Advances in Data Analysis and Classification (2024) 18:723–744
https://doi.org/10.1007/s11634-024-00589-3

REGULAR ARTICLE

Natural-neighborhood based, label-specific undersampling for imbalanced, multi-label data

Payel Sadhukhan[1] · **Sarbani Palit**[2]

Received: 22 September 2022 / Accepted: 3 March 2024 / Published online: 30 March 2024

Abstract

This work presents a novel undersampling scheme to tackle the imbalance problem in multi-label datasets. We use the principles of the natural nearest neighborhood and follow a paradigm of label-specific undersampling. Natural-nearest neighborhood is a parameter-free principle. Our scheme's novelty lies in exploring the parameter-optimization-free natural nearest neighborhood principles. The class imbalance problem is particularly challenging in a multi-label context, as the imbalance ratio and the majority–minority distributions vary from label to label. Consequently, the majority–minority class overlaps also vary across the labels. Working on this aspect, we propose a framework where a single natural neighbor search is sufficient to identify all the label-specific overlaps. Natural neighbor information is also used to find the key lattices of the majority class (which we do not undersample). The performance of the proposed method, NaNUML, indicates its ability to mitigate the class-imbalance issue in multi-label datasets to a considerable extent. We could also establish a statistically superior performance over other competing methods several times. An empirical study involving twelve real-world multi-label datasets, seven competing methods, and four evaluating metrics—shows that the proposed method effectively handles the class-imbalance issue in multi-label datasets. In this work, we have presented a novel label-specific undersampling scheme, NaNUML, for multi-label datasets. NaNUML is based on the parameter-free natural neighbor search and the key factor, neighborhood size 'k' is determined without invoking any parameter optimization.

Keywords Multi-label · Natural nearest neighborhood · Class imbalance · Undersampling

Mathematics Subject Classification 68T10 · 68T20 · 68T30

✉ Payel Sadhukhan
payel0410@gmail.com

1 Institute for Advancing Intelligence, TCG CREST, Kolkata, India

2 Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India

1 Introduction

Class imbalance is a note-worthy characteristic of data obtained from several real-world domains. The naturally occurring biases in the real world give rise to varying numbers of points in different classes of a dataset. Multi-label datasets—mostly obtained from real-world sources (Li et al. 2014; Katakis et al. 2008) is no exception to this. In a multi-label dataset, an instance is associated with more than one possible label. Let $\mathcal{D}$ be a multi-label dataset with $\mathcal{L}$ labels. $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{Y}_i), 1 \leq i \leq n\}$. $\mathbf{x}_i$'s denote the feature vectors and $\mathbf{Y}_i$ denotes its membership to $\mathcal{L}$ labels. $\mathbf{Y}_i = \{y_{i1}, y_{i2}, \ldots, y_{i\mathcal{L}}\}$ and for binary classification, y_{ij} can be either 0 (negative class) or 1 (positive class). The task is to correctly predict the class (0 or 1) for $\mathcal{L}$ labels of a test instance. In a two-class dataset, we term the class with a higher number of instances and the class with a lower number of instances as the majority class and the minority class respectively. In *yeast* dataset (Elisseeff and Weston 2001), the imbalance ratio (ratio of majority set cardinality to that of minority set cardinality) is greater than 1.5 for 12 out of 14 labels. Alternatively, we can say that, for 12 out of 14 labels in *yeast* dataset, one class has 50% more points than the other class. It is also observed that the different labels of a multi-label dataset possess differing degrees of imbalances. This aspect further intricates the issue and calls for dedicated and label-specific handling of the class imbalance issue in a multi-label context.

Data preprocessing is a popular technique for handling the class imbalance of the datasets. This particular technique is motivated to reduce the difference in cardinalities of the classes in a dataset by (1) either removing the points from the majority class (undersampling the majority class) or (2) by adding synthetic points to the minority class (oversampling the minority class). This helps mitigate the bias of the majority class in the classifier modeling phase and helps detect minority instances. In *under-sampling* of data, points are removed from the majority class to reduce the difference in the majority and the minority class cardinalities. It also reduces the overall training data volume, thereby reducing the computation of the classifier modeling. Undersampling is a convenient option for multi-label datasets as their dimensionality is high concerning the number of points and features. We should also remember that—the positive and negative class memberships vary across labels in a multi-label dataset. Even though the feature vectors reside in the same locations of the feature space (for all labels), their changing memberships lead to different majority and minority point configurations.

In this work, we propose a natural neighborhood-based undersampling scheme (NaNUML) to deal with the class imbalance of multi-label datasets. Due to disparate ranges of imbalance ratios and the diversified distributions of majority and minority points across the labels, we resort to a label-specific undersampling. We look at the mutual co-locations of the majority and the minority points within a neighborhood to find the majority candidates to be undersampled. Our principal aim is to find and remove the majority points that overlap with many minority points. Removing the majority points from the overlapped space will increase the cognition of the minority points in those regions.

To find the majority points overlapping in the minority spaces, we employ the technique of natural nearest neighborhood (Zhu et al. 2016). Two points $\mathbf{p}$ and $\mathbf{q}$

are natural neighbors of each other if (1) **p** is a k-nearest neighbor of **q** and also (2) **q** is also a k-nearest neighbor of **p**. Unlike the identification of the neighbors via a directional and one-sided nearness (like that in the k-nearest neighborhood), natural neighbors are computed based on the mutual nearness of two points (hence, *commutative*). The relative nearness of two points (relative to their neighborhood) is instrumental in chalking out the neighborhood relation. The mutual nearness protocol of natural neighborhoods aids in the efficient identification of the majority and minority class overlaps. The other significant advantage of the natural neighborhood scheme is computing the neighborhood size 'k' without human intervention or a parameter optimization phase. This characteristic is helpful in any machine learning context, and our scheme enjoys the advantage. In NaNUML, a single natural nearest neighbor search is sufficient to compute all labels' label-specific natural neighbor information.

For each label, we compute the minority natural neighbor count of the majority points. A high minority neighbor count for a majority point indicates its increased overlap with the minority space (as well as the minority points). Hence, the majority points with higher minority natural neighbor counts are potential candidates for undersampling. Accordingly, we remove the majority points in order of their decreasing minority neighbor count. The majority point with the highest minority neighbor count is removed first. The undersampled majority set and the original minority set form the augmented training set and are used to learn a set of label-specific classifiers.

