to $\mathbf{x}_i$ the k_i value that optimizes the classification performance of the set of instances, $N(\mathbf{x}_i)$, which has $\mathbf{x}_i$ as its nearest neighbor. The aforementioned optimization for calculating k_i can be performed using any chosen metric or a combination of different metrics.

However, for certain $\mathbf{x}_i$, the set $N(\mathbf{x}_i)$ might be too small or even empty. Thus, instead of using only the nearest neighbor, we can increase the number of instances in $N(\mathbf{x}_i)$ using the instances that have $\mathbf{x}_i$ as the second nearest neighbor or the instances that have $\mathbf{x}_i$ as the third nearest neighbor, etc.. Thus, $N(\mathbf{x}_i)$ will be composed of the training instances $\mathbf{x}_j$ that have $\mathbf{x}_i$ among their $k_{\text{neighbors}}$ nearest neighbors, that is:

$$N(\mathbf{x}_i) = \{x_j | nearest(\mathbf{x}_j, k_{\text{neighbors}}) = \mathbf{x}_i\} \quad (1)$$

The pseudo-code for the ML-LOCALkNN is given in Algorithm 1. In the training phase, we obtain the set $N(\mathbf{x}_i)$ for each training prototype $\mathbf{x}_i$. ML-kNN is then independently applied to each $N(\mathbf{x}_i)$ using different values of k between $[k_{min}, k_{max}]$. The best value of k is selected and assigned as the optimal local k_i for instance $\mathbf{x}_i$. An optimal k_i can be chosen by considering either one metric or a combination of the metrics explained in Section 4.1.

The test phase is described in Algorithm . A further modification is made to ML-LOCALkNN to improve its results. For certain instances due to the small number of prototypes considered to obtain the local value of k, very different values can be obtained for neighboring prototypes. To obtain a smoother k value, we consider a second hyper-parameter, n_{mean}. For a query instance q, instead of using just the k_t given by its nearest neighbor $\mathbf{x}_t$, we use the average k value given by its n_{mean} nearest neighbors ($n_{\text{mean}} \geq 1$). In our experiments, we consider values $n_{\text{mean}} \in [1, 3]$.

Our ML-LOCALkNN method is compared with the classic method, which uses a fixed value for k in every dataset. For every query instance $\mathbf{x}_t$, the standard ML-kNN uses the maximum a posterior probability (MAP) estimator that identifies the set of neighbors of $\mathbf{x}_t$ in the training set and computes category vector $y(t)$ using equation (2). H_1^l and H_0^l are the events where $\mathbf{x}_t$ is assigned and is not assigned, respectively, with label l, and E_c^l the event where $\mathbf{x}_t$ has c instances (c previously

computed) with label l assigned among its neighbors.

$$y(t) = \arg\max_{b\in[0,1]} P(H_b^l|E_c^l) \tag{2}$$

Using the Bayesian rule, we compute $P(H_b^l|E_c^l)$ in Eq. (2) using the prior probability and posterior probability $P(H_b^l)$ and $P(E_c^l|H_b^l)$, respectively.

One important aspect of our proposal is that during the testing time, the complexity of ML-LOCALkNN is not increased compared with the complexity of ML-kNN; thus, the fast testing time is retained. This is one of the most remarkable features of the ML-LOCALkNN.

In our experiments, we show results using the standard ML-kNN and locally adaptive ML-kNN (LAML-kNN). LAML-kNN assumes that the distribution of neighbors of instance x_i with label l, $C_x(l)$ among k nearest neighbors is significantly related to the location of instance x_i. Thus, the probabilities for assigning a label to the query instance changes depending on the cluster to which the query instance belongs. The method identifies which cluster i should be assigned to the query instance and then determines y_i for label l following the maximum a posteriori principle for cluster i.

Algorithm 1: ML-LOCALkNN training phase

Data: A training set $T = \{(\mathbf{x}_1, Y_1), ..., (\mathbf{x}_p, Y_p)\}$, $\mathbf{x}_i \in X$, $Y_i \in Y$; a minimum value of k, k_{min}; a maximum value of k, k_{max}; a maximum value of nearest neighbors $k_{neighbors}$
Result: The vector of local k_i values, $\mathbf{k}$

```
for every x_i ∈ X do
    for every x_j ∈ X do
        for k = 1 to k = k_neighbors do
            if x_j have x_i as its k nearest neighbor then
                Add x_j to N(x_i)
    ' for k = k_min to k = k_max do
        Apply ML_kNN(k) to N(x_i)
    Obtain optimal local value of k ∈ [k_min, k_max]
    Assign optimal value of k to k_i ∈ k
Return k
```

Algorithm 2: ML-LOCALkNN testing phase

Data: A testing set $T = \{(\mathbf{x}_1, Y_1), ..., (\mathbf{x}_p, Y_p)\}$, $\mathbf{x}_q \in X$, $Y_q \in Y$; the number of nearest neighbor prototypes used to calculate the average k, n_{mean}.
Result: Evaluation value, *eval*

```
for every x_q ∈ T do
    Obtain the average k value given by its n_mean nearest neighbor prototypes
    Assign average local k value to k_q ∈ k
Apply ML_kNN(k) to T using the local k_q values for each instance x_q
Obtain evaluation value, eval
Return eval
```

4. Experimental setup

We compare the proposed method with the standard ML-kNN method and the locally adaptative LAML-kNN method. In the experiments, a wide interval of k was selected, where $k_{min} = 1$ and $k_{max} = 100$. For the standard ML-kNN, the value of k was chosen by means of a 10-fold cross-validation in the same interval $k = [1, 100]$. We have also considered values for $k_{\text{neighbors}}$ and n_{mean} in the range $[1, 3]$.

To make a fair comparison among the three methods, we selected a set of 20 problems from Mulan (Tsoumakas et al., 2011b), Meka (Jesse Read and Peter Reutemann, 2016) and Cometa (Charte et al., 2018). These datasets represent a wide range in the number of instances, from 194 to 16 100; number of features, from 10 to 1 836; and number of labels, from 3 to 983. The datasets also contain marked differences in terms of label cardinality and label density. A summary of the characteristics of the datasets is shown in Table 1. To estimate the classifier performance of the three methods, we employed a 10-fold cross-validation approach.

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

Regarding the statistical tests, we selected the Wilcoxon test (Demšar, 2006) as the main statistical test for comparing the pairs of algorithms. This test was chosen because it assumes limited commensurability, is safer than parametric tests, and does not assume a normal distribution or variance homogeneity. The empirical results (Demšar, 2006) indicate that this test is stronger than other tests. We performed the Wilcoxon test with a significance level of $\alpha = 0.05$.

4.1. Evaluation metrics

The evaluation of the 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 (Boutell et al., 2004; Sorower, 2010).

Nevertheless, the traditional metrics from the multiclass problem are also interesting, such as the prediction *Hamming Loss*; thus, we use two exampled-based metrics (Tsoumakas and Katakis, 2007) *Hamming Loss* and F_1 (Tsoumakas et al., 2010).

In addition to these exampled-based metrics, we also use three ranking metrics to evaluate the algorithm's ranking of the different labels for each instance (Schapire and Singer, 2000; Madjarov et al., 2012; Tsoumakas et al., 2010). A description of these evaluation measures is given below.

