Hindawi
Scientific Programming
Volume 2020, Article ID 8829432, 10 pages
https://doi.org/10.1155/2020/8829432

Research Article

A Decoupling and Bidirectional Resampling Method for Multilabel Classification of Imbalanced Data with Label Concurrence

Shuyue Zhou,[1] **Xiaobo Li**,[2,3] **Yihong Dong**,[1] **and Hao Xu**[3]

[1]*Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China*
[2]*College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321004, China*
[3]*College of Engineering, Lishui University, Lishui 323000, China*

Correspondence should be addressed to Xiaobo Li; oboaixil@126.com

Received 17 March 2020; Revised 25 June 2020; Accepted 29 June 2020; Published 1 August 2020

Academic Editor: Iván García-Magariño

Label imbalance is one of the characteristics of multilabel data, and imbalanced data seriously affects the performance of the classifiers. In multilabel classification, resampling methods are mostly used to deal with imbalanced problems. Existing resampling methods balance the data by either undersampling or oversampling, which causes overfitting and information loss. Resampling has a significant impact on the minority labels. Furthermore, the high concurrency of majority labels and minority labels in many instances also affects the performance of classification. In this study, we proposed a bidirectional resampling method to decouple multilabel datasets. On one hand, the concurrency of labels can be reduced by setting termination conditions for decoupling, and on the other hand, the loss of instance information and overfitting can be alleviated by combining oversampling and undersampling. By measuring the minority labels of the instances, the instances that have less impact on minority labels are selected to resample. The number of resampling is limited to keep the original distribution of the data during the resampling phase. The experiments on seven benchmark multilabel datasets have proved the effectiveness of the algorithm, especially on datasets with high concurrency of majority labels and minority labels.

1. Introduction

With the advent of the era of big data, data classification has received much attention in recent years. The imbalanced data often occurs in the field of data classification, including medical data. Data imbalance means that some categories of instances are much higher or lower than others. Generally, compared with the balanced datasets, most algorithms perform poorly when dealing with imbalanced data. The performance of the classifiers is biased to the majority class, and a higher error rate will occur on the minority class. In practical application, we tend to pay more attention to the correct classification results of the minority classes; as a result, it is more important to correctly identify minority classes than to correctly identify majority classes. For example, in the field of tumor classification, nontumor patients are the majority class, while tumor patients are the minority class [1], but we are more concerned about the minority of tumor patients. These problems also exist in the fields of medical imaging classification, credit card fraud [2] detection, and network intrusion identification, etc.

The multilabel imbalance problem is different from the traditional imbalance problem. In multilabel imbalance problem, each instance is associated with a set of labels, instead of only one type of label as in binary classification. The class with a larger number of instances is called the majority class, which corresponds to the majority label, and the class with a smaller number of instances is called the minority class, which corresponds to the minority label [3–5]. For example, in the prediction of drug targets, since each drug molecule can correspond to multiple targets, targets can also correspond to multiple drug molecules, but

some targets contain far less instances than the rest of the targets, which greatly increases the difficulty of classification.

In traditional binary classification, because each sample only corresponds to one class, it is not necessary to consider the influence of different classes. But the multilabel classification has to face a new challenge, in which some instances contain both majority labels and minority labels. These two kinds of labels are highly concurrent, which makes it more difficult to correctly classify multilabel data. The multilabel data imbalance and the concurrency of majority labels and minority labels often coexist. When dealing with these two issues at the same time, they often need to be considered together.

2. Related Work

The imbalanced data is divided into the single-label imbalanced data and the multilabel imbalanced data according to the number of instance labels. This section will introduce resampling methods of traditional single-label imbalanced data and multilabel imbalanced data and describe their advantages and disadvantages in detail.

In traditional single-label imbalanced data processing methods, relevant studies can be divided into three aspects: algorithm-level methods, cost-sensitive learning methods [6–8], and data-level methods. Among algorithm-level methods, the classification algorithm is improved to adapt to imbalanced data sets. The improved algorithm usually moves the decision boundary to enhance the existence of minority label instances. Hong et al. [9] optimized the distribution of imbalanced datasets by improving the kernel classifier. Liu et al. [10] applied weighted Gini index (WGI) to choose a subset of features, which is conducive to the accurate determination of the minority class. Cost-sensitive learning achieves the goal of correct classification by penalizing misclassifications. The data-level methods primarily focus on resampling methods, including undersampling [11], oversampling [12], and SMOTE method (Synthetic Minority Oversampling Technique). The data was balanced by deleting the instances of majority class in the dataset or increasing instances of minority class. Galar et al. [13] compared common imbalanced learning algorithms and proved that data preprocessing combined with other classification methods is an effective imbalanced classification method. Kang et al. [14] proposed a Noise-filtered Undersampling Scheme (NUS) by incorporating a noise filter before executing resampling process.

Although the processing of multilabel imbalanced data is also based on the data-level methods and the algorithm-level methods, the traditional imbalanced data processing methods are not completely applicable to multilabel imbalanced datasets. Among the previous methods, the algorithm-level methods mainly focus on adjusting the existing classification methods to adapt to the imbalance data [15, 16]. The traditional multilabel [17] classification is to convert multilabel problems into two-class [18, 19] or multiclass problems [20, 21], such as Label Powerset (LP) [22] and Binary Relevance (BR) [4]. Zhang et al. [23] improved the traditional classification algorithm and proposed the COCOA algorithm, which converts the original multilabel dataset to one binary dataset and several multiclass datasets for each label and resamples each dataset to achieve the purpose of constructing the imbalanced classifiers.

The data-level methods change the distribution of the instances to achieve the balance of the dataset. The methods mainly focus on resampling, including oversampling to generate new instances from minority class and undersampling methods to remove some instances from majority labels. Among the multilabel imbalanced data processing methods, the method based on the data-level should be paid more attention, because this method has the following advantages: (1) it is independent of the classification process and can be applied without disturbing the classification algorithm; (2) the separation of tasks allows different algorithms to exert their advantages. Hence, some researchers have conducted relevant research on this aspect. In 2015, Dendamrongvit and Kubat proposed the data-level LP-RUS (LP-based Random Undersampling) and LP-ROS (LP-based Random Oversampling) [24] algorithms and their improved algorithms ML-RUS (Multilabel Random Undersampling) and ML-ROS (Multilabel Random Oversampling) [25].

