Chapter 8
Imbalance in Multilabel Datasets

Abstract The frequency of class labels in many datasets is not even. On the contrary, that a certain class appears in a large portion of the data samples while other is scarcely represented is something quite usual. This casuistic produces a problem generically labeled as class imbalance. Due to these differences between class distributions, a specific need arises, imbalanced learning. This chapter beings introducing the mentioned task in Sect. 8.1. Then, the specific aspects of imbalance in the multilabel area are discussed in Sect. 8.2. Section 8.3 explains how imbalance in MLC has been faced, enumerating a considerable set of proposals. Some of them are experimentally evaluated in Sect. 8.4. Lastly, Sect. 8.5 summarizes the contents.

8.1 Introduction

Learning from imbalanced data is a challenge for many classification algorithms. Since most classifiers are designed to minimize a certain global error measurement, when they have to deal with imbalanced data, they tend to benefit the most frequent class. Miss-classification of rare classes does not have a great impact in the global performance assessment conducted by most evaluation metrics. However, depending on the scenario, the main interest of the task could be on correctly label these rare patterns, instead of the most common ones.

Imbalanced learning is a well-studied problem in the binary and multiclass scenarios [10, 13, 16, 19, 22]. The imbalance level in binary datasets is computed as the ratio between the most frequent or majority class and the less frequent one or minority class. It is the so-called Imbalance Ratio (*IR*), later adapted to work with multiclass datasets.

The imbalanced learning task has been faced mostly following one of three approaches:

- **Data resampling**: Resampling techniques are usually implemented as a preprocessing step, thus producing a new dataset from the original one. To balance the class distribution, it is possible to remove instances associated with the majority class or to generate new samples linked to the minority class [18]. Resampling methods are mostly classifier independent, so they can be seen as a general

F. Herrera et al., *Multilabel Classification*,
DOI 10.1007/978-3-319-41111-8_8

solution to this problem. Nonetheless, there are also some resampling proposals for specific classifiers.

- **Algorithm adaptation**: This approach is classifier dependent. Its goal is to modify existent classification algorithms to take into account the imbalanced nature of the data to be processed. The usual procedure is based on reinforcing the learning of the minority class, biasing the classifier to recognize it.
- **Cost-sensitive learning**: Cost-sensitive classification is an approach which combines the two previous techniques. The data are preprocessed to balance the class distribution, while the learning algorithm is adapted to benefit correct classification of samples associated with the minority class. To do so weights are associated with the instances, and usually these weights are proportional to the size of each class.

From these three ideas, many others have been derived, such as the combination of data resampling and the use of ensembles of classifiers [11] as a more robust model with certain tolerance to class imbalance.

Overall, imbalance learning is a well-known and deeply studied task in binary classification, further extended to also cover the multiclass scenario. Imbalance in multilabeled data increases the complexity of the problem and potential solutions, since there are several class labels per instance. In the following, the specificities of imbalanced MLDs, related problems, and proposed methods to tackle them are described.

Most of the existent methods only consider the presence of one majority class and one minority class. This way, undersampling methods only remove samples from one class, and analogously oversampling methods generate new instances associated with one class.

8.2 Imbalanced MLD Specificities

The number of labels in an MLD can go from a few dozens to several thousands. Only a handful of them have less than ten labels. Despite the fact that most MLDs have a large set of labels, the average number of active labels per instance (their cardinality) seldom is above 5. Some exceptions are `cal500` ($Card = 26.044$) and `delicious` ($Card = 19.017$). With such a large set of labels and low *Card*, that some labels would be underrepresented while others would be much more frequent can be deducted. As a general rule, the more labels there are in an MLD, the higher would be the likelihood of having imbalance problems.

Another important fact, easily deducible from the own MLDs nature, is that there is not a single majority label and a single minority one, but several of them in each group. This have different implications, affecting the way the imbalance level of an

MLD can be measured or the behavior of resampling and classification methods, as will be further detailed in the following sections of this chapter.

The way in which multilabel classification is faced can make worse the imbalance problem. Transformation techniques such as BR sometimes produce extreme imbalance levels. The binary dataset corresponding to a minority class will only have a few instances representing it, while all the others will belong to the opposite class. On the other hand, the LP transformation has to deal with rare label combinations, those in which the scarce minority labels appear, on their own or jointly with some of the majority ones. All the BR- and LP-based methods will face similar problems.