4.2. Example-based metrics

1. *Hamming Loss* (Tsoumakas and Katakis, 2007) evaluates the number of times that a label that does not belong to an instance is predicted or that a label that belongs to an instance is not predicted.

$$Hamming\,Loss(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{q}|h(\mathbf{x}_i)\Delta Y_i|, \tag{3}$$

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

2. F_1, which is also known as balanced *F-score* or *F-measure* (Tsoumakas et al., 2010), is calculated taking into account both *Precision* and *Recall* values, which are defined as the harmonic mean between them, giving more weight to low *Precision* or *Recall* values, i.e., to false positives and false negatives:

$$F_1(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{2|h(\mathbf{x}_i)\bigcap Z_i|}{|h(\mathbf{x}_i)| + |Y_i|} \tag{4}$$

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

4.3. Ranking-based metrics

The most important ranking-based metrics are given below:

1. *Coverage* (Schapire and Singer, 2000; Madjarov et al., 2012) evaluates how far, on average, we must descend the ranked list to cover all the proper labels of an instance:

$$Coverage(f) = \frac{1}{p}\sum_{i=1}^{p}\max_{y\in Y_i} rank_f(\mathbf{x}_i, y) - 1 \tag{5}$$

With respect to this metric, higher values indicate a decrease in performance.

2. *Ranking Loss* (Madjarov et al., 2012) denotes the average fraction of label pairs that are reversely ordered for an instance and expresses the frequency of irrelevant labels being ranked higher than relevant labels:

$$Ranking\ Loss(f) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i \parallel \overline{Y_i}|}|\{(y_a, y_b) : f(\mathbf{x}_i, y_a) \leq f(\mathbf{x}_i, y_b), (y_a, y_b) \in Y_i \times \overline{Y_i}\}|, \quad (6)$$

where $\overline{Y_i}$ is the complementary set of Y_i. With respect to this metric, the performance is optimal when the value is zero, and higher values indicate a decrease in performance.

3. *Average Precision* (Tsoumakas et al., 2010) evaluates for every $y \in Y_i$ the average of the ground truth labels ranked above y:

$$Average\ Precision(f) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i|}\sum_{y \in Y_i}\frac{|y' \in Y_i : rank_i(y') \leq rank_i(y)|}{rank_i(y)} \quad (7)$$

The performance is optimal when its value is one, and lower values indicate a decrease in performance.

To obtain results for the standard ML-kNN and for LAML-kNN that are comparable to the results of our proposed method, the methods are compared by separately optimizing each metric. For each of the five described metrics, the optimal value of k for methods ML-kNN and LAML-kNN for each metric is obtained by cross-validation ($k \in [1, 100]$) and compared with the result of ML-LOCALkNN optimized for the same metric.

5. Experimental results

In this section, we present and discuss the results obtained by the standard ML-kNN and LAML-kNN method with k obtained by cross-validation and the results of our proposed method. Tables A.4 and A.5 of Appendix A show the results for the example based on the classification metrics *Hamming Loss* and F_1. In addition to the performance metric, the table shows the average value of k obtained in each method. A comparison of the methods by all the metrics discussed above is shown in Table 2 in terms of the win/draw/loss record and the Wilcoxon test. The results, together with that of experiments in Section 5.1, are illustrated in Figs. 1–5.

The first notable result is that the value of the optimum k highly depends on the metric utilized with very different values for each metric. This result supports the use of values obtained from a search. The table shows the suitable performance of our method. For *Hamming Loss*, the ML-LOCALkNN performed better than ML-kNN for 16 of 18 datasets, with only the `yeast` problem yielding suboptimal results. For F_1, our proposal has a distinct advantage, with a win/draw/loss records of 12/0/6. This result means that for example-based metrics, our algorithm performed consistently better than ML-kNN. As ML-LOCALkNN is based on the search for an optimal local value, its performance in regard to the example-based metrics shows the success of the local search to improve the classification performance of ML-kNN. In the case of LAML-kNN, for *Hammnig Loss*, the LAML-kNN performed better and for F_1, our method performed better.

Ranking-based metrics are shown in Tables A.6–A.8 of Appendix A. Although our proposal showed an overall better performance than ML-kNN for the *Ranking Loss* metric and a slightly better performance for *Average Precision*. On the other hand, the ML-kNN performed significantly better than ML-LOCALkNN for *Coverage* (refer to Table 2). The ranking-based measures take into account the whole set of relevant labels, especially *Coverage*, which, until they are in the ranking, the evaluation continues to penalize the labels through the addition of integers.

Table 2
Comparison between ML-kNN, LAML-kNN and our ML-LOCALkNN approach. The table shows the win/draw/loss record, p-value and R^+/R^- values of the Wilcoxon test for each metric: *Hamming Loss* (HL), F_1, *Coverage* (Cov), *Ranking Loss* (RL) and *Average Precision* (AP).

		LAML-kNN			ML-LOCALkNN		
		w/d/l	p-value	R+/R-	w/d/l	p-value	R+/R-
ML-kNN	HL	15/2/1	0.00041	166.5/4.5	16/1/1	0.00030	168.5/2.5
	F_1	10/0/8	0.82756	80.5/90.5	12/0/6	0.01556	141.0/30.0
	Cov	2/0/16	0.00377	19.0/152.0	4/0/14	0.01387	29.0/142.0
	RL	8/0/10	0.87880	82.0/89.0	13/0/5	0.04511	131.5/39.5
	AP	10/0/8	0.82732	80.5/90.5	4/11/3	0.68683	94.5/76.5
LAML-kNN	HL				3/3/12	0.01854	31.5/139.5
	F_1				13/0/5	0.04275	132.0/39.0
	Cov				15/0/3	0.07070	127.0/44.0
	RL				10/2/6	0.28575	110.0/61.0
	AP				8/1/9	0.81063	91.0/80.0

In the particular case of *Coverage*, which had the worst performance in our method, for most datasets, the difference between the two compared rankings is less than one position, which implies that the rankings are very similar. The largest difference between the two rankings is observed in datasets with more than 100 labels, but even the poorest result (dataset `delicious`) decreased the ranking only by 3.6% of the total 983 labels. Therefore, because only a few nearby instances are taken into account to obtain the value of k, these measures may decrease in the case of numerous labels.

As a summary of the results shown in Table 2, we determine for the comparison with ML-kNN that our approach improved the results for 4 of the 5 metrics: three metrics were significantly improved and one metric was non-significantly improved. With respect to LAML-kNN, ML-LOCALkNN improved the results for 3 of the 5 metrics: two metrics were significantly improved and one metric was non-significantly improved. Moreover, although for *Hamming Loss* and *Average Precision* our method obtained a worst performance, the differences among the results were really small (refer to Table A.4 and Table A.8).

The results obtained from comparing ML-kNN with our method are illustrated in Figs. 6–8. Fig. 6 shows the two metrics in which the smallest values indicate the best performance; these metrics include *Hamming Loss* and *Ranking Loss*. Fig. 7 shows the two metrics in which a higher value indicates the better performance; these metrics include F1 and *Average Precision*. The *Coverage* metric is separately shown in Fig. 8 due to its different scale, and, to improve visualization, *bibtex, CAL500, Corel5k* and *delicious* datasets has been discarded because of much higher values than the rest. The results from comparing LAML-kNN and our method are illustrated in Figs. 9–11 in a similar way. For F_1 and *Average Precision*, the points above the main diagonal indicate that our approach achieved a better performance. The remaining metrics points below the main diagonal, $y = x$, indicate that ML-LOCALkNN achieved a better performance.

The *Hamming Loss* and *Ranking Loss*, as shown in Fig. 6, demonstrate the general advantage of ML-LOCALkNN. Most of the points are below the main diagonal $y = x$, particularly for the *Hamming Loss* as it is corroborated by the Wilcoxon test (refer to Table 2).