Both LP-RUS and LP-ROS methods decide how to resample by considering the label set of the dataset. LP-RUS deletes the instances that appear in the most frequent label set, while LP-ROS clones the instances of the least frequent label set. In the resampling process, LP-RUS and LP-ROS may lead to new imbalance of some labels. In order to balance the dataset, the ML-ROS algorithm randomly copies instances related to minority labels to increase the frequency of occurrence of minority labels in the dataset, while ML-RUS randomly deletes the number of instances with majority labels to reduce the frequency of majority labels in the instance set.

ML-ROS and ML-RUS resample the dataset, which improves the classification performance. However, there are some disadvantages: (1) using oversampling or undersampling alone results in the redundancy of a minority labels information and the loss of majority labels information; (2) these methods destroy the original distribution of datasets and cause adverse effects on the classification [26]; (3) they cannot balance the highly concurrent instances of the majority labels and the minority labels.

In order to alleviate the problems where minority labels and majority labels being highly concurrent, Charte et al. [27] proposed the REMEDIAL, REMEDIAL-HwR-ROS (REMEDIAL Hybridizing with Random Oversampling), and REMEDIAL-HwR-HUS (REMEDIAL Hybridizing with Heuristic Undersampling). The REMEDIAL algorithm is independent of the resampling algorithms and can be combined with various resampling algorithms to decouple majority and minority labels, reducing the degree of concurrency among labels [28]. The REMEDIAL-HwR-ROS decouples the highly concurrent labels, then looks for instances linked to minority labels, and generates clones from them. REMEDIAL-HwR-HUS decouples the highly concurrent labels and applies the undersampling processing. But there are several problems in these algorithms: (1) the algorithms do not fundamentally change their original

disadvantages and may still cause serious overfitting or loss of information; (2) the algorithms divide the datasets into two parts. Even though the high concurrency problem has been solved during the decoupling process, decoupling will continue, and overfitting may occur during the classification process.

3. Our Approach

In this section, we proposed a Multilabel Decoupling Bidirectional Resampling algorithm (ML-DBR).

3.1. Related Definitions. In the study of the imbalance problem of multilabel data, to measure the degree of data imbalance, there are two measurement indicators to distinguish different labels in multilabel imbalance data: Imbalance Ratio per Label (IR) and Mean Imbalance Ratio (MeanIR).

Let a multilabel dataset $D = \{(X_i, L_i) \mid 0 \le i \le n, L_i \in Y\}$, where X_i represents the i-th instance of the dataset, L_i is the label set of X_i, and Y is the label set of the dataset.

$$\mathrm{IR}(y) = \frac{\arg\max_{y=Y_1}^{Y_{|Y|}} \left(\sum_{i=1}^{|D|} h(y, L_i)\right)}{\sum_{i=1}^{|D||} h(y, L_i)},$$
$$h(y, L_i) = \begin{cases} 1, & y \in L_i, \\ 0, & y \notin L_i. \end{cases} \tag{1}$$

3.2. MeanIR. MeanIR represents the average level of imbalance in the dataset, as shown in equation (2). MeanIR is the mean of all labels IR:

$$\mathrm{MeanIR} = \frac{1}{|Y|} \sum_{y=Y_1}^{Y_{|Y|}} (\mathrm{IR}(y)). \tag{2}$$

According to MeanIR and IR, we can define majority and minority labels. If the IR value of a label is higher than MeanIR, it is a minority label; otherwise, it is a majority label. For label y, if $\mathrm{IR}(y) > \mathrm{MeanIR}$, it belongs to minBag; otherwise, it belongs to majBag.

3.3. SCUMBLE [29]. Besides, we use SCUMBLE metric to assess the degree of concurrency between majority label and minority label, and their values are in the (0, 1) range. The higher the value, the more the instances containing minority and majority labels existing in the dataset:

$$\mathrm{SCUMBLE}(D) = \frac{1}{n} \sum_{i=1}^{n} (\mathrm{SCUMBLEIns}(i)), \tag{3}$$

$$\mathrm{SCUMBLEIns}(i) = 1 - \frac{1}{\overline{\mathrm{IR}}_i} \left(\prod_{y \in Li} \mathrm{IR}_i(y) \right)^{1/k}. \tag{4}$$

In equation (3), n is the number of instances in the dataset, and in equation (4), k is the number of labels of X_i and IR_i is the IR set of L_i.

3.4. Min-SCUMBLE. Relevant studies [27] have shown that the resampling of minority label instances has the most impact on other minority labels contained in the instance. When resampling a certain label, the resampling of this label also resamples other minority labels included in the instance, which will interfere with the resampling of other minority labels. Based on the SCUMBLE metric, we propose a Min-SCUMBLE metric that particularly measures for the minority labels in the instance when resampling:

$$\mathrm{Min-SCUMBLEIns}(i) = 1 - \frac{1}{\overline{\mathrm{IR}}_i} \left(\prod_{y \in \mathrm{minBag}} \mathrm{IR}_i(y) \right)^{(1/k)}, \tag{5}$$

where k is the number of minority labels.

3.5. MeanSamples. In addition, MeanSamples is used in the ML-DBR. MeanSamples represents the number of instances required for all labels to reach the balance state of MeanIR. It is calculated by dividing the number of label instances with the highest occurrence frequency by the MeanIR value:

$$\mathrm{Mean\ Samples} = \frac{\arg\max_{y'=Y_1}^{Y_{|Y|}} \left(\sum_{i=1}^{D} h(y', Y_i)\right)}{\mathrm{MeanIR}}. \tag{6}$$

3.6. Proposed Algorithm. The pseudocode of ML-DBR is shown in Algorithm 1. The algorithm is divided into two stages, i.e., decoupling and resampling. In the first stage, the decoupling strategy decouples high concurrency labels and prevents decoupling of instances with low concurrency labels (Steps 4–10 in Algorithm 1). In the second stage, oversampling and undersampling are combined to select instances that have less impact on the minority label for resampling (Steps 11–24 in Algorithm 1).