8.2.1 How to Measure the Imbalance Level

The metrics related to imbalance measurement for MLDs were provided in Chap. 3 (see Sect. 3.3.2). Since there are multiple labels, this trait cannot be easily reduced to a single value. For that reason, a triplet of metrics was proposed in the study conducted in [5]:

- *IRLbl*: It is independently computed for each label. The value of this metric will be 1 for the most frequent label and higher for all others. The larger is the *IRLbl* the less frequent is the assessed label in the MLD.
- *MeanIR*: By averaging the *IRLbl* for all labels in a MLD, its *MeanIR* is obtained. This value typically will be above 1. The higher is the *MeanIR*, the more imbalanced labels there are in the MLD.
- *CVIR*: The *MeanIR* is intended to give a measurement on the amount of imbalanced labels in the MLD, but it is also influenced by extreme values. A few very high-imbalanced labels can produce a high *MeanIR*, the same that a lot of less imbalanced labels. The *CVIR* is an indicator of the situation being assessed. Large *CVIR* values would denote high variances in *IRLbl*.

Besides the use of specific characterization metrics, such as the ones just described, one of the best approaches to analyze label imbalance in MLDs is to visually explore the data. In Fig. 8.1, the relative frequencies for the ten most frequent labels (left side) and the ten least frequent ones (right side) in a dozen MLDs have been plotted.[1] As can be observed, the difference between frequent and rare labels is huge. Even among the most frequent labels, there are significant disparities, with one or two labels having much more presence than the others. This pattern is common to many MLDs. Therefore, the imbalance problem is almost intrinsically linked to multilabel data.

[1]The frequency (Y-axis) scale is individually adjusted to show better the relevance of labels in each MLD, instead of being common to all plots.