The comparison for F_1 and *Average Precision*, as shown in Fig. 7, shows a similar pattern of behavior. The advantage in the performance in terms of F_1 is distinct, whereas the results for *Average Precision* are more similar between the two methods. Fig. 8 shows the better behavior of the standard ML-kNN for the *Coverage* metric.

The same plots are shown for LAML-kNN in Figs. 9–11. Fig. 9 shows the comparison of *Hamming* and *Ranking Loss*. For *Ranking Loss*, there is a distinct advantage of our approach, as most of the points are located below the main diagonal. For *Hamming Loss*, our approach obtained a worst performance. However, the plot shows that the differences in both methods were always very small.

Fig. 10 shows the comparison of *Average Precision* and F_1. For F_1, the advantage of ML-LOCALkNN is distinct. Most of the points are located

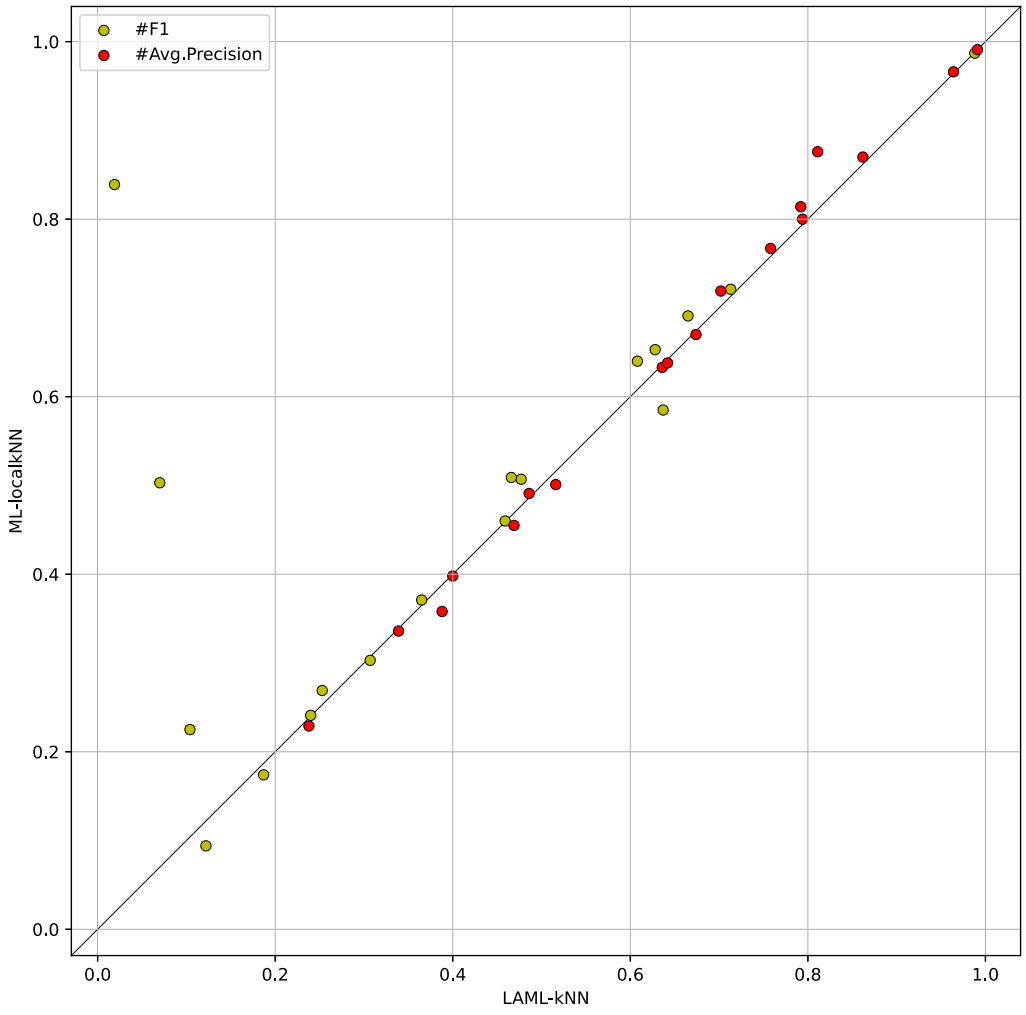


Fig. 10. Comparison of ML-localkNN vs LAML-kNN using F_1 and *Average Precision* metrics.

above the main diagonal, and the differences are large. There was also a better behavior for *Average Precision* but with less marked differences. Fig. 11 shows the comparison in terms of *Coverage* with results that corroborate the previous discussion.

Regarding execution time, our method is fast. As previously stated, to obtain the local k_i values associated with $\mathbf{x}_i$ only a few instances, instances that have $\mathbf{x}_i$ among one of its nearest neighbors, are involved. Furthermore, as the value of k_i does not affect the local values of other instances, every value can be concurrently obtained using an easy parallel implementation. At testing time, once we obtain the optimal value for k for our query instance, our method executes an algorithm similar to ML-kNN.

Comparing our algorithm against standard ML-kNN and its variants in terms of runtime is difficult as ML-kNN has no task similar to our step for obtaining the optimal local k values. However, to obtain the performance for ML-kNN and LAML-kNN showed in the previous tables a 10-fold cross-validation process had to be carried out for both standard methods. That process must be taken into account when considering runtime as using a fixed k value showed poorer performance. Table 3 shows the mean execution time in minutes for ML-localkNN and the standard methods. All the experiments were run in a AMD EPYC 7402 24-Core processor at 1.5 GHz with 512 MB of memory. The table shows that our proposal runs fast even for the largest datasets and much faster that ML-kNN.

5.1. Comparison with other multi-label methods

In the previous section we have shown that ML-localkNN is competitive when compared with other versions of ML-kNN methods. However, it is interesting to know how well does our proposal perform when compared with other widely used multi-label methods. For that purpose, we carried out experiments using other classifiers. We compared six multilabel classification methods: binary relevance (Boutell et al., 2004) (BR), Classifier chains (Read et al., 2011) (CC), RAndom k-labELsets (Tsoumakas et al., 2011a) (RAkEL) with overlapping and non-overlapping labelsets, Label powerset (Tsoumakas et al., 2011a) (LP), MLARAM (Benites and Sapozhnikova, 2015) and Deep Learning (DL) (Maxwell et al., 2017). RAkEL with non-overlapping labelsets consistently outperformed RAkEL with overlapping labelsets so the results of the latter are not reported. The methods were implemented using `scikit multi-learn` (Szymański and Kajdanowicz, 2017). The hyper-parameters were obtained by means of 10-fold cross-validation. For BR, RAkEL, CC and LP we used decision trees as multiclass classifiers.