The major highlights of our work are as follows:

- We undersample the label-specific majority points to obtain an augmented yet reduced training set for each label.
- We employ a parameter-optimization-free technique to compute the neighbors of the points. The computation of the neighbors is based on a mutual nearness calculation, which helps in an enhanced identification of the majority–minority overlaps.
- This is the first work to introduce the paradigm of natural neighborhoods in multi-label learning.
- While undersampling the majority class, we also preserve the key lattice points of the majority class by preserving (and not allowing the undersampling of) the majority points (top 10%) with the highest majority natural neighbor count.
- The natural neighborhood search is not label-dependent and depends on the distribution of the points in the feature space. Hence, only one natural neighbor search is required (for all labels).
- The outcomes from an experimental study involving twelve real-world multi-label datasets, seven competing methods (multi-label learners and generic class-imbalance focused learning paradigms), and four evaluating metrics indicate the proposed method's competence over other competing learners.

2 Related works

This work is focused on the class-imbalance aspect of multi-label learning. The study of the extant works will be devoted to both these aspects—(1) class imbalance learning and (2) multi-label learning in general.

Several diversified approaches are followed in the domain of class-imbalance learning to mitigate the bias of the majority class (He and Garcia 2009). Algorithm-based methods are one of the earliest methods in this field. The methods mostly function in one of two ways—(1) by shifting the boundary away from the minority class to add more region in their favor, or (2) by employing a cost-sensitive learning framework where the misclassification of minority instances incur a higher penalty. Other approaches like kernel-based methods, multi-objective optimization methods, and ensemble-based learners also focus on achieving the same goal.

Data preprocessing is a popular technique of handling the class-imbalance problem (Ali et al. 2019). Here, the schemes are motivated to balance the cardinalities of the majority and the minority classes. This can be done in the following ways—(1) *undersampling* or removing points from the majority class (Pereira et al. 2020; Tahir et al. 2012), (2) *oversampling* or adding synthetic points to the minority class (Charte et al. 2015; Chawla et al. 2002), (3) *hybrid sampling* where both undersampling and oversampling are involved (Choirunnisa and Lianto 2018; Ludera 2021). This step of data sampling occurs before the classification step, and the classifier modeling is done on the augmented data (obtained through preprocessing).

The focus of the researchers on multi-label learning dates back to the beginning of this century (Joachims 1998; Godbole and Sarawagi 2004). The community's ongoing efforts have provided several ways of handling this issue (Moyano et al. 2018).

Multi-label methods are principally classified into (1) *Problem transformation approaches:* in which several classifiers are modeled and learned to facilitate an overall multi-label learning of the data at various levels of label association (they are further classified into *first-order, second-order and higher order* according to the degree of label associations in the classifiers (Zhang and Wu 2015; Sadhukhan and Palit 2020; Tsoumakas et al. 2011; Fürnkranz et al. 2008), and (2) *Algorithm adaptation approaches:* which consider tweaking of an existing classifier like Support Vector Machine, nearest-neighborhood based classifier, random forest to accommodate the multi-label learning (Gonzalez-Lopez et al. 2018; Nam et al. 2014; Liu et al. 2018; Siblini et al. 2018).

The researchers in multi-label were quick to notice the issue of class imbalance in multi-label datasets (Tahir et al. 2012). We should note that handling the class-imbalance issue in multi-label datasets is way more knotty than single-label traditional datasets. The principal causes are (1) the multi-output nature where the degree of imbalance in each output varies from the others and (2) a set of imbalance ratios, one for each label. Data pre-processing, being a popular choice, is explored in multi-label contexts. MLeNN (Charte et al. 2014) uses the edited Nearest Neighbor rule principles to undersample the majority points with similar label sets of its neighbors in a multi-label dataset. In a hybrid sampling technique, ML-RUS deletes the instances belonging to the majority classes of a multi-label dataset. ML-ROS deletes the clone examples with minority labels to facilitate better learning of the imbalanced multi-label datasets (Pereira et al. 2020). ML-SMOTE resorts to the oversampling of minority classes to balance the cardinalities of the majority and the minority classes of the labels (Charte et al. 2015). Liu and Tsoumakas (2020) couples the majority class undersampling with the classifier chain scheme's ensembles to tackle the class imbalance issue. COCOA (Zhang et al. 2020) presents a scheme where the asymmetric distribution of classes and

the pair-wise label correlations are considered, and a three-way learner is produced for each pair of labels. Daniels and Metaxas (2017) exploits the Hellinger forests to design an imbalance-aware multi-label classifier. In LIIML (Sadhukhan and Palit 2019), an imbalance-informed label-specific feature set is constructed for the labels, followed by a cost-sensitive learning scheme to learn the multi-label datasets. In the next section, we briefly describe the intuition and working principles of the natural nearest neighborhood.

3 Principles of natural nearest neighborhood

Let us have a set of points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ and we want to find the natural neighbors of $\mathbf{x}_i$ from the given search space (excluding itself). For some $k = \alpha$, ($\alpha >= 1$), we say that $\mathbf{x}_i$ is a natural neighbor of $\mathbf{x}_j$ (at $k = \alpha$), if $\mathbf{x}_i$ is a α-nearest neighbor of $\mathbf{x}_j$ and $\mathbf{x}_j$ is also a α-nearest neighbor of $\mathbf{x}_i$ (Zhu et al. 2016). Let $NN(\mathbf{x}_j)$ be a natural neighbor of $\mathbf{x}_i$ and $KNN_\alpha(\mathbf{x}_j)$ be a α-nearest neighbor of $\mathbf{x}_i$. $\mathbf{x}_j \in NN(\mathbf{x}_i) \iff (\mathbf{x}_i \in KNN_\alpha(\mathbf{x}_j)) \cap (\mathbf{x}_j \in KNN_\alpha(\mathbf{x}_i))$.

The authors of this work have also stated the procedure for selecting a natural neighbor eigenvalue (λ) (the neighborhood size). In a dataset, the minimum k-value at which all points get at least one natural neighbor is to be noted. Let this critical k-value be β. The natural neighbor eigenvalue, λ is computed from β. According to the authors,

$$\lambda = \sqrt{\beta}$$

Unlike k-nearest neighborhood search or reverse nearest neighborhood search, natural neighborhood search retrieves a symmetric neighborhood configuration of a dataset. We can identify the true majority and minority class overlaps via the symmetric neighborhood or hand-shake configurations. In this work, NaNUML, the nearest neighbor eigenvalue for each dataset, is computed and used in the subsequent stages for undersampling the majority class. The proposed approach is described in the next section.

4 NaNUML approach

Let a multi-label dataset be denoted $\mathcal{D}$, and the number of labels be $\mathcal{L}$.