For each label, IR and MeanIR are calculated to determine which category the label belongs to. Resampling rate P represents the proportion of increase or decrease in the dataset. In ML-ROS and ML-RUS, it causes the dataset to swell or shrink the proportion of P. In ML-DBR, P is not the proportion of the dataset to increase or decrease, but to calculate the number of instances that need to be adjusted. Next, we introduce the strategies used in the ML-DBR.

3.6.1. Decoupling Strategy. ML-DBR calculates the SCUMBLEIns value for each instance in the dataset, sets the initial SCUMBLE(D) of the dataset as $\mathrm{SCUMBLE}(D)_1$, and decouples the instances that meet the requirements according to the $\mathrm{SCUMBLE}(D)_1$, so as to reduce the instances with highly concurrent labels. if $\mathrm{SCUMBLEIns}(i) > \mathrm{SCUMBLE}(D)_1$, clone the instance D_i as D_i', L_i is the label set of D_i, L_i' is the label set of D_i', $L_i' = L_{i[\mathrm{IR}(y) \ge \mathrm{MeanIR}]}$, $L_i = L_{i\,[\mathrm{IR}(y) \le \mathrm{MeanIR}]}$. Then, when every 1% of the instances in the dataset are decoupled, the SCUMBLE(D) of the uncoupled dataset is recalculated. When $\mathrm{SCUMBLE}(D)_{j-1} - \mathrm{SCUMBLE}(D)_j \ge t$, it is considered that the high concurrency of the dataset has been solved, where j means being decoupled to j% instances, and $j-1$ means being decoupled

Algorithm ML-DBR:
Input: A multilabel dataset D, resampling rate P
Output: Preprocessed dataset D
Decoupling strategy
(1) Calculate samplesToResampling = $|D|$*P, IR, Mean IR & Mean Samples
(2) Calculate SCUMBLEIns in D and set the SCUMBLE(D) as SCUMBLE$(D)_1$
(3) For D_i in D,
(4) If SCUMBLEIns(i) > SCUMBLE$(D)_0$, then
(5) clone $D_i \longrightarrow D_i', L_i' = L_{i[\mathrm{IR}(y)\geq \mathrm{MeanIR}]}, L_i = L_{i[\mathrm{IR}(y)\leq \mathrm{MeanIR}]}$ (L_i' is the label set of D_i')
(6) $D_d = D_d + D_i + D_i'$, $D = D - D_i$ (D_d is the decoupled dataset)
(7) For j% samples ($j \in (2, 3 \ldots)$)
(8) Recalculate the SCUMBLE(D) as SCUMBLE$(D)_j$
(9) If SCUMBLE$(D)_{j-1}$ −SCUMBLE$(D)_j < t$, then stop decoupling
(10) $D = D + D_d$
Resampling strategy
(11) While samplesToResampling > 0
(12) random select y
(13) if $y \in \min Bag \&\& |y| <$ MeanSamples&&IR$(y) >$ MeanIR then
(14) x = random$(0,$ MeanSamples $- |y|)$
(15) While $x > 0$
(16) random get m samples from y
(17) let the max Min-SCUMBLEIns sample of samples as Z, clone Z
(18) $D = D + Z$, $x - -$, samplesToResampling −−
(19) if $y \in majBag \&\& |y| >$ MeanSamples&&IR$(y) <$ MeanIR then
(20) x = random$(0,$ MeanSamples $- |y|)$
(21) While $x > 0$
(22) random get m samples from y
(23) Let the max Min-SCUMBLEIns sample of m samples as Z, Set $y$of Z to 0
(24) $x - -$, samplesToResampling −−
(25) Recalculate MeanIR, if MeanIR ≤ 1.5, then stop algorithm
(26) return D

Algorithm 1: The pseudocode of ML-DBR.

to $(j-1)$ % instances. If SCUMBLE$(D)_{j-1}$ − SCUMBLE$(D)_j <$ t, continue decoupling the instances of SCUMBLEIns > SCUMBLE$(D)_1$.

Instances with highly concurrent labels can be separated from both minority labels and majority labels by decoupling, and the highly concurrent labels can be found according to Step 4. From Step 4 to Step 6, the instances were decoupled into two instances. Although the characteristics of decoupled instances are the same, the label sets are different.

3.6.2. Resampling Strategy. Firstly, this strategy randomly selects m instances of a certain label y. MeanSamples is used to limit the number of samples, which can balance the distribution between samples and do not exceed or fall below the number of samples required to achieve balance when resampling. Next, generate a random x and randomly picked m instances from y. The Min-SCUMBLEIns metric was used to resample randomly selected instances and compared Min-SCUMBLE of m instances to select an instance with less impact on the minority label for resampling. If y belongs to minBags, x = Random (0, MeanSamples − $|y|$), and clone the instance with the lower Min-SCUMBLE. If y belongs to majBags, x = Random (0, $|y|$ − MeanSamples), and set the label y of the instance with the lower Min-SCUMBLE to 0.

At the end of each resampling, the MeanIR and IR are recalculated, MeanSamples only records the initial value and does not recalculate during the resampling process, so that the original distribution of the dataset will not be affected too much. According to the study, when MeanIR ≤ 1.5, the performance improvement of the classifier by resampling the dataset is limited [25], so when MeanIR ≤ 1.5, the ML-DBR stops.

4. Results and Discussion

4.1. Evaluation Metrics. The performance of general multilabel classifiers can be measured in a variety of ways, which can be divided into multiple types: example-based, label-based, and ranking-based. In order to better evaluate the performance of different methods, we use label-based evaluation method. This method can better reflect the correct classification of majority labels and minority labels. There are two types of label-based evaluation methods: macromeasurement and micromeasurement. Accuracy, macro-F, and micro-F were selected as evaluation indicators [30] for the purpose of obtaining a comprehensive evaluation. For a label, TP represents true positives, TN represents negatives, FP represents false positives, and FN represents false negatives.