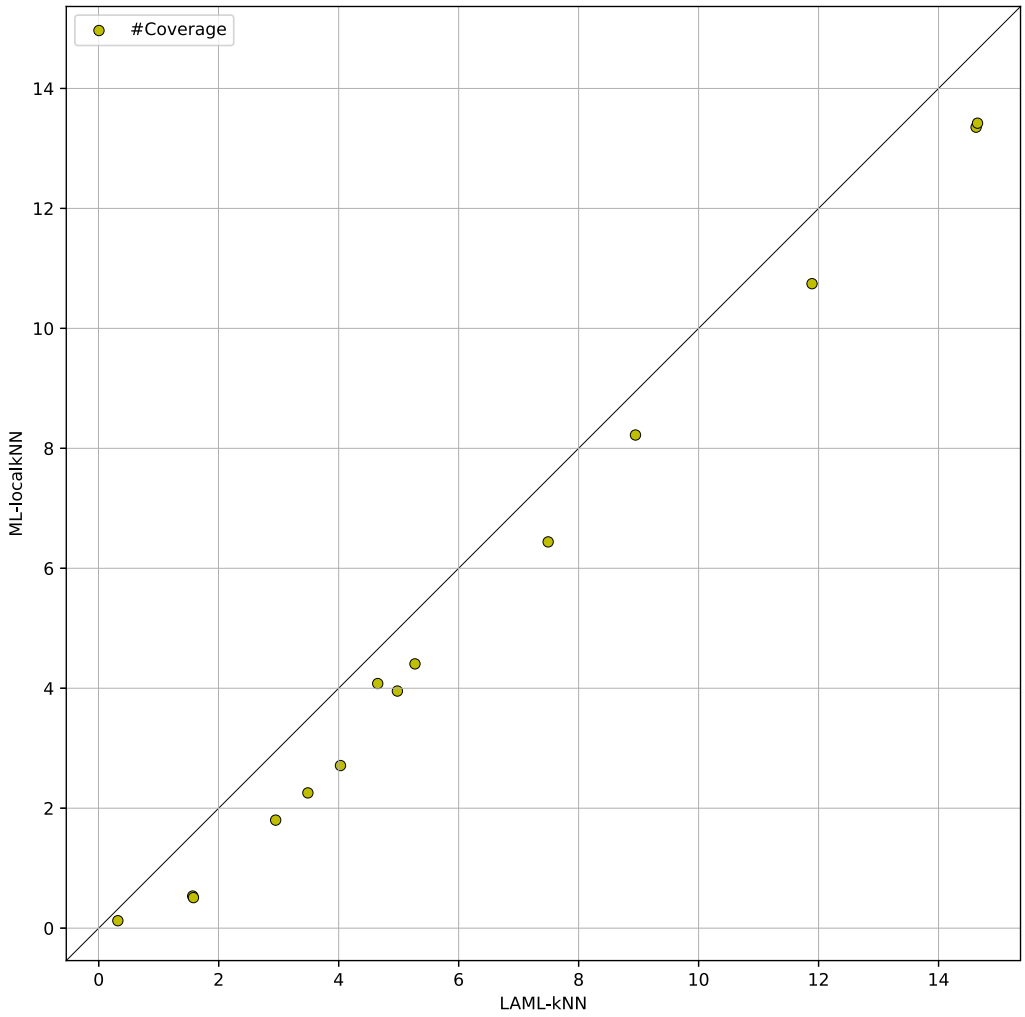


Fig. 11. Comparison of ML-localkNN vs LAML-kNN using *Coverage* metric.

Table 3
Mean execution time in minutes for ML-kNN, LAML-kNN and ML-localkNN.

Dataset	ML-kNN	LAML-kNN	ML-localkNN
bibtex	12623.50	583.52	59.53
birds	17.97	14.36	0.15
CAL500	13.10	51.42	0.54
Corel5k	1719.88	841.41	16.04
delicious	21268.30	5763.71	178.33
emotions	5.65	11.38	0.04
enron	366.73	50.79	2.21
flags	0.58	2.39	0.00
genbase	68.59	26.39	0.32
LLOG-F	273.77	74.65	1.80
medical	177.38	24.72	0.98
Music	5.39	11.19	0.04
OHSUMED-F	24866.30	422.49	119.70
REUTERS-K500-EX2	2535.36	275.73	12.31
scene	234.68	54.22	1.17
SLASHDOT-F	1830.78	50.21	10.25
solar_flare-L3	0.93	3.83	0.00
yeast	119.45	47.37	0.77

For this comparison, as it involved many different methods, we first applied the Iman–Davenport test to ascertain whether there were significant differences between 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. When the Iman–Davenport test rejects a null hypothesis, we proceed with a post hoc Nemenyi test (Nemenyi, 1963), 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:

$$CD = q_\alpha \sqrt{k(k+1)\frac{6}{N}}, \tag{8}$$

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 use the plots described by Demšar (Demšar, 2006). When comparing the algorithms against one another, we connect each group of algorithms that are not significantly different with a horizontal line. We also show the critical difference above the graph. Detailed results are shown in Tables A.9–A.13 of Appendix A.

Fig. 12 shows the comparison in terms of Nemenyi test for the five studied metrics. The p-values of the Iman–Davenport test were always below the critical level. The plot shows the good performance of our approach. For three of the metrics, F1, Ranking Loss and Average Precision, ML-localkNN was the best performing method in terms of Friedman's ranks. For the remaining two metrics, Hamming Loss and

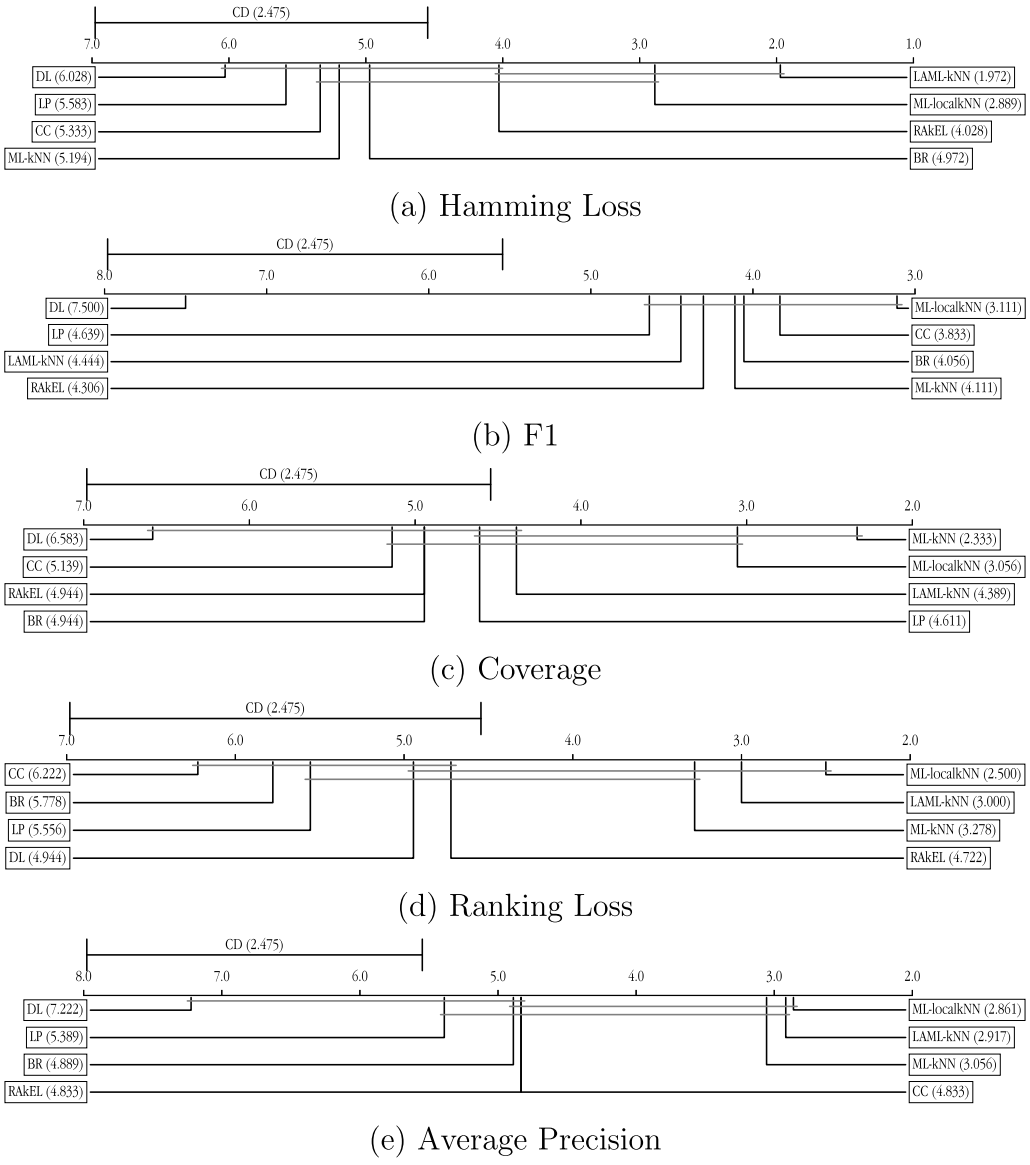


Fig. 12. Nemenyi test for the different classification methods and the five performance metrics.