Algorithm 1 NaNUML

Input: $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{Y}_i), 1 \leq i \leq n\}$, α
Output: Augmented dataset, $\mathcal{UD}_{(j)}$ for $1 \leq j \leq \mathcal{L}$

1: Find λ from the feature space information of $\mathcal{D}$ ▷ It is calculated irrespective of the class information.
2: Find the natural neighbors of $\mathbf{x}_i$, $\mathbf{x}_i \in \mathcal{D}$ acc. to (Equation 1). ▷ It is calculated irrespective of the class information.
3: **for** j **do**=1 to $\mathcal{L}$ do
4: Segregate $\mathcal{D}$ into $\mathcal{D}_{M(j)}$ and $\mathcal{D}_{m(j)}$ (Equation 2). ▷ According to their majority and minority memberships w.r.t. label j
5: Compute u_j acc. to (Equation 3) ▷ Number of points to be undersampled for label j
6: Compute $count_{M(i)(j)}$ and $count_{m(i)(j)}$ acc. (Equation 4). ▷ Number of majority and minority neighbors of instance i w.r.t. label j
7: Mark the points in $\mathcal{D}_{M(j)}$ with highest $count_{M(i)(j)}$ values and do not undersample them.
8: Sort the points in $\mathcal{D}_{M(j)}$ in descending order of their $count_{m(i)(j)}$ values and select the first u_j points in $\mathcal{U}_{(j)}$.
9: Undersample $\mathcal{U}_{(j)}$ from $\mathcal{D}_{M(j)}$ to get the undersampled majority set, $\mathcal{UM}_{(j)}$ (Equation 5).
10: $\mathcal{UD}_{(j)}$ is obtained by taking the union of $\mathcal{D}_{m(j)}$ and $\mathcal{UM}_{(j)}$ (Equation 6).
11: **end for**

$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i),\ 1 \leq i \leq n\}$. $\mathbf{x}_i$ denote the ith feature vector and $\mathbf{y}_i$ denotes its class information corresponding to $\mathcal{L}$ labels. $\mathbf{y}_i = \{y_{i1}, y_{i2}, \ldots, y_{i\mathcal{L}}\}$ and each y_{ij} can be either 0 (negative membership) or 1 (positive membership). Example, $y_{14} = 1$ signifies that $\mathbf{x}_1$ belongs to (has) the positive class of the 4th label. Our primary task is to predict the correct membership of the test points for all the labels.

1. *Finding the natural neighbors of points in $\mathcal{D}$* Following the natural neighbor principles, we find the natural neighbors of all points in $\mathcal{D}$ for $k = \lambda$ (where λ is the natural neighbor eigenvalue). λ is specific to a dataset. Let $N(\mathbf{x}_i)$ be the natural neighbor set of $\mathbf{x}_i$.

$$\mathcal{N}(\mathbf{x}_i) = \{\mathbf{x}_j; (\mathbf{x}_i \in KNN_\lambda(\mathbf{x}_j)) \cap (\mathbf{x}_j \in KNN_\lambda(\mathbf{x}_i))\}, \quad i = 1, 2, \ldots, n \tag{1}$$

   This step is common for all labels as the labels share the same feature points.
2. *Imbalance ratios of the labels and the number of points removed* For each label, the points belonging to the positive and negative classes are segregated into two mutually exclusive sets. In a multi-label dataset, usually, the positive class qualifies as the minority class, and the negative class becomes the majority class. Class inversion can indeed occur, where the negative and positive classes change their roles. But, for clarity and synchronization, we denote the positive and negative classes as the minority and majority classes, respectively. Let $\mathcal{D}_{M(j)}$ and $\mathcal{D}_{m(j)}$ be the majority and the minority classes of label j, respectively.

$$\begin{aligned} \mathcal{D}_{M(j)} &= \{\mathbf{x}_i; 1 \leq i \leq n \text{ and } y_{ij} = 0\} \\ \mathcal{D}_{m(j)} &= \{\mathbf{x}_i; 1 \leq i \leq n \text{ and } y_{ij} = 1\} \\ \mathcal{D} &= \mathcal{D}_{M(j)} \cup \mathcal{D}_{m(j)} \end{aligned} \tag{2}$$

   For each label, we compute the cardinality of the undersampled set from the difference between the cardinalities of the majority and the minority classes. Let u_j be the number of points to be removed from $\mathcal{D}_{M(j)}$. Let α be a number such that $0 < \alpha \leq 1$.

$$u_j = max(\alpha \times (|\mathcal{D}_{M(j)}| - |\mathcal{D}_{m(j)}|), 0), \quad j = 1, 2, \ldots, \mathcal{L} \tag{3}$$

   α allows us to choose the number of points to be removed from the majority point set. When $\alpha = 1$, we equate the cardinality of the undersampled majority point set with that of the minority point set. After the undersampling, the difference in cardinalities of the undersampled majority class and the minority class is equal to the $(1-\alpha)\%$ of the original difference between the two sets. *Note that* When there is an inversion of the positive and the negative class for a label, [majority class (class 0) has lesser number of points than the minority class (class 1)], $(|\mathcal{D}_{M(j)}| - |\mathcal{D}_{m(j)}|)$ will be negative and u_j will be 0. We will not remove any point for that label.
3. *Finding the majority points to be undersampled for each label and generating the augmented dataset* For each label, we find the natural neighbor count of the majority points. The majority point set and the minority point set vary across the labels

depending on the label-specific membership of the points. Additionally, we segregate this count into two mutually exclusive counts—(1) majority natural neighbor count and (2) minority natural neighbor count. Let $count_{Mi(j)}$ and $count_{mi(j)}$ denote the majority natural neighbor count and minority natural neighbor count, respectively, of an instance $\mathbf{x}_i$ for label j.

$$\begin{aligned} count_{M(i)(j)} &= |\{\mathbf{x}_k : (\mathbf{x}_k \in \mathcal{N}_i) \text{ and } (\mathbf{x}_k, \mathbf{x}_i) \in \mathcal{D}_{M(j)}\}| \\ count_{m(i)(j)} &= |\{\mathbf{x}_k : (\mathbf{x}_k \in \mathcal{N}_i) \text{ and } (\mathbf{x}_k \in \mathcal{D}_{m(j)}) \text{ and } (\mathbf{x}_i \in \mathcal{D}_{M(j)})\}| \end{aligned} \tag{4}$$

- *Finding the label-specific majority points, which are the key structural components and preserving them from undersampling* We explore the majority natural neighbor counts to find the key structural points of the majority set. The points with the higher majority natural neighbor counts are selected as the key structural points, and the top 10% points are kept away from the undersampling in the next phase (even if their minority counts are higher).
- *Finding the majority points to be removed from the remaining set of points* For a label j, we look at the minority natural neighbor count of the remaining majority points. The majority point with the highest minority natural neighbor count is removed (undersampled) first from the majority set. This procedure of undersampling is continued (according to the decreasing order of the minority natural neighbor counts of the majority points) till u_j points are removed. *A majority point in a majority class-minority class overlapped region will have a high minority natural neighbor count and is a good candidate for removal.*

Let $\mathcal{U}_{(j)}$ be the set of removed points from the majority set $\mathcal{D}_{M(j)}$. The undersampled majority set for label j, UM_j is obtained by taking the difference of $\mathcal{U}_{(j)}$ from $\mathcal{D}_{M(j)}$.

$$\mathcal{UM}_{(j)} = \mathcal{D}_{M(j)} \backslash \mathcal{U}_{(j)}, \quad j = 1, 2, \ldots, \mathcal{L} \tag{5}$$

The undersampled training set for label j, $\mathcal{UD}_{(j)}$ is obtained by taking the union of $\mathcal{UM}_{(j)}$ and $\mathcal{D}_{m(j)}$.

$$\mathcal{UD}_{(j)} = \mathcal{UM}_{M(j)} \cup \mathcal{D}_{m(j)}, \quad j = 1, 2, \ldots, \mathcal{L} \tag{6}$$

$\mathcal{UD}_{(j)}$ is used to train the label-specific classifier for label j, and the classifier is subsequently used to make the predictions for label j.