Accuracy is the ratio of the number of correctly predicted instances to the total number of predicted instances, regardless of whether the instances are positive or negative. The accuracy is calculated as follows:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{7}$$

Macro-F and micro-F inherit the advantages of F-Measure and can better reflect the classification effect of minority label.

Macro-F refers to the arithmetic mean of each statistical indicator value of all categories. The calculation method of macro-F is shown in equation (10), where equations (8) and (9) are the macro-Precision (macro-P) and the macro-Recall (macro-R). In equations (8) and (9), P and R represent precision and recall:

$$\text{macro\_}P = \frac{1}{n}\sum_{i=1}^{n} P_i, \tag{8}$$

$$\text{macro\_}R = \frac{1}{n}\sum_{i=1}^{n} R_i, \tag{9}$$

$$\text{macro} - F = \frac{2 * \text{macro\_}P * \text{macro\_}R}{\text{macro\_}P + \text{macro\_}R}. \tag{10}$$

Micro-F is to calculate a global confusion matrix for each instance in the dataset regardless of the category. Micro-F is calculated as in equation (11), and (12) and (13) are micro-precision(micro-P) and micro-recall (micro-R):

$$\text{micro\_}P = \frac{\overline{\text{TP}}}{\overline{\text{TP}} + \overline{\text{FP}}} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\sum_{i=1}^{n} \text{TP}_i + \sum_{i=1}^{n} \text{FP}_i}, \tag{11}$$

$$\text{micro\_}R = \frac{\overline{\text{TP}}}{\overline{\text{TP}} + \overline{\text{FN}}} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\sum_{i=1}^{n} \text{TP}_i + \sum_{i=1}^{n} \text{FN}_i}, \tag{12}$$

$$\text{micro} - F = \frac{2 * \text{micro\_}P * \text{micro\_}R}{\text{micro\_}P + \text{micro\_}R}. \tag{13}$$

4.2. Datasets. As shown in Table 1, the seven benchmark multilabel datasets of *yeast*, *enron*, *tmc-2007*, *cal500*, *Corel-16k*, *Corel-5k* and *mediamill* were selected as experimental datasets [31]. The classification performance of multilabels is related not only to the number of labels but also to the different characteristics of the dataset. To measure different characteristics of the datasets, we introduce Dens, Card, and TCS [32] as the measurement of datasets. Dens indicates the density of labels, as shown in equation (14). The higher the value, the denser the labels. Card represents the average number of labels for each instance, as shown in equation (15). The higher the number is, the more the average number of labels per instance is. TCS is used to evaluate the complexity of a dataset, as shown in equation (16). A higher value indicates that the dataset is more complex, and it is more difficult for the classifier to predict the correct classification result:

$$\text{Dens}(D) = \frac{1}{k}\frac{1}{n}\sum_{i=1}^{n} |Yi|, \tag{14}$$

$$\text{Card}(D) = \frac{1}{n}\sum_{i=1}^{n} |Yi|, \tag{15}$$

$$\text{TCS}(D) = \log(f * k * ls), \tag{16}$$

where n is the number of instances, f is the number of input features, k is the number of labels, and ls is the number of different label sets.

4.3. Optimal Values of t and m. The parameters t and m in our method (algorithm ML-DBR) directly affect the performance of the algorithm, so it is also important to explore the appropriate values of t and m. t is the threshold value of decoupling. When t is high, some instances are not decoupled. If t is lower, the instances continue decoupling when label concurrency is balanced and t should lower than the SCUMBLE value of different datasets. The lowest SCUMBLE value is 0.1 on the different dataset, so $t \leq 0.1$. m is the number of instances extracted during each resampling. When m is high, it is possible to increase the frequency of some instances. When m is low, instances that have a greater impact on minority labels may be selected. In addition, m needs to be less than the minimum number of instances for minority labels. In the ML-DBR, t is set to 0.1, 0.01, and 0.001, m is set to 3, 5, and 7 for comparison. In the traditional multilabel classification, it is the most common way to convert multilabel into binary classification problem, such as LP and BR. In this paper, LP, BR, and ML-kNN [33] were selected for classification, and C4.5 was used as the underlying classifier in BR and LP. All the parameters in the algorithm were chosen as default parameters, the resampling rate P in the experiment was set to 0.1, which is the best resampling rate for ML-ROS, and the number of neighbors for ML-kNN was set to 10. Ten-fold cross-validation was used in this experiment. *Yeast*, *enron*, *tmc-2007*, *cal500*, and *Corel16k* were selected as experimental datasets.

The experimental results of m and t values are shown in Figures 1–5. When $m = 3$, it performs better on different datasets than $m = 5$ and $m = 7$. In the measurement of micro-F, the performance of $m = 3$ far exceeds that of the other two values. The main reason is that when m is 5 and 7, it increases the frequency of a part of instances with high MinSCUMBLEIns, and the overfitting is more serious than $m = 3$ in the classification. Therefore, $m = 3$ is an appropriate value in the ML-DBR. It is also found in the experiment that the performance at $t = 0.01$ is better than 0.001 and 0.1. The reason is that the threshold is lower when $t = 0.001$, and all instances of SCUMBLEIns > SCUMBLE$(D)_1$ are almost decoupled, and the decoupling cannot be terminated after the dataset is balanced; when $t = 0.1$, the threshold is higher, and the decoupling is terminated when the dataset is not balanced. These figures show that $t = 0.01$ and $m = 3$ obtained the best results for most of the datasets and the combination of ML-DBR and ML-kNN classification algorithms has the

Table 1: Description of seven benchmarking datasets.