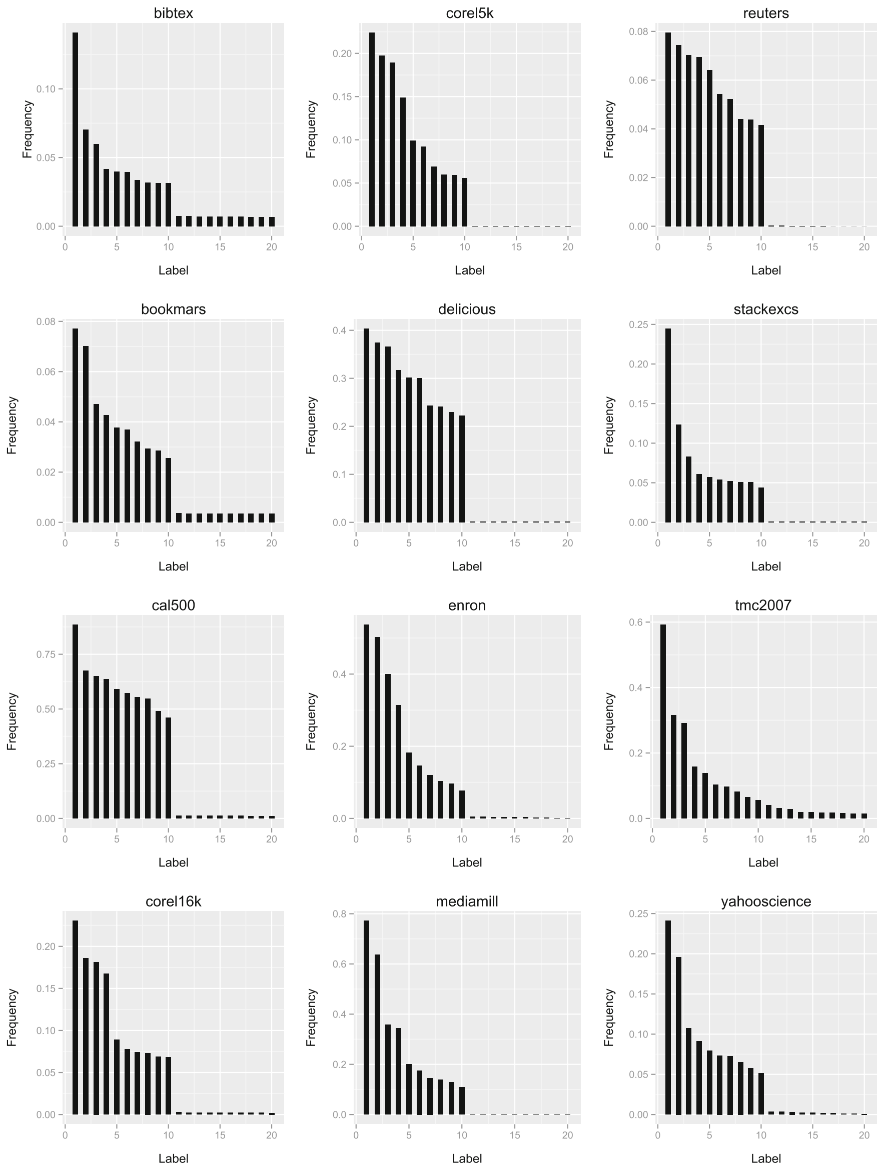

Fig. 8.1 Ten most frequent and ten least frequent labels in some datasets

8.2.2 Concurrence Among Imbalanced Labels

Looking at the graphical representations on Fig. 8.1, as well as to the imbalance levels reported in the tables on Chap. 3, it seems legitimate to think that applying resampling methods, as in traditional classification, the labels distribution on the MLDs could be balanced. However, MLDs have a specific characteristic which is not present on traditional datasets. As we are already aware, each data sample is associated with several outputs, and some of them can be minority labels while others are majority ones.

Due to this peculiarity, entitled as concurrence among imbalanced labels in [3], resampling methods could be not as effective as they should. In the same paper, a specific metric to assess this casuistic, named *SCUMBLE*, is proposed. It was defined in Sect. 3.3.3. As was demonstrated in this study, MLDs with large *SCUMBLE* values, that is with a high concurrence between minority and majority labels, usually do not benefit from resampling techniques as much as MLDs without this problem.

Visualizing the concurrence among imbalanced labels is not easy, since most MLDs have too many labels to show them at once along with their interactions.

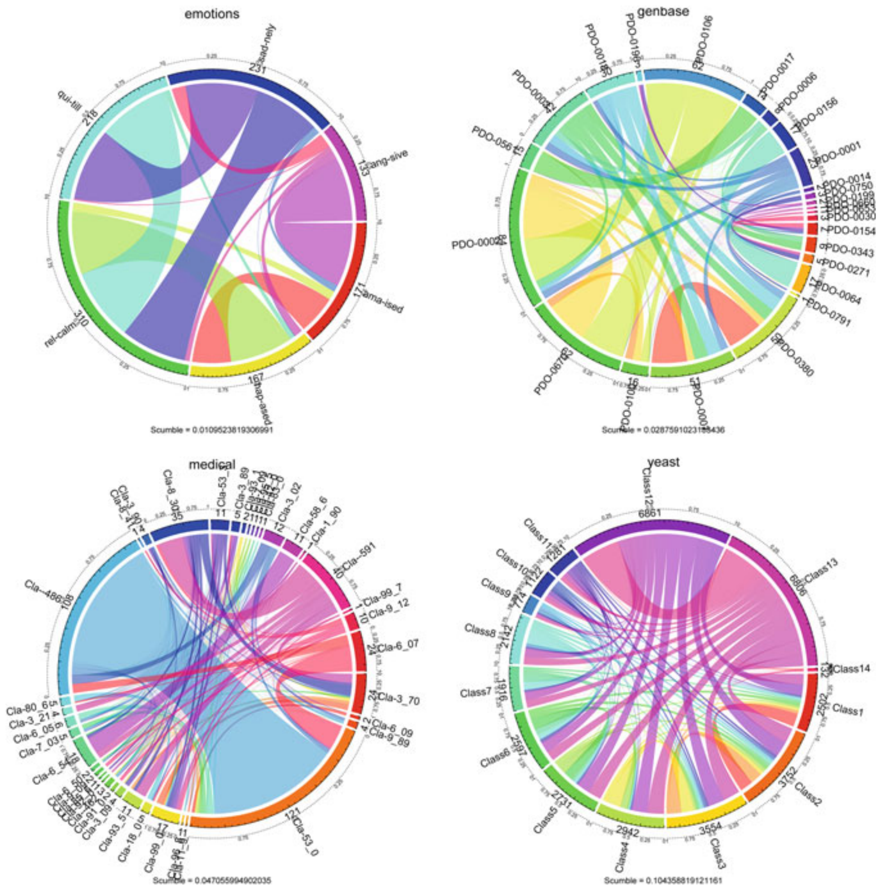

Fig. 8.2 Concurrence among imbalance labels in four MLDs

Nonetheless, it is possible to limit the number of labels to show, choosing those with higher dependencies, producing plots such as the ones shown[2] in Fig. 8.2.