Remarks In this work, we suggest preserving 10% majority points as the key structural components of the majority class. In datasets with an imbalance ratio ($r > 10$), this will impose an upper limit on α.

$$\alpha = \frac{0.9r}{r-1} \tag{7}$$

Given that, it is not possible to equate the cardinalities of the minority and the undersampled majority classes when $r > 10$. The experimental results on exploring α

manifest that it is a fair trade-off. Too much removal of majority points can lead to the distortion of the majority class. If it is of utmost necessity to balance the cardinalities of the majority and minority classes, it has to be done by lessening the degree of preservation.

In order, we present the Experimental Setup, Results and Discussion, and Conclusion in the following three sections.

5 Experimental setup

- *Datasets* We have performed the experiments on 12 real-world multi-label datasets enlisted in Table 1.[1] Here, *instances*, *inputs*, and *labels* indicate the cardinality, features, and the number of labels respectively in each dataset. *Type* indicates the nominal or numeric nature of the features. The number of unique label combinations present in a dataset is indicated by *Distinct label sets*. *Cardinality* is the average number of labels per instance, and *Density* is *Cardinality* weighted by the number of labels. We have pre-processed the datasets according to the recommendations in Zhang et al. (2020) and He and Garcia (2009). Labels having a very high degree of imbalance (50 or greater) or having too few positive samples (20 in this case) are removed. For text datasets (*medical*, *enron*, *rcv1-s1*, *rcv1-s2*), only the input space features with a high degree of document frequencies are retained.
- *Comparing algorithms* Seven schemes, comprising of, (1) six multi-label learning schemes and (2) one generic class-imbalance focused learners are employed in the empirical study. The multi-label learners involved in the study are COCOA (Zhang et al. 2020), THRESHL (Pillai et al. 2013), IRUS (Tahir et al. 2012), CLR (Fürnkranz et al. 2008), RAKEL (Tsoumakas et al. 2011) and ECC (Read et al. 2011). In COCOA, several imbalance-focused multi-class learners are implemented in the Weka platform using the J48 decision tree with undersampling, where the number of coupling class labels is set as $\mathrm{K} = \min(\mathcal{L} - 1,\ 10)$. IRUS is a label-specific undersampling scheme like the proposed method, NaNUML where $\mathcal{L}$ are trained, one for each label. Each label-specific classifier is trained using the label-specific undersampled training data. IRUS is an ensemble method and the random undersampling is repeated several times to produce a classifier ensemble. THRESHL also learns in a label-specific setting with one classifier for each label. The scheme of THRESHL is to maximize the F-scores in a hold-out setting to find the threshold for classification. CLR is a second-order learning scheme that exploits pair-wise label correlations to obtain a multi-label learning performance. In ECC, the classification outputs of a label are used as an input feature for predicting the succeeding labels, thereby involving the correlations of the labels. RAKEL is also a higher-order learning approach where the set of overlapping and non-overlapping subsets of labels are considered, and multi-class classifiers are learned on the power set of the labels. RML (Tahir et al. 2012) is a generic class-imbalance learner used in the comparative study. In RML, the macro-averaging F measure is used as the optimization metric while modeling the classifier. In IRUS, the C4.5 decision tree

[1] http://mulan.sourceforge.net/datasets-mlc.html.

Table 1 Description of datasets

Dataset	Instances	Inputs	Labels	Type	Cardinality	Density	Distinct labelsets	Proportion of distinct labelsets	Imbalance ratio		
									Min	Max	Avg
CAL500	502	68	124	Numeric	25.058	0.202	502	1.000	1.040	24.390	3.846
Emotions	593	72	6	Numeric	1.869	0.311	27	0.046	1.247	3.003	2.146
Scene	2407	294	6	Numeric	1.074	0.179	15	0.006	3.521	5.618	4.566
Yeast	2417	103	13	Numeric	4.233	0.325	189	0.078	1.328	12.500	2.778
Image	2000	294	5	Numeric	1.236	0.247	20	0.010	2.448	3.890	3.117
Rcv1-s1	6000	472	42	Numeric	2.458	0.059	574	0.096	3.342	49.000	24.966
Rcv1-s2	6000	472	39	Numeric	2.170	0.056	489	0.082	3.216	47.780	26.370
medical	978	144	14	Nominal	1.075	0.077	42	0.043	2.674	43.478	11.236
Llog	1460	100	18	Nominal	0.851	0.047	109	0.075	7.538	46.097	24.981
Enron	1702	50	24	Nominal	3.113	0.130	547	0.321	1.000	43.478	5.348
Slashdot	3782	53	14	Nominal	1.134	0.081	118	0.031	5.464	35.714	10.989
Corel5k	5000	499	44	Nominal	2.241	0.050	1037	0.207	3.460	50.000	17.857

is used as the base learner. In RAKEL, the recommended settings of $k = 3$ and the number of subsets $m = 2q$ are employed. In ECC, an ensemble size of 100 is chosen. In CLR, a synthetic label is used to differentiate between the relevant and the irrelevant labels. In NaNUML, we have used Support Vector Machine Classifier with linear kernel and the regularization parameter is set to 1.