Dataset	Instances	Labels	Dens	Card	MeanIR	SCUMBLE	TCS
Yeast	2417	14	0.303	4.24	7.20	0.10	12.56
Enron	1702	53	0.06	3.38	73.95	0.30	17.50
tmc2007	28596	22	0.10	2.16	15.16	0.18	16.37
cal500	502	174	0.15	26.04	20.59	0.34	15.60
Corel-16k	13766	153	0.02	2.86	34.16	0.27	19.72
Corel-5k	5000	161	0.01	3.52	189.57	0.39	20.20
Mediamill	43907	101	0.04	4.38	256.41	0.36	18.19

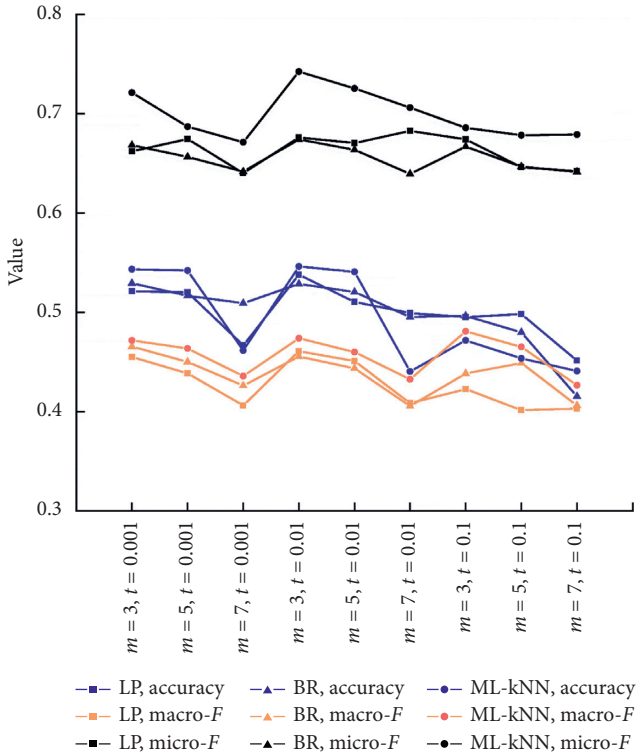

Figure 1: Results for different t and m values in the yeast dataset. Different colors represent different evaluation metrics, and different linetypes represent different algorithms.

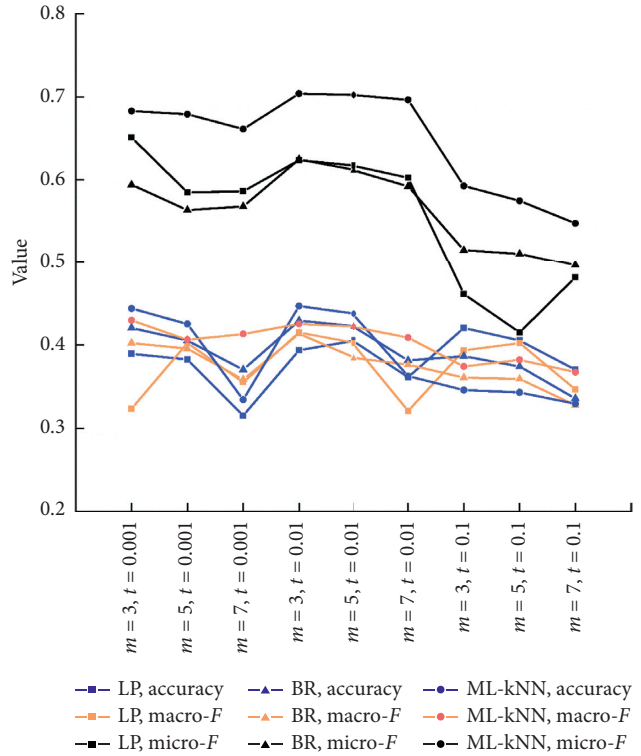

Figure 2: Results for different t and m values in the enron dataset. Different colors represent different evaluation metrics, and different linetypes represent different algorithms.

best effect, and it is better than LP and BR in different measurements, indicating that ML-kNN is more suitable for ML-DBR.

4.4. Experiment and Analysis. The proposed ML-DBR algorithm was compared with three algorithms: REMEDIAL-HwR-HUS, REMEDIAL-HwR-ROS [28], and the combination of REMEDIAL [27] and LP-ROS. The REMEDIAL-HwR-HUS and REMEDIAL-HwR-ROS algorithms have achieved good results in previous experiments, especially in the imbalanced dataset. The LP, BR, and ML-kNN classifiers were used to classify the dataset, and tenfold cross-validation was used. In ML-DBR, the m value was set to 3, and t was set to 0.01. The resampling rate P of all algorithms was 0.1, and all the other parameters were default. 10 experiments were performed on each dataset, and the results were averaged.

Tables 2–4 show the experimental results assessed with the accuracy, macro-F, and micro-F, respectively. The best results are highlighted with bold typeface. As shown in Table 2, compared with other algorithms, ML-DBR achieves the best results. In Tables 3 and 4, ML-DBR also has the best performance in both macro-F and micro-F values. The performance of ML-DBR is far ahead of the other algorithms on the *Corel16k*, *enron*, *Corel-5k*, and *mediamill* datasets, which indicates that our proposed ML-DBR algorithm obtains the best results when SCUMBLE and TCS are higher. In addition, ML-DBR also has certain advantages on datasets with lower SCUMBLE and TCS. ML-DBR achieved the best results on the *tmc-2007* dataset. On the *yeast* dataset, compared with REMEDIAL-HwR-ROS, the ML-DBR did not obtain the best results in some metrics. This happens because the *yeast* dataset has lower SCUMBLE and MeanIR values, and there is no obvious difference between the two algorithms when preprocessing the *yeast* dataset. On the *cal-500* dataset,