Each arc in the external circumference represents a label. The arc's amplitude is proportional to the frequency of the label, so small arcs are associated with minority labels, and analogously large arcs indicate majority labels. The width of the bands connecting arcs denote the number of samples in which each label pair appears together.

Multilabel imbalance-aware methods able to take into account label concurrence could potentially produce better results than those that do not consider this information. A further section details such a method developed by the authors, called REMEDIAL.

8.3 Facing Imbalanced Multilabel Classification

On the basis of the specific characteristics associated with imbalanced MLDs, highlighted in the previous section, the design of algorithms capable of dealing with this problem is a challenge. Three main approaches have been followed in the literature, classifier adaptation, resampling methods, and ensembles of classifiers. Most of them are portrayed in the subsections below according to the aforementioned categorization scheme.

8.3.1 Classifier Adaptation

One way to face the imbalance problem consists in adapting the classifier to take this aspect into consideration, for instance assigning weights to each label depending on its frequency. Obviously, it is a solution tightly attached to the adjusted algorithm. Although it is not a general application approach, what can be seen as a disadvantage, the adaptation can strengthen the best point of a good classifier, something that a preprocessing method cannot do.

Some of the multilabel classifiers adapted to deal with imbalanced MLDs proposed in late years are the following:

- **Min–max modular with SVM (M^3-SVM)**: This method, proposed in [8], relies on a Min–Max Modular network [17] to divide the original multilabel classification problem into a set of simpler tasks. Several strategies are used to guarantee that the

[2]These plots were generated by the mldr R package, described in the following chapter.

imbalance level in these smaller tasks is lower than in the original MLD, following random, clustering, and PCA approaches. The simpler tasks are always binary classification jobs, using SVM as base classifier. Therefore, the proposal can be seen as a combination of data transformation and method adaptation techniques.

- **Enrichment process for neural networks**: The proposal made in [25] is an adaptation of the training process for neural networks. This task is divided into three phases. The first one uses a clustering method to group similar instances and gets a balanced representation to initialize the neural network. In the second stage, the network is iteratively trained, as usual, while data samples are added and removed from the training set, according to their prevalence. The final phase checks if the enrichment process has reached the stop condition or it has to be repeated. This way, the overall balance of the neural network used as classifier is improved.
- **Imbalanced multimodal multilabel learning (IMMML)**: In [14], the authors face an extremely imbalanced multilabel task, specifically the prediction of subcellular localization of human proteins. Their algorithm is based on a Gaussian process model, combined with latent functions on the feature space and covariance matrices to obtain correlations among labels. The imbalance problem is tackled giving each label a weighting coefficient linked to the likelihood of labels on each sample. Therefore, it is very specific solution to a definite problem, hardly applicable in a different context.
- **Imbalanced multiinstance multilabel radial basis function neural networks (IMIMLRBF)**: It was introduced in [15] as an extension to the MIMLRBF algorithm [26], a multiinstance and multilabel classification algorithm based on radial basis neural networks. The adaptation consists in two key points. Firstly, the number of units in the hidden layer, which in MIMLRBF is constant, is computed according to the number of samples of each label. Secondly, the weights associated with the links between the hidden and output layers are adjusted, biasing them depending on the label frequencies.

8.3.2 Resampling Techniques

The resampling approach is based on removing samples which belong to the majority label, adding samples associated with the minority label, or both actions at once. The way the instances to be removed are selected, and the technique used to produce new instances, usually follows one of two possible ways. The first one is the random approach, whereas the second one is known as heuristic approach. The former randomly chooses the data samples to delete, imposing the restriction that they have to belong to a certain label. Analogously, new samples are produced by randomly picking and cloning instances associated with a specific label. The latter path can be based on disparate heuristics to search the proper instances, as well as to generate new ones.