- *Evaluating metrics* Four multi-label domain-specific metrics, namely—*macro averaging* F_1, *macro-averaging AUC*, *average precision*, and *ranking loss* are used to compute the performance of the proposed and the competing methods. They are briefly described as follows:
  - *Macro-averaging* F_1: it is the average of all the label-specific F_1 scores. Let F_{1j} be the F_1 score for label j. The higher the macro averaging F_1 score, the better the performance.

$$\text{Macro } F_1 = \frac{1}{\mathcal{L}} \sum_{j=1}^{\mathcal{L}} F_{1j} \tag{8}$$

  - *Macro-averaging AUC*: it is the sum of the label-specific AUC scores, weighted by the number of labels $\mathcal{L}$. Let AUC_j be the AUC score for label j. The higher the macro averaging AUC score, the better the learner's performance.

$$\text{Macro } AUC = \frac{1}{\mathcal{L}} \sum_{j=1}^{\mathcal{L}} AUC_j \tag{9}$$

  - *Average precision*: average precision evaluates the average fraction of relevant labels ranked higher than a particular label. It is desirable that, for instance, the relevant labels will be predicted with higher scores (more confidence) than that of the irrelevant or absent ones. Let $\mathcal{R}(\mathbf{x}_i, l_k) = \{l_j | rank(\mathbf{x}_i, l_j) \leq rank(\mathbf{x}_i, l_k), l_j \in \mathbf{Y}_i\}$

$$AveragePrecision = \frac{1}{n} \sum_{1}^{t} \frac{1}{|\mathbf{Y}_i|} \sum \frac{|\mathcal{R}(\mathbf{x}_i, l_k)|}{rank(\mathbf{x}_i, l_k)} \tag{10}$$

  - *Ranking loss*: is used to evaluate the percentage of misordered label pairs. Let $\mathcal{R}(\mathbf{x}_i, l_k) = \{l_j | rank(\mathbf{x}_i, l_j) \leq rank(\mathbf{x}_i, l_k), l_j \in \mathbf{Y}_i\}$. $\mathbf{Y}'_i$ denotes the labels not belonging to $\mathbf{x}_i$. The lower the value, the better the performance.

$$Ranking\ loss = \frac{1}{n} \sum_{1}^{t} \frac{1}{|\mathbf{Y}_i||\mathbf{Y}'_i|} \frac{(y_{ik}, y_{ij}) | f_k(\mathbf{x}_i)) \leq f_j(\mathbf{x}_i), (y_k, y_j) \in (\mathbf{Y}_i \times \mathbf{Y}'_i)}{rank(\mathbf{x}_i, l_k)} \tag{11}$$

  - *Statistical significance test* We have conducted Wilcoxon signed rank test to evaluate the difference in the methods' performance statistically. We have conducted the tests for a pair of methods—(NaNUML-50% or NaNUML-100% or *Best of*

two) and each competing method on the results obtained from all four evaluating metrics. We have made the evaluations at $p = 0.05$ significance level.

6 Results and discussion

We have randomly partitioned each dataset into two equal (or nearly equal), mutually exclusive halves to construct a training set and a test set for a single run. For each run, we have obtained the results on three metrics. The values in the table are the mean scores obtained from ten experiment runs. The scores obtained on *macro-averaging* F_1, *macro-averaging AUC*, *average precision* and *ranking loss* are shown in Tables 2, 3, 4, and 5 respectively. NaNUML (NaNUML-50% and NaNUML-100%) has obtained the best scores on *macro-averaging* F_1 in 9 out of 12 datasets. Of the nine best scores obtained, NaNUML-50% obtains four, and NaNUML-100% obtains five. COCOA (two) and RML (one) obtain the remaining three best performances. This feat by NaNUML indicates its appropriateness in handling class-imbalance problems in a multi-label context. The performance of NaNUML on *macro-averaging AUC* is a bit subdued as compared to that of *macro-averaging* F_1. NaNUML has obtained the best scores in 6 out of 12 datasets only. The remaining best scores are shared by COCOA (3 out of 6), CLR (2 out of 6), and ECC (3 out of 6). Between NaNUML-50% and NaNUML-100%, the latter has attained a relatively better performance. NaNUML has attained the best scores on *average precision* in 7 out of 12 datasets. We may also note that NANUML-50% achieves six out of those cases, and only one is achieved by NaNUML-100%. ECC has attained the remaining five best scores. The probable reason regarding the loss of performance by NaNUML-100% is due to the deletion of some majority instances, which leads to the loss of some pertinent information. On *ranking loss*, NaNUML has the lowest loss values in 7 out of 12 cases. Out of these, NaNUML-50% and NANUML-100% have achieved 4 and 3, respectively. ECC and CLR have achieved four and one of the best scores, respectively.

We report the statistical significance of the improvement achieved by NaNUML. We have presented the results of the statistical significance test in Table 6. On *macro-averaging* F_1, the performance of NaNUML (best of NaNUML-50% and NaNUML-100%) is better and statistically superior to all competing methods. Concerning *macro-averaging AUC*, NaNUML has delivered a statistically significant improvement against three competing methods and has failed to do so against three. The three methods are COCOA, CLR, and ECC. This finding is in congruence with the data presented in Table 3. On *average precision* and *ranking loss*, NaNUML has obtained statistically superior performance against four competing methods, and NaNUML's performance is statistically comparable to that of COCOA and ECC. We should also note that, only in *one* case, NaNUML-100% has achieved a statistically inferior performance (against COCOA, on *average precision*). The above-summarized results ascertain the appropriateness of the proposed method, NaNUML, over existing schemes dedicated to multi-label learning and class-imbalance mitigation. It is to be noted that, being an undersampling scheme, NaNUML reduces the complexity associated with the classifier modeling.

Table 2 Macro F_1 results

Datasets	NaNUML-50%	NaNUML-100%	COCOA	THRESHL	IRUS	RML	CLR	ECC	RAKEL
CAL500	0.152	**0.292***	0.210	0.252	0.277	0.209	0.081	0.092	0.193
Emotions	0.646	**0.669***	0.660	0.562	0.622	0.645	0.595	0.638	0.613
Scene	0.692	0.649	**0.729**	0.627	0.632	0.682	0.630	0.715	0.687
Yeast	0.388	**0.503***	0.462	0.427	0.428	0.471	0.414	0.392	0.421
Image	0.613	0.618	**0.640**	0.526	0.573	0.512	0.546	0.616	0.613
RCV1	0.171	0.364	0.364	0.294	0.262	**0.385**	0.228	0.192	0.227
RCV2	0.161	**0.393***	0.339	0.273	0.226	0.370	0.212	0.164	0.229
Medical	**0.780***	0.671	0.759	0.733	0.537	0.707	0.724	0.733	0.672
Llog	0.200	**0.207***	0.085	0.095	0.124	0.095	0.024	0.024	0.022
Enron	**0.368***	0.345	0.341	0.292	0.293	0.308	0.244	0.268	0.267
Slashdot	**0.442***	0.382	0.372	0.335	0.258	0.342	0.288	0.305	0.296
Corel5k	**0.217***	0.185	0.196	0.144	0.106	0.215	0.48	0.054	0.083

The bold values indicate the best scores obtained for the datasets on the given metric
Higher the score ↑, better is the performance
*indicates that the performance of the proposed method (NaNUML-50% or NaNUML-100%) is better than all the comparing methods