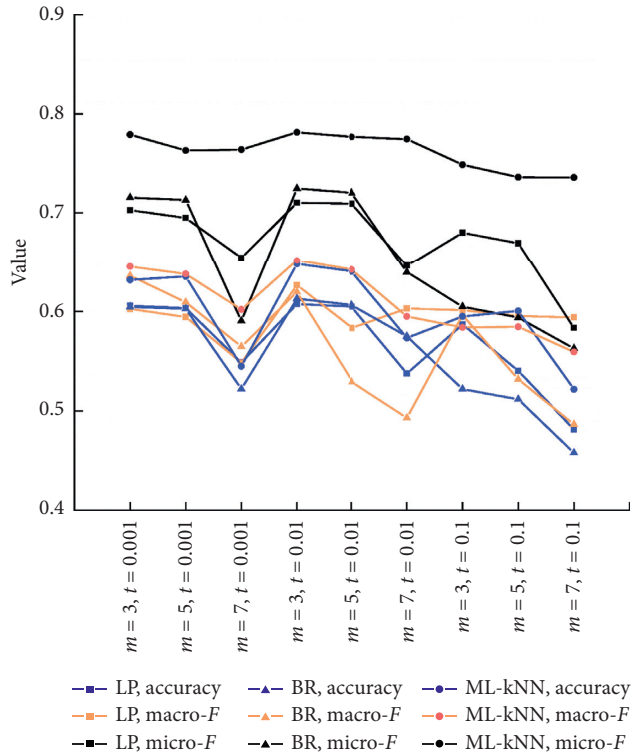

Figure 3: Results for different t and m values in the tmc-2007 dataset. Different colors represent different evaluation metrics, and different linetypes represent different algorithms.

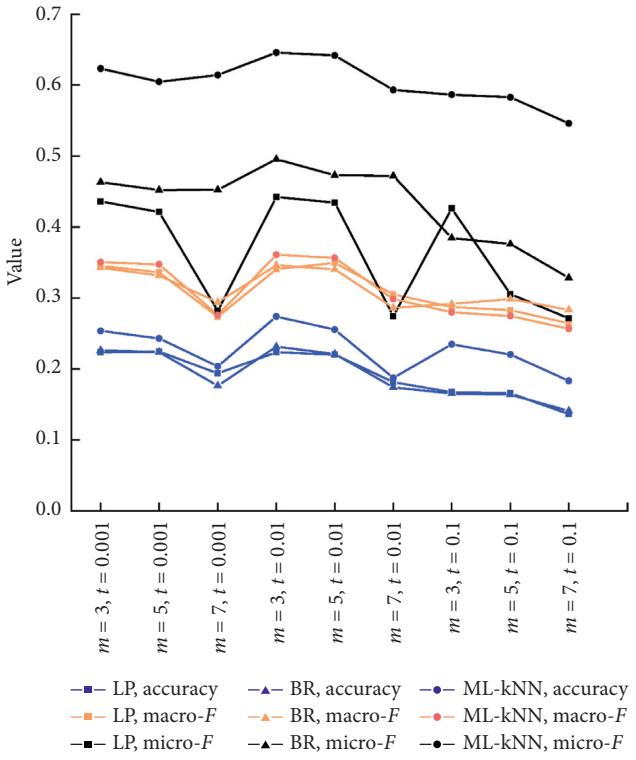

Figure 4: Results for different t and m values in the cal-500 dataset. Different colors represent different evaluation metrics, and different linetypes represent different algorithms.

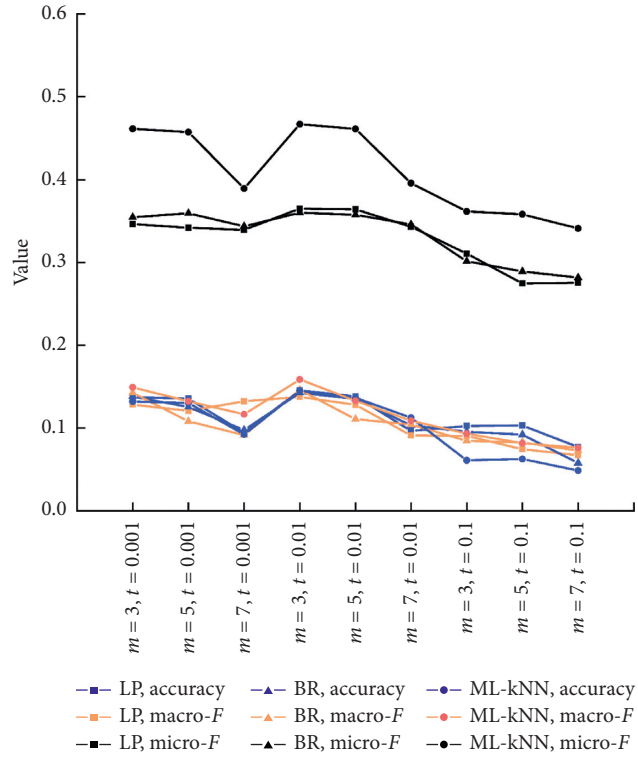

Figure 5: Results for different t and m values in the Corel-16k dataset. Different colors represent different evaluation metrics, and different linetypes represent different algorithms.

Table 2: Accuracy values of different resampling algorithms.

Algorithm	Dataset	LP-ROS	REMEDIAL-HwR-HUS	REMEDIAL-HwR-ROS	ML-DBR
LP	Yeast	0.4038	0.4179	**0.4344**	0.4321
LP	Enron	0.3447	0.3523	0.3605	**0.3936**
LP	tmc-2007	0.5929	0.6027	0.6038	**0.6052**
LP	cal-500	0.2069	0.1851	**0.2291**	0.2284
LP	Corel-16k	0.0951	0.1018	0.1196	**0.1356**
LP	Corel-5k	0.0864	0.1011	0.0924	**0.1028**
LP	Mediamill	0.4039	0.4326	0.4462	**0.4697**
BR	Yeast	0.4027	**0.4306**	0.4191	0.4286
BR	Enron	0.3134	0.3912	0.3903	**0.4291**
BR	tmc-2007	0.4862	0.6012	0.6068	**0.6127**
BR	cal-500	0.1961	0.1849	**0.2336**	0.2315
BR	Corel-16k	0.0958	0.1067	0.1214	**0.1338**
BR	Corel-5k	0.0871	0.0951	0.0894	**0.0972**
BR	Mediamill	0.4018	0.4276	0.4250	**0.4429**
ML-kNN	Yeast	0.4046	0.4219	0.4367	**0.4463**
ML-kNN	Enron	0.3875	0.4286	0.4322	**0.4468**
ML-kNN	tmc-2007	0.5328	0.6340	0.6379	**0.6479**
ML-kNN	cal-500	0.2494	0.2231	**0.2771**	0.2741
ML-kNN	Corel-16k	0.1028	0.1242	0.1266	**0.1453**
ML-kNN	Corel-5k	0.1020	**0.1238**	0.1136	0.1221
ML-kNN	Mediamill	0.4457	0.4813	0.4879	**0.4908**

Table 3: Macro-F values of different resampling algorithms.