Therefore, resampling methods can be grouped depending on the way they try to balance the label frequency, removing or adding samples, and the strategy to do so, randomly or heuristically. There are quite a few proposals based on resampling techniques, among them:

- **Undersampling for imbalanced training sets in text categorization domains**: The proposal made in [9] combines the data transformation approach, producing a set of binary classifiers, with undersampling techniques, removing instances linked to the majority label to balance the distribution in each binary dataset. In addition, a decision tree is used to get the most relevant features for each label. kNN is used as underlying binary classifier, and different k values were tested in the conducted experimentation.

Algorithm 1 LP-RUS algorithm's pseudo-code.

```
   Inputs: <Dataset> $D$, <Percentage> $P$
   Outputs: Preprocessed dataset
1: $samplesToDelete \leftarrow |D|/100 * P$                ▷ P% size reduction
2: ▷ Group samples according to their labelsets
3: for $i = 1 \rightarrow |labelsets|$ do
4:     $labelSetBag_i \leftarrow$ samplesWithLabelset($i$)
5: end for
6: ▷ Calculate the average number of samples per labelset
7: $meanSize \leftarrow 1/|labelsets| * \sum_{i=1}^{|labelsets|} |labelSetBag_i|$
8: ▷ Obtain majority labels bags
9: for each $labelSetBag_i$ in $labelSetBag$ do
10:     if $|labelSetBag_i| > meanSize$ then
11:         $majBag_i \leftarrow labelSetBag_i$
12:     end if
13: end for
14: $meanRed \leftarrow samplesToDelete/|majBag|$
15: $majBag \leftarrow$ SortFromSmallestToLargest($majBag$)
16: ▷ Calculate # of instances to delete and remove them
17: for each $majBag_i$ in $majBag$ do
18:     $rBag_i \leftarrow \min(|majBag_i| - meanSize, meanRed)$
19:     $remainder \leftarrow meanRed - rBag_i$
20:     distributeAmongBags$_{j>i}$($remainder$)
21:     for $n = 1 \rightarrow rBag_i$ do
22:         $x \leftarrow$ random(1, $|majBag_i|$)
23:         deleteSample($majBag_i, x$)
24:     end for
25: end for
```

- **LP-based resampling (LP-ROS/LP-RUS)**: In [2], two resampling methods, named LP-ROS (*Label Powerset Random Oversampling*) and LP-RUS (*Label Powerset Random Undersampling*), are presented. As their names suggest, they do not evaluate the frequency of individual labels, but of full labelsets. LP-RUS removes instances from the most frequent labelsets, whereas LP-ROS clones samples associated with the least frequent ones. The pseudo-code for the LP-RUS algorithm is shown in Algorithm 1. As can be seen, the algorithm takes as input the percentage of samples to remove from the MLD. After computing the average number of samples sharing each labelset, a set of majority bags are produced. The number of instances to delete is distributed among these majority bags, randomly picking the data samples to remove. The LP-ROS algorithm works in a very similar fashion, but obtaining bags with minority labelsets and adding to them clones of samples randomly retrieved from them. These are simple techniques, and they consider the presence of several majority and minority combinations, instead of only one as most resampling methods assume.

Algorithm 2 ML-ROS algorithm's pseudo-code.

```
Inputs: <Dataset> $D$, <Percentage> $P$
Outputs: Preprocessed dataset
1: $samplesToClone \leftarrow |D|/100 * P$                          ▷ P% size increment
2: $L \leftarrow$ labelsInDataset($D$)                              ▷ Obtain the full set of labels
3: $MeanIR \leftarrow$ calculateMeanIR($D$, $L$)
4: for each $label$ in $L$ do                                      ▷ Bags of minority labels samples
5:     $IRLbl_{label} \leftarrow$ calculateIRperLabel($D$, $label$)
6:     if $IRLbl_{label} > MeanIR$ then
7:         $minBag_{i++} \leftarrow Bag_{label}$
8:     end if
9: end for
10: while $samplesToClone > 0$ do                                  ▷ Instances cloning loop
11:     ▷ Clone a random sample from each minority bag
12:     for each $minBag_i$ in $minBag$ do
13:         $x \leftarrow$ random(1, $|minBag_i|$)
14:         cloneSample($minBag_i$, $x$)
15:         if $IRLbl_{minBag_i}$ <=$MeanIR$ then
16:             $minBag \rightarrow minBag_i$                         ▷ Exclude from cloning
17:         end if
18:         - -$samplesToClone$
19:     end for
20: end while
```