Table 3 Macro AUC results

Datasets	NaNUML-50%	NaNUML-100%	COCOA	THRESHL	IRUS	CLR	ECC	RAKEL
CAL500	0.532	0.528	0.558	0.509	0.545	**0.561**	0.554	0.528
Emotions	0.819	0.824	0.844	0.687	0.804	0.797	**0.851**	0.797
Scene	0.909	0.902	**0.942**	0.759	0.922	0.894	0.941	0.893
Yeast	0.648	0.666	**0.712**	0.574	0.653	0.651	0.704	0.652
Image	0.817	0.822	0.864	0.681	0.824	0.798	**0.865**	0.812
RCV1	0.898*	**0.920***	0.889	0.642	0.881	0.882	0.876	0.742
RCV2	0.902*	**0.917***	0.881	0.643	0.880	0.880	0.874	0.701
Medical	**0.970***	0.967*	0.964	0.870	0.963	0.954	0.952	0.860
Enron	0.736	0.748	**0.752**	0.597	0.737	0.722	0.750	0.654
Llog	0.730*	**0.741***	0.664	0.518	0.678	0.613	0.674	0.519
Slashdot	**0.817***	**0.817***	0.774	0.635	0.752	0.740	0.765	0.632
Corel5k	0.698	0.706	0.716	0.597	0.684	**0.741**	0.722	0.553

The bold values indicate the best scores obtained for the datasets on the given metric
Higher the score ↑, better is the performance
*indicates that the performance of the proposed method (NaNUML-50% or NaNUML-100%) is better than all the comparing methods

Table 4 Average precision results

Datasets	NaNUML-50%	NaNUML-100%	COCOA	THRESHL	IRUS	CLR	ECC	RAKEL
CAL500	**0.512***	0.377	0.478	0.333	0.276	0.506	0.511	0.401
Emotions	0.788	0.806	0.801	0.683	0.756	0.767	**0.809**	0.766
Scene	0.839	0.830	0.865	0.707	0.844	0.809	**0.871**	0.822
Yeast	**0.767***	0.729	0.762	0.596	0.543	0.742	0.766	0.717
Image	0.793	0.782	0.819	0.671	0.780	0.766	**0.821**	0.775
RCV1	**0.630***	0.628	0.601	0.427	0.556	0.596	0.626	0.501
RCV2	**0.678***	0.676*	0.612	0.457	0.569	0.611	0.632	0.516
Medical	**0.934***	0.926*	0.922	0.870	0.882	0.913	0.920	0.829
Enron	0.669	0.606	0.712	0.595	0.532	0.704	**0.717**	0.654
Llog	**0.618***	0.612*	0.346	0.306	0.308	0.342	0.353	0.218
Slashdot	0.672*	**0.680***	0.605	0.565	0.507	0.593	0.598	0.486
Corel5k	0.396	0.360	0.396	0.343	0.190	0.387	**0.406**	0.213

The bold values indicate the best scores obtained for the datasets on the given metric
Higher the score ↑, better is the performance
*indicates that the performance of the proposed method (NaNUML-50% or NaNUML-100%) is better than all the comparing methods

Table 5 Ranking loss results

Datasets	NaNUML-50%	NaNUML-100%	COCOA	THRESHL	IRUS	CLR	ECC	RAKEL
CAL500	**0.233***	0.332	0.265	0.383	0.482	0.241	0.237	0.340
Emotions	0.159	0.162	0.159	0.306	0.202	0.193	**0.151**	0.200
Scene	0.085	0.096	0.073	0.248	0.089	0.111	**0.073**	0.112
Yeast	**0.180***	0.233	0.186	0.348	0.439	0.204	0.182	0.230
Image	0.168	0.182	0.149	0.312	0.182	0.199	**0.147**	0.198
RCV1	0.073*	**0.062***	0.078	0.287	0.104	0.077	0.074	0.187
RCV2	0.068*	**0.061***	0.081	0.269	0.108	0.079	0.079	0.194
Medical	0.018*	**0.016**	0.023	0.052	0.030	0.027	0.022	0.087
Enron	0.121	0.168	0.116	0.230	0.250	0.121	**0.112**	0.200
Llog	**0.173***	0.177*	0.221	0.265	0.258	0.228	0.223	0.356
Slashdot	**0.138***	0.140*	0.189	0.217	0.246	0.183	0.186	0.330
Corel5k	0.200	0.207	0.201	0.256	0.362	**0.186**	0.189	0.570

The bold values indicate the best scores obtained for the datasets on the given metric
Lower the score ↓, better is the performance
*indicates that the performance of the proposed method (NaNUML-50% or NaNUML-100%) is better than all the comparing methods

Table 6 Results of Wilcoxon signed rank test (two-tailed) at $p = 0.05$

Methods	NaNUML-50%	NaNUML-100%	COCOA	THRESHL	IRUS	RML	CLR	ECC	RAKEL
Macro-averaging F_1									
Best	–	–	↑	↑	↑	↑	↑	↑	↑
NaNUML-50%	–	–	–	–	–	–	–	–	–
NaNUML-100%	–	–	–	↑	↑	–	–	↑	↑
Macro-averaging AUC									
Best	–	–	–	↑	↑	*	–	–	↑
NaNUML-50%	–	↓	–	↑	–	*	–	–	↑
NaNUML-100%	↑	–	–	↑	↑	*	–	–	↑
Averaging precision									
Best	–	–	–	↑	↑	*	↑	–	↑
NaNUML-50%	–	–	–	↑	↑	*	↑	–	↑
NaNUML-100%	–	–	↓	↑	↑	*	–	–	↑
Ranking loss									
Best	–	–	–	↑	↑	*	↑	–	↑
NaNUML-50%	–	–	–	↑	↑	*	↑	–	↑
NaNUML-100%	–	–	–	↑	↑	*	–	–	↑

↑ in (i, j)th cell signifies that the performance of the method in ith row is better and statistically significant than that of the method present in jth column. ↓ in (i, j)th cell signifies that the performance of the method in ith row is poor and statistically significant than that of the method present in jth column. – in (i, j)th cell signifies that there is no statistical significance in the difference in the performance of the method in ith row and the method in jth column. * indicates evaluation was not performed

6.1 The role of α

The degree of undersampling performed on an imbalanced label is quantified and controlled through α. It is varied between 0 and 1 where α=0 signifies no undersampling and $\alpha = 1$ leads to equal cardinalities of the majority and the minority classes. A low value of α results in the prevalence of imbalance, whereas a too-high value can lead to the distortion of the majority class. Hence, it is a critical parameter. We vary α across the following range—0.25, 0.5,0.75, and 1, correlating with the variation of multi-label performances. We explore six datasets—three with numeric features (CAL500, Yeast, and Scene) and three with nominal features (Medical, Llog, and Enron). The plots are shown in Fig. 1.

On macro-averaging F_1, the best scores are rendered at $\alpha = 0.5$ on three cases, $\alpha = 1$ at two case and $\alpha = 0.75$ in one case. At $\alpha = 0.25$, quite low macro-averaging F_1 scores are obtained. The results indicate that a medium to higher α range is effective in improving the recognition of the minority class of each label. On ranking loss, the best (lowest) scores are served at $\alpha = 0.0.5$ in three cases, $\alpha = 0.75$ in two cases, and $\alpha = 0.25$ in one case. At $\alpha = 1$, ranking loss is pretty high.