Coverage, it was the second best method. Thus, as a whole it had the best overall performance of the 8 methods tested.

6. Conclusions and future research lines

In this paper, we have presented a new method for introducing a local value of k for the ML-kNN classifier in multi-label problems. The method consists of associating a local value of k with every prototype and using that k value for any query instance whose nearest neighbor is the prototype. Our method performed better than the standard ML-kNN algorithm and LAML-kNN, a new version that also takes into account local information.

The method was compared with the standard ML-kNN and LAML-kNN using 10-fold cross-validation. The proposed method demonstrated an overall better performance for a set of 18 datasets using five different performance metrics. We also investigated the average values for k obtained by our approach and compared them by the standard cross-validation method.

The time needed to obtain the optimal k value for each prototype was shown to be less with the use of our proposed algorithm, and furthermore, the whole process can be run in parallel, which means that the size of the dataset is not a constraint in our approach.

Many different lines of research can be initiated from this approach. The most obvious means of improving this work is to couple the local value of k with instance selection (Del Castillo et al., 2021). An evolutionary algorithm that simultaneously performs both tasks would be a promising approach. In such an evolutionary method, the value of k for each instance and whether an instance is removed would be simultaneously considered. There are also several works that have developed methods for locally improving the distance metrics used for obtaining the neighbors of an instance (Jiao et al., 2015; Wang et al., 2020). Many distance metrics are available (Alfeilat et al., 2019) and most distance metrics can be modified using weight tuning (Kahraman, 2016). Weight-tuning methods and distance metrics have a significant impact on the k-nearest neighbor-based classification. A major challenge is how to explore the optimal weight values of the features and how to measure distances between the neighbors. Both task are closely related. Kahraman (2016) proposed an heuristic weighting method coupled with a similarity distance metric called fuzzy distance metric. Kumbure et al. (2020) developed also a variant of the fuzzy

Table A.4
Hamming Loss metric results for ML-kNN, LAML-kNN and our proposal.

	ML-kNN		LAML-kNN		ML-localkNN	
	HL ↓	k	HL ↓	k	HL ↓	k
bibtex	0.023	1	0.013	24.8	0.015	7.0
birds	0.084	1	0.054	21.4	0.057	7.5
CAL500	0.193	1	0.142	20	0.150	8.4
Corel5k	0.016	1	0.009	24	0.011	7.7
delicious	0.027	1	0.018	24.2	0.019	8.0
emotions	0.293	1.7	0.270	11.2	0.267	15.3
enron	0.071	1	0.050	18.3	0.054	8.4
flags	0.337	5.3	0.303	8.3	0.323	16.4
genbase	0.001	1	0.001	1.1	0.001	5.0
LLOG-F	0.026	1	0.015	21.3	0.017	6.9
medical	0.020	1	0.010	16.3	0.015	11.1
Music	0.200	6.4	0.200	10.3	0.196	15.6
OHSUMED-F	0.097	2.9	0.070	24.9	0.081	6.5
REUTERS-K500-EX2	0.015	1	0.011	9.4	0.011	10.9
scene	0.097	3	0.088	11.2	0.087	9.9
SLASHDOT-F	0.072	1	0.053	24.9	0.061	7.3
solar_flare-L3	0.091	2.5	0.081	15.7	0.082	5.7
yeast	0.198	11.4	0.201	14.7	0.201	12.1

Table A.5
F_1 metric results for ML-kNN, LAML-kNN and our proposal.

	ML-kNN		LAML-kNN		ML-localkNN	
	F_1 ↑	k	F_1 ↑	k	F_1 ↑	k
bibtex	0.205	23.6	0.240	1.3	0.241	5.9
birds	0.512	5.3	0.070	1.2	0.503	5.8
CAL500	0.346	25	0.365	2.3	0.371	6.2
Corel5k	0.099	4.7	0.122	1	0.094	5.9
delicious	0.225	25	0.253	2	0.269	6.0
emotions	0.508	9.4	0.477	2.2	0.507	9.4
enron	0.494	12.4	0.466	16	0.509	6.8
flags	0.668	23.9	0.665	9.9	0.691	10.5
genbase	0.968	25	0.988	1	0.987	5.0
LLOG-F	0.210	10.4	0.104	1	0.225	5.9
medical	0.601	21.8	0.637	1.9	0.585	8.9
w Music	0.636	25	0.628	7.3	0.653	9.8
OHSUMED-F	0.183	13.7	0.187	1.8	0.174	5.7
REUTERS-K500-EX2	0.418	13.9	0.459	1.9	0.460	7.3
scene	0.709	24.2	0.713	4.3	0.721	7.7
SLASHDOT-F	0.287	9	0.307	1	0.303	5.4
solar_flare-L3	0.838	25	0.019	3.3	0.839	5.5
yeast	0.649	25	0.608	11.2	0.640	8.6

Table A.6
Coverage metric results for ML-kNN, LAML-kNN and our proposal.

	ML-kNN		LAML-kNN		ML-localkNN	
	COV ↓	k	COV ↓	k	COV ↓	k
bibtex	57.217	25	40.155	1.2	64.827	5.4
birds	3.983	1.5	4.651	11.6	4.078	6.5
CAL500	129.612	1	134.801	14.3	131.921	5.1
Corel5k	117.275	22.4	116.974	1	118.092	5.6
delicious	585.312	24.9	603.863	1	621.940	8.1
emotions	2.267	8.1	3.487	1.1	2.255	14.0
enron	13.415	6.2	14.626	18.7	13.354	6.6
flags	3.947	10	4.979	6.6	3.953	11.5
genbase	0.550	1.8	1.567	20	0.535	5.0
LLOG-F	13.637	6.6	14.650	1	13.421	5.7
medical	2.690	1.6	4.030	1	2.711	5.9
Music	1.794	8.8	2.950	1	1.801	13.4
OHSUMED-F	7.309	24.2	8.948	3.5	8.222	5.4
REUTERS-K500-EX2	8.557	23.8	11.891	1	10.745	7.5
scene	0.475	8.7	1.579	1	0.509	9.0
SLASHDOT-F	4.208	19.1	5.272	1.9	4.406	6.7
solar_flare-L3	0.118	2.8	0.319	9.5	0.124	5.3
yeast	6.332	2.9	7.492	1.9	6.440	8.6

Table A.7
Ranking Loss metric results for ML-kNN, LAML-kNN and our proposal.