Algorithm	Dataset	LP-ROS	REMEDIAL-HwR-HUS	REMEDIAL-HwR-ROS	ML-DBR
LP	Yeast	0.4328	0.4592	**0.4629**	0.4609
LP	Enron	0.3582	0.3863	0.3927	**0.4151**
LP	tmc-2007	0.5174	0.6138	0.6199	**0.6264**
LP	cal-500	0.3041	0.3028	0.3325	**0.3406**
LP	Corel-16k	0.0924	0.1057	0.1129	**0.1377**
LP	Corel-5k	0.1236	0.1429	0.1354	**0.1438**
LP	Mediamill	0.2248	0.2379	0.2406	**0.2428**
BR	Yeast	0.4320	**0.4573**	0.4512	0.4556
BR	Enron	0.3602	0.3667	0.4021	**0.4134**
BR	tmc-2007	0.5228	0.5854	0.6039	**0.6194**
BR	cal-500	0.3112	0.2935	0.3209	**0.3461**
BR	Corel-16k	0.0941	0.1124	0.1136	**0.1461**
BR	Corel-5k	0.1265	**0.1479**	0.1323	0.1468
BR	Mediamill	0.2392	0.2416	0.2473	**0.2605**
ML-kNN	Yeast	0.4292	0.4760	**0.5374**	0.5360
ML-kNN	Enron	0.4338	0.3851	0.4417	**0.4624**
ML-kNN	tmc-2007	0.4864	0.6133	0.6420	**0.6515**
ML-kNN	cal-500	0.3543	0.3691	0.3757	**0.4204**
ML-kNN	Corel-16k	0.1245	0.1263	0.1565	**0.1829**
ML-kNN	Corel-5k	0.2012	0.2205	0.2187	**0.2274**
ML-kNN	Mediamill	0.2572	0.2938	0.2856	**0.3067**

the accuracy of ML-DBR is not significantly improved, but the macro-F and micro-F values of ML-DBR are superior to that of other algorithms, which indicates that the classification accuracy of minority labels has been improved on the *cal-500* dataset. In general, the combination of ML-DBR and ML-kNN classifier achieves the best performance.

Table 5 shows the new SCUMBLE and MeanIR values for each dataset after the ML-DBR was applied. The SCUMLE and MeanIR values were decreased compared with Table 1, which verifies the effectiveness of the decoupling strategy and resampling strategy adopted by the proposed ML-DBR algorithm.

In summary, compared with other resampling algorithms, our experiments prove that ML-DBR has the best performance among several multilabel resampling algorithms. It can effectively balance multilabel imbalanced data at the data level. ML-DBR can effectively deal with multilabel imbalanced data with high concurrency of minority label and majority label, and it has a significant effect on improving the classification performance of minority labels.

Table 4: Micro-F values of different resampling algorithms.

Algorithm	Dataset	LP-ROS	REMEDIAL-HwR-HUS	REMEDIAL-HwR-ROS	ML-DBR
LP	Yeast	0.5737	**0.6774**	0.6743	0.6763
LP	Enron	0.5581	0.5649	0.6016	**0.6238**
LP	tmc-2007	0.6125	0.7033	0.7072	**0.7103**
LP	cal-500	0.4279	0.4213	0.4316	**0.4424**
LP	Corel-16k	0.2374	0.2867	0.3416	**0.3652**
LP	Corel-5k	0.2229	0.2752	0.2433	**0.2809**
LP	Mediamill	0.5136	0.5625	0.5593	**0.5671**
BR	Yeast	0.5917	0.6667	**0.6753**	0.6742
BR	Enron	0.5784	0.5939	0.6016	**0.6429**
BR	tmc-2007	0.6379	0.7221	0.7230	**0.7248**
BR	cal-500	0.4213	0.4652	0.4679	**0.4956**
BR	Corel-16k	0.2466	0.3455	0.3492	**0.3601**
BR	Corel-5k	0.2185	0.2673	0.2357	**0.2710**
BR	Mediamill	0.5278	0.5562	0.5641	**0.5698**
ML-kNN	Yeast	0.6939	0.6985	0.7011	**0.7213**
ML-kNN	Enron	0.6061	0.5732	0.6108	**0.6805**
ML-kNN	tmc-2007	0.7248	0.7225	0.7354	**0.7826**
ML-kNN	cal-500	0.5849	0.5320	0.6287	**0.6735**
ML-kNN	Corel-16k	0.4266	0.4212	0.4597	**0.4675**
ML-kNN	Corel-5k	0.3901	0.4670	0.4524	**0.4709**
ML-kNN	Mediamill	0.6817	0.7724	0.7782	**0.7815**

Table 5: MeanIR and SCUMBLE values after ML-DBR applied.

Dataset	MeanIR	SCUMBLE
Yeast	2.43	0.07
Enron	48.19	0.23
tmc2007	12.42	0.14
cal500	18.19	0.24
Corel-16k	27.32	0.21
Corel-5k	69.75	0.26
Mediamill	80.61	0.26

5. Conclusions

The multilabel data has the problems of imbalance and high concurrency of majority labels and minority labels. This paper proposed the ML-DBR algorithm at the data-level. By decoupling the highly concurrent data of the majority labels and minority labels and measuring the influence of the labels during resampling, the imbalance of the labels is reduced and the independence of the instances is guaranteed. Therefore, ML-DBR has the following advantages: (1) decoupling strategy is more effective and reasonable; (2) it combines undersampling and oversampling processes, which can reduce the redundancy of minority labels information caused by oversampling and the loss of majority labels information caused by undersampling, thus making the instance distribution more balanced and reducing the impact on minority label during the sampling process; (3) the original distribution state of the dataset will not be changed too much, and the original distribution of the dataset is maintained. Experiments show that the ML-DBR can effectively improve the classification performance of the classifier. ML-DBR achieved outstanding results on datasets with high TCS value, large number of labels, and high concurrency of labels with high scumble value. How to find more suitable m and t values for different datasets is the focus of our future work.