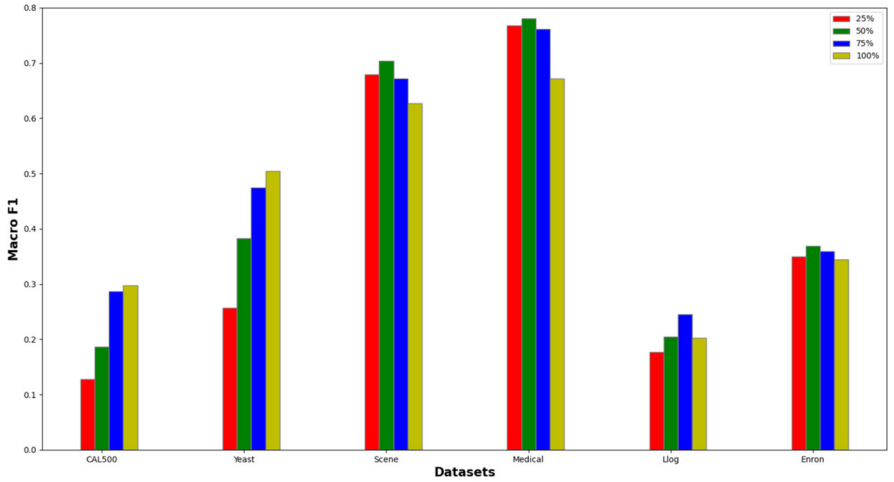
A cumulative analysis in the context of the two metrics reveals that a low α can be detrimental to recognizing the minority class instances. On the contrary, a high value of α leads to a distortion of the majority case, thereby increasing the ranking loss. Given these two findings, fixing α between 0.5 and 0.75 is judicious.

6.2 Exploration of classifiers

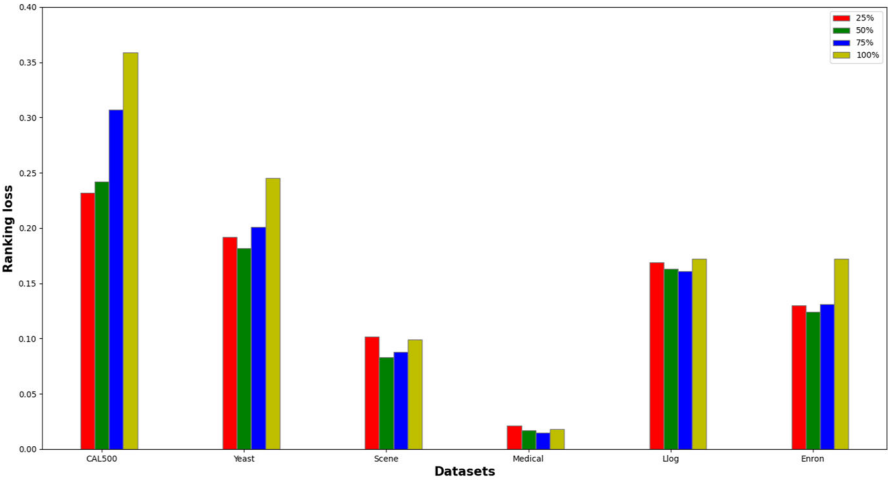
NaNUML is a data pre-processing method where we reduce the skewness in the representation of the majority and the minority classes. To assess its intrinsic efficacy, learning it using different classifiers is essential. We explore five classifiers—k-nearest neighbors classifiers at $k = 5$, Decision Tree classifier, Support Vector Machine classifier with linear kernel and regularization parameter = 1, Random Forest Classifier with depth level = 5 and number of estimators = 10, and AdaBoost classifier (with base classifier Decision Tree at depth level = 1) in this study. The same six datasets and two metrics used in the previous experiment are employed in this study. The outcomes are shown in Fig. 2. For macro-averaging F_1, the Adaboost classifier has performed best. On ranking loss, the SVC has rendered the lowest loss values in most cases. Decision Tree classifier could not fare well on both metrics. Random Forest Classifier has delivered admissible performance on numeric datasets.

7 Conclusion

In this work, we have presented a novel label-specific undersampling scheme, NaNUML, for multi-label datasets. NaNUML is based on the parameter-free natural neighbor search, and the critical factor, neighborhood size 'k', is determined without invoking any parameter optimization. In our scheme, we eliminate the majority instances closer to the minority class. In addition, we preserve the critical lattices of the



(a) Variation of *macro-averaging* F_1 across different α. The higher the score, the better the performance.

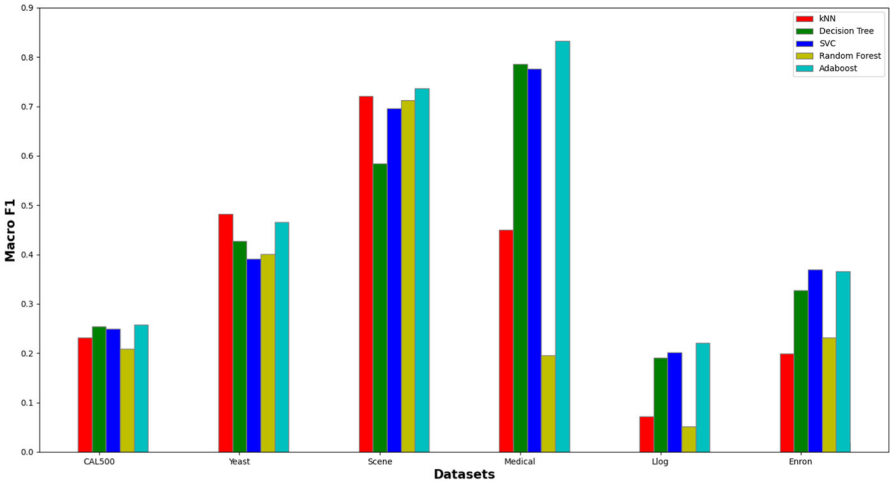


(b) Variation of *ranking loss* across different α. The lower the score, the better the performance.