	ML-kNN		LAML-kNN		ML-localkNN	
	RL ↓	k	RL ↓	k	RL ↓	k
bibtex	0.286	1.7	0.168	1.4	0.260	5.6
birds	0.160	1	0.162	11.7	0.162	7.0
CAL500	0.197	10.9	0.214	1.9	0.212	5.3
Corel5k	0.145	1	0.141	7.8	0.146	5.7
delicious	0.159	1	0.137	1	0.142	7.3
emotions	0.295	2.5	0.303	1.1	0.273	15.8
enron	0.098	5.5	0.101	1	0.097	7.0
flags	0.260	1	0.282	1.2	0.251	15.7
genbase	0.007	1	0.006	19.8	0.006	5.0
LLOG-F	0.153	1	0.145	2.2	0.146	5.8
medical	0.039	1	0.050	1	0.036	6.2
Music	0.196	3.1	0.193	1	0.168	15.3
OHSUMED-F	0.258	1	0.268	3.1	0.279	5.5
REUTERS-K500-EX2	0.096	2.1	0.076	1	0.072	8.2
scene	0.096	4.5	0.098	1	0.085	9.3
SLASHDOT-F	0.197	1.2	0.182	1.9	0.187	7.0
solar_flare-L3	0.039	1	0.048	9.6	0.047	5.2
yeast	0.190	1	0.203	1	0.178	9.3

k-NN rule (Keller et al., 1985) using Bonferroni mean. Although these methods might be related to ours, they are still to be developed for multi-label learning. Thus an interesting future research lines is testing how the combination of both approaches can obtain a better method.

CRediT authorship contribution statement

J.A. Romero-del-Castillo: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Validation. **Manuel Mendoza-Hurtado:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Validation. **Domingo Ortiz-Boyer:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Validation. **Nicolás García-Pedrajas:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Validation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Nicolas Garcia-Pedrajas reports financial support was provided by This work was supported in part by Project TIN2015-66108-P of the Spanish Ministry of Science and Innovation.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by grant PID2019-109481GB-I00 of the Spanish Ministry of Science and Innovation and grant UCO-1264182 of the Junta de Andalucía Excellence in Research Program and FEDER Funds.

Appendix. Tables with detailed results

In this appendix tables with detailed results are shown (see Tables A.4 and A.13).

Table A.8
Average Precision metric results for ML-kNN, LAML-kNN and our proposal.

	ML-kNN		LAML-kNN		ML-localkNN	
	Avg.P ↑	k	Avg.P ↑	k	Avg.P ↑	k
bibtex	0.358	25	0.388	25	0.358	25
birds	0.670	25	0.674	10.3	0.670	23.4
CAL500	0.491	25	0.486	22.1	0.491	25
Corel5k	0.229	25	0.238	25	0.229	25
delicious	0.336	25	0.339	25	0.336	24.7
emotions	0.717	24.7	0.702	8.8	0.719	24.4
enron	0.633	25	0.636	22.5	0.633	25
flags	0.807	23	0.792	19.9	0.814	25
genbase	0.989	25	0.991	1.4	0.991	25
LLOG-F	0.398	24.9	0.400	23.6	0.398	25
medical	0.875	25	0.811	13.8	0.876	24.6
Music	0.801	22.1	0.794	13.9	0.800	24.2
OHSUMED-F	0.455	25	0.469	25	0.455	25
REUTERS-K500-EX2	0.640	25	0.642	16.9	0.638	16.7
scene	0.872	25	0.862	16.3	0.870	18.1
SLASHDOT-F	0.501	25	0.516	24.7	0.501	25
solar_flare-L3	0.966	25	0.964	13.5	0.966	25
yeast	0.767	25	0.758	13.8	0.767	24.9

Table A.9
Hamming Loss metric results for Binary relevance, Classifier chains, Label powerset, RAkEL and Deep learning methods.

Dataset	BR	CC	LP	RAkEL	DL
bibtex	0.0169	0.0171	0.0193	0.0155	0.0148
birds	0.1093	0.1053	0.1134	0.1066	0.2861
CAL500	0.2102	0.2073	0.1723	0.1924	0.4120
Corel5k	0.0123	0.0118	0.0155	0.0111	0.0095
delicious	0.0206	0.0238	0.0353	0.0195	0.0194
emotions	0.2361	0.2417	0.2611	0.2361	0.4500
enron	0.0651	0.0719	0.0645	0.0601	0.1255
flags	0.2643	0.2500	0.3143	0.2857	0.5357
genbase	0.0017	0.0017	0.0017	0.0022	0.3201
LLOG-F	0.0282	0.0295	0.0256	0.0248	0.1191
medical	0.0107	0.0111	0.0168	0.0107	0.1218
Music	0.2611	0.2528	0.2500	0.2806	0.3889
OHSUMED-F	0.0611	0.0610	0.0756	0.0594	0.0634
REUTERS-K500-EX2	0.0235	0.0242	0.0236	0.0223	0.0139
scene	0.1404	0.1390	0.1349	0.1369	0.1404
SLASHDOT-F	0.0687	0.0673	0.0823	0.0667	0.0548
solar_flare-L3	0.4000	0.5333	0.2000	0.2000	0.4667
yeast	0.2441	0.2618	0.2597	0.2503	0.2553
Mean	0.1208	0.1284	0.1148	0.1101	0.2110

Table A.10
F1 metric results for Binary relevance, Classifier chains, Label powerset, RAkEL and Deep learning methods.

Dataset	BR	CC	LP	RAkEL	DL
bibtex	0.3953	0.3942	0.1509	0.3972	0.0210
birds	0.4565	0.4912	0.3367	0.3619	0.1989
CAL500	0.3367	0.3523	0.3035	0.3358	0.2184
Corel5k	0.0963	0.1042	0.1171	0.0988	0.0000
delicious	0.2371	0.1534	0.1374	0.2348	0.0000
emotions	0.6159	0.5950	0.5532	0.6219	0.4795
enron	0.4972	0.4914	0.4773	0.5572	0.2755
flags	0.7189	0.7237	0.6533	0.6820	0.4478
genbase	0.9900	0.9913	0.9900	0.9875	0.0608
LLOG-F	0.1809	0.1716	0.2212	0.1704	0.0524
medical	0.8080	0.7963	0.6929	0.7985	0.0458
Music	0.5856	0.5780	0.6314	0.5549	0.3446
OHSUMED-F	0.4376	0.4589	0.3740	0.4087	0.2214
REUTERS-K500-EX2	0.0103	0.0132	0.0129	0.0107	0.0000
scene	0.5674	0.6027	0.6244	0.5760	0.3597
SLASHDOT-F	0.1897	0.2100	0.1955	0.1719	0.0000
solar_flare-L3	0.1333	0.2667	0.7467	0.7467	0.5029
yeast	0.5831	0.5512	0.5724	0.5829	0.5552
Mean	0.4355	0.4414	0.4328	0.4610	0.2102

Table A.11
Coverage metric results for Binary relevance, Classifier chains, Label powerset, RAkEL and Deep learning methods.