Data Availability

The datasets used to support the findings of this study have been deposited in the Mulan repository (http://mulan.sourceforge.net/datasets-mlc.html).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Natural Science Foundation of China under Grant no. 61373057, Natural Science Foundation of Zhejiang Province under Grant no. LY20F020009, and Science and Technology Planning Project of Lishui City under Grant no. 2019RC05.

References

[1] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 92–122, 2014.

[2] H. Zhu, G. Liu, M. Zhou, Y. Xie, A. Abusorrah, and Q. Kang, "Optimizing weighted extreme learning machines for imbalanced classification and application to credit card fraud detection," *Neurocomputing*, vol. 407, pp. 50–62, 2020.

[3] H. Elghazel, A. Aussem, O. Gharroudi, and W. Saadaoui, "Ensemble multi-label text categorization based on rotation forest and latent semantic indexing," *Expert Systems with Applications*, vol. 57, pp. 1–11, 2016.

[4] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[5] Z. A. Daniels and D. N. Metaxas, "Addressing imbalance in multi-label classification using structured hellinger forests," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 1826–1832, San Francisco, CA, USA, February 2017.

[6] C. L. Castro and A. P. Braga, "Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 888–899, 2013.

[7] S.-J. Huang, J.-L. Chen, X. Mu, and Z.-H. Zhou, "Cost-effective active learning from diverse labelers," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 1879–1885, Melbourne, Australia, August 2017.

[8] P. Cao, D. Zhao, and O. Zaiane, "An optimized cost-sensitive SVM for imbalanced data learning," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 280–292, Singapore, May 2013.

[9] X. Hong, S. Chen, and C. J. Harris, "A kernel-based two-class classifier for imbalanced data sets," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 28–41, 2007.

[10] H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 703–715, 2019.

[11] A. Akkasi, E. Varoğlu, and N. Dimililer, "Balanced undersampling: a novel sentence-based undersampling method to improve recognition of named entities in chemical and biomedical text," *Applied Intelligence*, vol. 48, no. 8, pp. 1965–1978, 2018.

[12] W. Lin and D. Xu, "Imbalanced multi-label learning for identifying antimicrobial peptides and their functional types," *Bioinformatics*, vol. 32, no. 24, pp. 3745–3752, 2016.

[13] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

[14] Q. Kang, X. Chen, S. Li, and M. Zhou, "A noise-filtered undersampling Scheme for imbalanced classification," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4263–4274, 2017.

[15] H. Ryang and U. Yun, "Top-*k* high utility pattern mining with effective threshold raising strategies," *Knowledge-Based Systems*, vol. 76, pp. 109–126, 2015.

[16] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, "EFIM: a fast and memory efficient algorithm for high-utility itemset mining," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 595–625, 2017.

[17] C. W. Wu, B.-E. Shie, V. S. Tseng, and P. S. Yu, "Mining top-*K* high utility itemsets," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD'12*, pp. 78–86, New York, NY, USA, August 2012.

[18] K. Chen, B.-L. Lu, and J. T. Kwok, "Efficient classification of multi-label and imbalanced data using min-max modular classifiers," in *Proceedings of the 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp. 1770–1775, Vancouver, Canada, July 2006.

[19] S. Wan, Y. Duan, and Q. Zou, "HPSLPred: an ensemble multi-label classifier for human protein subcellular location prediction with imbalanced source," *Proteomics*, vol. 17, no. 17-18, Article ID 1700262, 2017.

[20] M. A. Tahir, J. Kittler, and F. Yan, "Inverse random under sampling for class imbalance problem and its application to multi-label classification," *Pattern Recognition*, vol. 45, no. 10, pp. 3738–3750, 2012.

[21] S. Dendamrongvit and M. Kubat, "Undersampling approach for imbalanced training sets and induction from multi-label text-categorization domains," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 40–52, Bankok, Thailand, 2009.

[22] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 22–30, Sydney, Australia, May 2004.

[23] M.-L. Zhang, Y.-K. Li, and X.-Y. Liu, "Towards class-imbalance aware multi-label learning," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 4041–4047, Quebec City, Canada, 2015.

[24] F. Charte, A. Rivera, M. J. del Jesus, and F. Herrera, "A first approach to deal with imbalance in multi-label datasets," in *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*, pp. 150–160, Salamanca, Spain, September 2013.

[25] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, "Addressing imbalance in multilabel classification: measures and random resampling algorithms," *Neurocomputing*, vol. 163, pp. 3–16, 2015.

[26] N. Japkowicz and S. Stephen, "The class imbalance problem: a systematic study1," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[27] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, "Dealing with difficult minority labels in imbalanced mutilabel data sets," *Neurocomputing*, vol. 326-327, pp. 39–53, Jan. 2019.

[28] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, "REMEDIAL-HwR: tackling multilabel imbalance through label decoupling and data resampling hybridization," *Neurocomputing*, vol. 326-327, pp. 110–122, 2019.

[29] F. Charte, A. Rivera, M. J. del Jesus, and F. Herrera, "Concurrence among imbalanced labels and its influence on multilabel resampling algorithms," in *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*, pp. 110–121, Salamanca, Spain, June 2014.

[30] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognition*, vol. 45, no. 9, pp. 3084–3104, 2012.

[31] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: a java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, pp. 2411–2414, 2011.

[32] F. Charte, A. Rivera, M. J. del Jesus, and F. Herrera, "On the impact of dataset complexity and sampling strategy in multilabel classifiers performance," in *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*, pp. 500–511, Seville, Spain, April 2016.

[33] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.