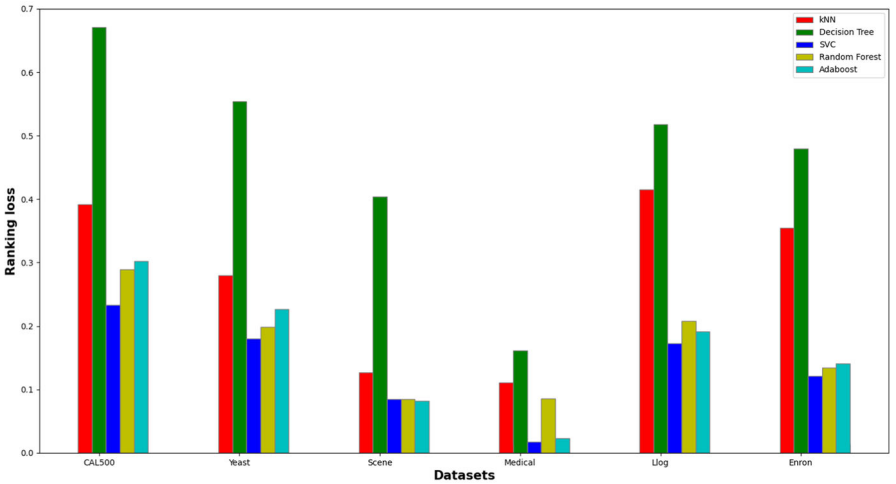
Fig. 1 Exploration of α on six datasets. SVC is used as the base classifier

majority class by looking at the majority natural neighbor count of the majority class. The other advantage of the scheme is that we require only one natural neighbor search for all labels. Undersampling schema has the intrinsic characteristic of reducing the complexity in the classifier modeling phase (through the reduction in training data), and NaNUML is no exception. The performance of NaNUML indicates its ability to mitigate the class-imbalance issue in multi-label datasets to a considerable extent.

In our future work, we would like to design a natural-neighborhood-based oversampling scheme for class-imbalanced datasets. We would also like to explore if we can incorporate label correlations in our undersampling scheme.



(a) *Macro-averaging* F_1 rendered by different classifiers. The higher the score, the better the performance.



(b) *Ranking loss* rendered by different classifiers. The lower the score, the better the performance.

Fig. 2 Exploration of different classifiers on six, NaNUML-undersampled dataset. α is fixed at 0.5 for all cases

References

Ali H, Salleh MNM, Hussain K, Ahmad A, Ullah A, Muhammad A, Naseem R, Khan M (2019) A review on data preprocessing methods for class imbalance problem. Int J Eng Technol 8:390–397

Charte F, Rivera AJ, del Jesus MJ, Herrera F (2015) MLSMOTE: approaching imbalanced multilabel learning through synthetic instance generation. Knowl Based Syst 89:385–397

Charte F, Rivera AJ, del Jesus MJ, Herrera F (2015) MLSMOTE: approaching imbalanced multilabel learning through synthetic instance generation. Knowl Based Syst 89:385–397

Charte F, Rivera AJ, del Jesus MJ, Herrera F (2014) Mlenn: a first approach to heuristic multilabel undersampling. In: International conference on intelligent data engineering and automated learning. Springer, pp 1–9

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Int Res 16(1):321–357

Choirunnisa S, Lianto J (2018) Hybrid method of undersampling and oversampling for handling imbalanced data. In: International seminar on research of information technology and intelligent systems (ISRITI). IEEE, pp 276–280

Daniels Z, Metaxas D (2017) Addressing imbalance in multi-label classification using structured hellinger forests. In: Proceedings of the AAAI conference on artificial intelligence, vol 31

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. NIPS'01, MIT Press, Cambridge, MA, USA, pp 681–687

Fürnkranz J, Hüllermeier E, Loza Mencía E, Brinker K (2008) Multilabel classification via calibrated label ranking. Mach Learn 73(2):133–153

Godbole S, Sarawagi S (2004) Discriminative methods for multi-labeled classification. In: Proceedings of the 8th Pacific-Asia conference on knowledge discovery and data mining, pp 22–30

Gonzalez-Lopez J, Ventura S, Cano A (2018) Distributed nearest neighbor classification for large-scale multi-label data on spark. Future Gener Comput Syst 87:66–82

He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284

Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: European conference on machine learning. Springer, pp 137–142

Katakis I, Tsoumakas G, Vlahavas I (2008) Multilabel text classification for automated tag suggestion. In: Proceedings of the ECML/PKDD-08 workshop on discovery challenge

Liu B, Tsoumakas G (2020) Dealing with class imbalance in classifier chains via random undersampling. Knowl Based Syst 192:105292

Liu Y, Wen K, Gao Q, Gao X, Nie F (2018) SVM based multi-label learning with missing labels for image annotation. Pattern Recognit 78:307–317

Li X, Zhao F, Guo Y (2014) Multi-label image classification with a probabilistic label enhancement model. In: Uncertainty in artificial intelligence

Ludera DT (2021) Credit card fraud detection by combining synthetic minority oversampling and edited nearest neighbours. In: Future of information and communication conference. Springer, pp 735–743

Moyano JM, Gibaja EL, Cios KJ, Ventura S (2018) Review of ensembles of multi-label classifiers: models, experimental study and prospects. Inf Fusion 44:33–45

Nam J, Kim J, Mencía EL, Gurevych I, Fürnkranz J (2014) Large-scale multi-label text classification—revisiting neural networks. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 437–452

Pereira RM, Costa YM, Silla CN Jr (2020) MLTL: a multi-label approach for the Tomek Link undersampling algorithm. Neurocomputing 383:95–105

Pereira RM, Costa YM, Silla CN Jr (2020) MLTL: a multi-label approach for the Tomek Link undersampling algorithm. Neurocomputing 383:95–105

Pillai I, Fumera G, Roli F (2013) Threshold optimisation for multi-label classifiers. Pattern Recognit 46(7):2055–2065

Read J, Pfahringer B, Holmes G, Frank E (2011) Classifier chains for multi-label classification. Mach Learn 85(3):333

Sadhukhan P, Palit S (2019) Lattice and imbalance informed multi-label learning. IEEE Access 8:7394–7407

Sadhukhan P, Palit S (2020) Multi-label learning on principles of reverse k-nearest neighbourhood. Expert Syst 38:e12615

Siblini W, Kuntz P, Meyer F (2018) Craftml, an efficient clustering-based random forest for extreme multi-label learning. In: International conference on machine learning. PMLR, pp 4664–4673

Tahir MA, Kittler J, Yan F (2012) Inverse random under sampling for class imbalance problem and its application to multi-label classification. Pattern Recognit 45(10):3738–3750

Tsoumakas G, Katakis I, Vlahavas I (2011) Random k-labelsets for multilabel classification. IEEE Trans Knowl Data Eng 23(7):1079–1089

Zhang ML, Wu L (2015) Lift: multi-label learning with label-specific features. IEEE Trans Pattern Anal Mach Intell 37(1):107–120

Zhang ML, Li YK, Yang H, Liu XY (2020) Towards class-imbalance aware multi-label learning. IEEE Trans Cybern 52:4459–4471
Zhu Q, Feng J, Huang J (2016) Natural neighbor: a self-adaptive neighborhood method without parameter k. Pattern Recognit Lett 80:30–36