Dataset	BR	CC	LP	RAkEL	DL
bibtex	58.8311	56.5581	67.8122	55.7527	74.1919
birds	7.4103	7.3846	7.2564	9.3846	11.0769
CAL500	148.2745	150.3333	131.2157	152.7647	165.8235
Corel5k	151.8360	147.5660	115.0340	164.0880	149.3860
delicious	714.3599	771.4966	661.6358	703.5426	684.5214
emotions	2.1500	2.4833	2.6000	2.2000	2.8667
enron	21.9415	23.2456	19.1462	23.2573	30.8889
flags	3.6500	3.4000	3.7000	3.5000	5.4000
genbase	0.2090	0.2090	0.4925	0.2090	15.2388
LLOG-F	29.4344	29.5902	20.1148	32.3689	35.9262
medical	5.8061	5.2449	5.9082	4.6429	27.4898
Music	2.2167	2.4833	2.4333	2.6500	3.0500
OHSUMED-F	6.1149	6.1450	6.9060	5.9440	5.6116
REUTERS-K500-EX2	57.5617	55.5833	57.9400	55.6867	52.8133
scene	1.4481	1.3942	0.9668	1.1328	0.7386
SLASHDOT-F	5.8179	6.0475	6.3668	5.8813	9.1662
solar_flare-L3	0.4000	0.8000	0.4000	0.4000	0.8000
yeast	9.2355	8.6529	6.5496	8.2521	8.2231
Mean	68.1499	71.0343	62.0266	68.4254	71.2896

Table A.12
Ranking Loss metric results for Binary relevance, Classifier chains, Label powerset, RAkEL and Deep learning methods.

Dataset	BR	CC	LP	RAkEL	DL
bibtex	0.3332	0.3344	0.2961	0.2674	0.2715
birds	0.3879	0.3802	0.3267	0.3457	0.6683
CAL500	0.6352	0.6404	0.4251	0.5735	0.4110
Corel5k	0.5293	0.5685	0.5632	0.5169	0.5137
delicious	0.1526	0.2703	0.1285	0.1442	0.1111
emotions	0.3139	0.4153	0.3944	0.3722	0.3000
enron	0.2572	0.2880	0.1853	0.2350	0.2360
flags	0.4667	0.4146	0.5958	0.5250	0.5487
genbase	0.0000	0.0000	0.0075	0.0000	0.1928
LLOG-F	0.6168	0.6216	0.3982	0.5377	0.7393
medical	0.1505	0.1760	0.1216	0.1480	0.3444
Music	0.3972	0.3806	0.3968	0.3690	0.3005
OHSUMED-F	0.1814	0.1802	0.2951	0.1750	0.2086
REUTERS-K500-EX2	0.8538	0.8530	0.9044	0.8787	0.8950
scene	0.2863	0.2687	0.1940	0.2448	0.0685
SLASHDOT-F	0.1380	0.1521	0.1490	0.1206	0.0911
solar_flare-L3	0.4000	0.4000	0.4000	0.4000	0.4000
yeast	0.3543	0.3699	0.2800	0.3320	0.3086
Mean	0.3586	0.3730	0.3368	0.3436	0.3672

Table A.13
Average Precision metric results for Binary relevance, Classifier chains, Label powerset, RAkEL and Deep learning methods.

Dataset	BR	CC	LP	RAkEL	DL
bibtex	0.4392	0.4372	0.2472	0.4595	0.1987
birds	0.5137	0.5313	0.5328	0.4437	0.2629
CAL500	0.3003	0.3289	0.4853	0.3022	0.1797
Corel5k	0.2096	0.2312	0.2262	0.2235	0.2046
delicious	0.2984	0.2419	0.2400	0.3080	0.2240
emotions	0.7505	0.6943	0.7005	0.7446	0.5956
enron	0.5607	0.5194	0.6128	0.5782	0.3162
flags	0.8085	0.8345	0.7388	0.7815	0.5667
genbase	1.0000	1.0000	0.9932	1.0000	0.0936
LLOG-F	0.2249	0.2321	0.2810	0.2135	0.1017
medical	0.8164	0.8084	0.7445	0.8243	0.0806
Music	0.7175	0.7093	0.7491	0.6847	0.5863
OHSUMED-F	0.6263	0.6247	0.5038	0.6163	0.5923
REUTERS-K500-EX2	0.0576	0.0612	0.0530	0.0601	0.0580
scene	0.6994	0.7186	0.7563	0.7281	0.7855
SLASHDOT-F	0.4172	0.4193	0.3653	0.3972	0.1667
solar_flare-L3	0.8000	0.6000	0.8000	0.8000	0.6667
yeast	0.6134	0.6150	0.7352	0.6493	0.6753
Mean	0.5474	0.5337	0.5425	0.5453	0.3530

References

Afzal, A., Mussa, H., Turner, R., Bender, A., Glen, R., 2015. A multi-label approach to target prediction taking ligand promiscuity into account. J. Cheminformatics 7, http://dx.doi.org/10.1186/s13321-015-0071-9.

Alberghini, G., Barbon Junior, S., Cano, A., 2022. Adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label drifting data streams. Neurocomputing 481, 228–248. http://dx.doi.org/10.1016/j.neucom.2022.01.075, https://www.sciencedirect.com/science/article/pii/S0925231222000984.

Aldrees, A., Chikh, A., Berri, J., 2016. Comparative evaluation of four multi-label classification algorithms in classifying learning objects. Comput. Sci. Inform. Technol. 6, 651–660.

Alfeilat, H.A.A., Hassanat, A.B.A., Lasassmeh, O., Tarawneh, A.S., Alhasanat, M.B., Salman, H.S.E., Prasath, V.B.S., 2019. Effects of distance measure choice on k-nearest neighbor classifier performance: A review. Big Data 7, 221–248.

Benites, F., Sapozhnikova, E., 2015. Haram: A hierarchical aram neural network for large-scale text classification. In: 2015 IEEE International Conference on Data Mining Workshop. ICDMW, pp. 847–854. http://dx.doi.org/10.1109/ICDMW.2015.14.

Boutell, M.R., Luo, J., Shen, X., Brown, C.M., 2004. Learning multi-label scene classification. Pattern Recognit. 37, 1757–1771.

Charte, F., Rivera, A.J., Charte, D., del Jesus, M.J., Herrera, F., 2018. Tips, guidelines and tools for managing multi-label datasets: the mldr.datasets R package and the cometa data repository. Neurocomputing http://dx.doi.org/10.1016/j.neucom.2018.02.011.

Chen, W., Yan, J., Zhang, B., Chen, Z., Yang, Q., 2007. Document transformation for multi-label feature selection in text categorization. In: Seventh IEEE International Conference on Data Mining. ICDM 2007, pp. 451–456.

Clare, A., King, R.D., 2001. Knowledge discovery in multi-label phenotype data. In: De Raedt, L., Siebes, A. (Eds.), Principles of Data Mining and Knowledge Discovery. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 42–53.

Del Castillo, J.A.R., Ortiz-Boyer, D., García-Pedrajas, N., 2021. Instance selection for multi-label learning based on a scalable evolutionary algorithm. In: 2021 International Conference on Data Mining Workshops. ICDMW, pp. 843–851. http://dx.doi.org/10.1109/ICDMW53433.2021.00108.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Elisseeff, A., Weston, J., 2001. A kernel method for multi-labelled classification. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic. MIT Press, Cambridge, MA, USA, pp. 681–687.

García-Pedrajas, N., Cerruela-García, G., 2021. Cooperative coevolutionary instance selection for multilabel problems. Knowl.-Based Syst. 234, 107569.

García-Pedrajas, N., del Castillo, J.A.R., Cerruela-García, G., 2017. A proposal for local k values for k-nearest neighbor rule. IEEE Trans. Neural Netw. Learn. Syst. 28, 470–475.

Jesse Read, Peter Reutemann, B.P.G.H., 2016. Meka: A multi-label/multi-target extension to weka. J. Mach. Learn. Res. 17, 1–5.

Jiao, L., Pan, Q., Feng, X., 2015. Multi-hypothesis nearest-neighbor classifier based on class-conditional weighted distance metric. Neurocomputing 151, 1468–1476. http://dx.doi.org/10.1016/j.neucom.2014.10.039.