• **Random resampling by label (ML-ROS/ML-RUS)**: As in the previous study, two resampling methods are also introduced in [5], one for oversampling and another one for undersampling. Both evaluate the individual imbalance level per label, deleting instances linked to the majority labels (ML-RUS) or cloning those associated with the minority ones (ML-ROS). The imbalance level is assessed by means of the *IRLbl* and *MeanIR* metrics defined in [2]. The removing/cloning process is iterative, and it reassess the imbalance levels in each iteration aiming to achieve the best balance for all labels. ML-ROS increases the number of instances in a given percentage, by cloning those with minority labels, while ML-RUS does the opposite by removing majority labels. The pseudo-code for ML-ROS is provided in Algorithm 2. Once the number of clones to produce is computed, the *IRLbl* and *MeanIR* are used to get a bag with the instances in which each minority label appears. The clones will be generated from these bags, following the iterative approach aforementioned. A new sample is created from each minority bag, reassessing their condition of minority bags in each cycle. This way, the best possible balance for each group is set as goal. The ML-RUS algorithm behavior is quite similar, but it gets bags with majority labels and iteratively removes samples from them.

- **A case study with the SMOTE algorithm**: The authors of the study published in [12] stated the imbalance problem in MLC, and proposed to face it using the original SMOTE (*Synthetic Minority Over-sampling Technique*) algorithm [18]. SMOTE was designed to produce synthetic instances of the minority class for binary datasets. In [12], three ways to feed SMOTE with multilabel data are tested, all of them considering one minority label only. The first path is similar to BR, giving to SMOTE all the instances in which the minority label appears to obtain synthetic samples from them and their neighbors. The second approach is quite limited, since only considers instances having the minority label alone, without any other labels. The third way, which probed to be the most effective, grouped the minority label instances according to the combinations of labels in which it appeared.

Algorithm 3 MLeNN algorithm pseudo-code.

```
Inputs: <Dataset> $D$, <Threshold> $HT$, <NumNeighbors> $NN$
Outputs: Preprocessed dataset

1: for each $sample$ in $D$ do
2:     for each $label$ in $getLabelset(D)$ do
3:         if IRLbl($label$) > $MeanIR$ then
4:             Jump to next sample                    ▷ Preserve instance with minority labels
5:         end if
6:     end for
7:     $numDifferences \leftarrow 0$
8:     for each $neighbor$ in nearestNeighbors($sample$, $NN$) do
9:         if adjustedHammingDist($sample$, $neighbor$) > $HT$ then
10:            $numDifferences \leftarrow numDifferences + 1$
11:        end if
12:    end for
13:    if $numDifferences \geq NN/2$ then
14:        markForRemoving($sample$)
15:    end if
16: end for
17: deleteAllMarkedSamples($D$)
```

- **Multilabel edited nearest neighbor (MLeNN)**: MLeNN is an heuristic undersampling algorithm. The method is proposed in [4], and it is build upon the well-known ENN (*Edited Nearest-Neighbor*) rule [21], foundation of a simple data cleaning procedure. It compares the class of each instance against the one of its NNs, usually its three NNs. Those samples whose class differs from the class of two or more NNs are marked for removing. The algorithm, presented in [4] and whose pseudo-code is shown in Algorithm 3, adapts the ENN rule to the MLC field introducing two key ideas, a principle to chose the samples acting as candidates to be removed and a comparison operator to determine when the labelsets of two instances are considered to be different. Only the instances which do not contain any minority label are used as candidates, instead of all the samples as in the original ENN implementation. Regarding how the classes of these instances are compared, a metric based on the Hamming distance among labelsets, but only taking into account active labels, is defined.