Kahraman, H.T., 2016. A novel and powerful hybrid classifier method: Development and testing of heuristic k-nn algorithm with fuzzy distance metric. Data Knowl. Eng. 103, 44–59.

Keller, J.M., Gray, M.R., Jr, J.A.G., 1985. A fuzzy k-nearest neighbor algorithm. IEEE Trans. Syst. Man Cybern. SMC-15 580–585.

Kumbure, M.M., Luukka, P., Collan, M., 2020. A new fuzzy k-nearest neighbor classifier based on the bonferroni mean. Pattern Recognit. Lett. 140, 172–178.

Lai, H., Yan, P., Shu, X., Wei, Y., Yan, S., 2016. Instance-aware hashing for multi-label image retrieval. IEEE Trans. Image Process. 25 (1), http://dx.doi.org/10.1109/TIP.2016.2545300.

Liu, S.M., Chen, J.H., 2015. A multi-label classification based approach for sentiment classification. Expert Syst. Appl. 42.

Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S., 2012. An extensive experimental comparison of methods for multi-label learning. Pattern Recognit. 45, 3084–3104. http://dx.doi.org/10.1016/j.patcog.2012.03.004.

Maxwell, A., Li, R., Yang, B., Weng, H., Ou, A., Hong, H., Zhou, Z., Gong, P., Zhang, C., 2017. Deep learning architectures for multi-label classification of intelligent health risk prediction. BMC Bioinform. 18, 523. http://dx.doi.org/10.1186/s12859-017-1898-z.

Nemenyi, P.B., 1963. Distribution-Free Multiple Comparisons (Ph.D. thesis). Princeton University.

Read, J., Bifet, A., Holmes, G., Pfahringer, B., 2012. Scalable and efficient multi-label classification for evolving data streams. Mach. Learn. 88, http://dx.doi.org/10.1007/s10994-012-5279-6.

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2011. Classifier chains for multi-label classification. Mach. Learn. 85, 333–359.

Roseberry, M., Krawczyk, B., Djenouri, Y., Cano, A., 2021. Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams. Neurocomputing 442, 10–25.

Schapire, R.E., Singer, Y., 2000. BoosTexter: A boosting-based system for text categorization. Mach. Learn. 39, 135–168.

Sorower, M.S., 2010. A Literature Survey on Algorithms for Multi-Label Learning. Oregon State University, Ph. D Qualifying Review Paper. Major Professor: Thomas G. Dietterich, Ph.D, Computer Science.

Sumbul, G., Demir, B., 0000. A deep multi-attention driven approach for multi-label remote sensing image classification. IEEE Access. 8. pp. 95934–95946. http://dx.doi.org/10.1109/ACCESS.2020.2995805.

Szymański, P., Kajdanowicz, T., 2017. A scikit-based python environment for performing multi-label classification. ArXiv e-prints arXiv:1702.01460.

Tang, L., Rajan, S., Narayanan, V., 2009. Large scale multi-label classification via metalabeler. In: Proceedings of the 18th International Conference on World Wide Web. pp. 211–220. http://dx.doi.org/10.1145/1526709.1526738.

Thabtah, F., Cowling, P., Peng, Y., 2004. Mmac: A new multi-class, multi-label associative classification approach. In: Proceedings - Fourth IEEE International Conference on Data Mining. ICDM 2004, pp. 217–224. http://dx.doi.org/10.1109/ICDM.2004.10117.

Toledano, J.P.P., García-Pedrajas, N., Cerruela-García, G., 2019. Multilabel and missing label methods for binary quantitative structure–activity relationship models: An application for the prediction of adverse drug reactions. J. Chem. Inform. Model. 59, 4120–4130.

Tsoumakas, G., Katakis, I., 2007. Multi-label classification: an overview. Int. J. Data Warehous. Min. 3, 1–13.

Tsoumakas, G., Katakis, I., Vlahavas, I., 2008. Effective and efficient multilabel classification in domains with large number of labels. In: Proceedings of ECML/PKDD 2008 Workshop on Mining Multidimensional Data. MMD08, pp. 30–44.

Tsoumakas, G., Katakis, I., Vlahavas, I., 2010. Mining Multi-Label Data. Springer, pp. 667–685. http://dx.doi.org/10.1007/978-0-387-09823-4_34.

Tsoumakas, G., Katakis, I., Vlahavas, I., 2011a. Random k-labelsets for multi-label classification. IEEE Trans. Knowl. Data Eng. 23, 1079–1089.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I., 2011b. Mulan: A java library for multi-label learning. J. Mach. Learn. Res. 12, 2411–2414.

Ueda, N., Saito, K., 2003. Parametric mixture models for multi-label text. In: Becker, S., Thrun, S., Obermayer, K. (Eds.), Advances in Neural Information Processing Systems. Vol. 15. MIT Press, pp. 721–728.

Veloso, A., Meira, W., Gonçlves, M., 2007. Multi-label lazy associative classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (Eds.), Knowledge Discovery in Databases: PKDD 2007. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 605–612.

Wang, D., Wang, J., Hu, F., Guoqing, L., Zhang, X., 2018a. A Locally Adaptive Multi-Label K-Nearest Neighbor Algorithm. pp. 81–93. http://dx.doi.org/10.1007/978-3-319-93034-3_7.

Wang, D., Wang, J., Hu, F., Li, L., Zhang, X., 2018b. A locally adaptive multi-label k-nearest neighbor algorithm. In: Proceedings of the 22nd Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. PAKDD, pp. 81–93.

Wang, Z.W., Wang, S.K., Wan, B., Song, W.W., 2020. A novel multi-label classification algorithm based on k-nearest neighbor and random walk. Int. J. Distrib. Sens. Netw. 16, http://dx.doi.org/10.1177/1550147720911892.

Wang, D., Zhang, S., 2020. Unsupervised person re-identification via multi-label classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR.

Yu, Y., Pedrycz, W., Miao, D., 2013. Neighborhood rough sets based multi-label classification for automatic image annotation. Internat. J. Approx. Reason. 54, 1373–1387. http://dx.doi.org/10.1016/j.ijar.2013.06.003, http://www.sciencedirect.com/science/article/pii/S0888613X13001308.

Zhang, D., Islam, M.M., Lu, G., 2012. A review on automatic image annotation techniques. Pattern Recognit. 45, 346–362. http://dx.doi.org/10.1016/j.patcog.2011.05.013, http://www.sciencedirect.com/science/article/pii/S0031320311002391.

Zhang, Y., Wang, Y., Liu, X.Y., Mi, S., Zhang, M.L., 2020b. Large-scale multi-label classification using unknown streaming images. Pattern Recognit. 99, 107100. http://dx.doi.org/10.1016/j.patcog.2019.107100, https://www.sciencedirect.com/science/article/pii/S0031320319304017.

Zhang, D., Zhao, S., Duan, Z., Chen, J., Zhang, Y., Tang, J., 2020a. A multi-label classification method using a hierarchical and transparent representation for paper-reviewer recommendation. ACM Trans. Inf. Syst. 38, http://dx.doi.org/10.1145/3361719.

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

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

Zheng, X., Li, P., Chu, Z., Hu, X., 2020. A survey on multi-label data stream classification. IEEE Access 8, 1249–1275. http://dx.doi.org/10.1109/ACCESS.2019.2962059.

Zufferey, D., Hofer, T., Hennebert, J., Schumacher, M., Ingold, R., Bromuri, S., 2015. Performance comparison of multi-label learning algorithms on clinical data for chronic diseases. Comput. Biol. Med. 65, 34–43.