Algorithm 4 MLSMOTE algorithm's pseudo-code.

```
Inputs:
    $D$                                              ▷ Dataset to oversample
    $k$                                              ▷ Number of nearest neighbors

1: $L \leftarrow$ labelsInDataset($D$)                ▷ Full set of labels
2: $MeanIR \leftarrow$ calculateMeanIR($D$, $L$)
3: for each $label$ in $L$ do
4:     $IRLbl_{label} \leftarrow$ calculateIRperLabel($D$, $label$)
5:     if $IRLbl_{label} > MeanIR$ then
6:         ▷ Bags of minority labels samples
7:         $minBag \leftarrow$ getAllInstancesOfLabel($label$)
8:         for each $sample$ in $minBag$ do
9:             $distances \leftarrow$ calcDistance($sample$, $minBag$)
10:            sortSmallerToLargest($distances$)
11:            ▷ Neighbor set selection
12:            $neighbors \leftarrow$ getHeadItems($distances$, $k$)
13:            $refNeigh \leftarrow$ getRandNeighbor($neighbors$)
14:            ▷ Feature set and labelset generation
15:            $synthSmpl \leftarrow$ newSample($sample$,
16:                              $refNeigh$, $neighbors$)
17:            $D = D + synthSmpl$
18:        end for
19:    end if
20: end for
```

- **Multilabel SMOTE (MLSMOTE)**: This is another MLC oversampling method based on the SMOTE algorithm. However MLSMOTE, the proposal introduced in [6], is a full adaptation of the original algorithm toward the use of MLDs, instead of a procedure to use the unchanged SMOTE method with multilabel data as proposed in [12]. MLSMOTE considers several minority labels, instead of only one, taking the samples in which these labels appear as seeds to generate new data instances. To do so, firstly their nearest neighbors are found and the input features are obtained by interpolation techniques. Thus, the new instances are synthetic rather than mere clones of existing samples. Three approaches are tested to produce the synthetic labelsets associated with the new samples. Two of them rely on set operations among the labelsets of the NNs, computing the union or the intersection of active labels. The third one, eventually the one that produced better results, generates a ranking of labels in the NNs, keeping those present on half or more of the neighbors. As can be seen in Algorithm 4, corresponding to the main body of the MLSMOTE algorithm, the method relies on the *IRLbl* and *MeanIR* measurements to extract a collection of minority bags, each one corresponding to a label. Then, the k-nearest neighbors are retrieved. One of them will be used to reference instance to produce the synthetic features, while the labels on all of them (see Algorithm 5) serve to generate the synthetic labelset.

Algorithm 5 Function: Generation of new synthetic instances.

```
21: function NEWSAMPLE(sample, refNeigh, neighbors)
22:     synthSmpl ← new Sample                              ▷ New empty instance
23:     ▷ Feature set assignment
24:     for each feat in synthSmpl do
25:         if typeOf(feat) is numeric then
26:             diff ← refNeigh.feat - sample.feat
27:             offset ← diff * randInInterval(0,1)
28:             value ← sample.feat + offset
29:         else
30:             value ← mostFreqVal(neighbors,feat)
31:         end if
32:         syntSmpl.feat ← value
33:     end for
34:     ▷ Label set assignment
35:     lblCounts ← counts(sample.labels)
36:     lblCounts + ← counts(neighbors.labels)
37:     labels ← lblCounts > (k+1) / 2
38:     synthSmpl.labels ← labels
39:     return synthSmpl
40: end function
```

- **Resampling by decoupling highly imbalanced labels (REMEDIAL)**: None of the above resampling methods consider the concurrence among imbalanced labels, the problem previously described in Sect. 8.2.2. This is the differential factor of REMEDIAL, the method presented in [1] and whose pseudo-code is shown in Algorithm 6. It is an algorithm specifically designed to work with MLDs having a high *SCUMBLE*, the metric used to assess the concurrence level. It works both as an oversampling method and as an editing procedure. Firstly, the instances with high *SCUMBLE* values, those in which minority and majority labels appear together, are located. Then, for each sample in the previous set a new sample is produced by preserving the original features, but containing only minority labels. Lastly, the original sample is edited by removing these same minority labels. This way, the samples which can make harder the learning process are decoupled. As the authors highlight in [1], this method can be used as a previous step to apply other resampling techniques.

Algorithm 6 REMEDIAL algorithm.

```
1: function REMEDIAL(MLD $D$, Labels $L$)
2:     $IRLbl_l \leftarrow$ calculateIRLbl($l$ in $L$)            ▷ Calculate imbalance levels
3:     $IRMean \leftarrow \overline{IRLbl}$
4:     $SCUMBLEIns_i \leftarrow$ calculateSCUMBLE($D_i$ in $D$)    ▷ Calculate SCUMBLE
5:     $SCUMBLE \leftarrow \overline{SCUMBLEIns}$
6:     for each instance $i$ in $D$ do
7:         if $SCUMBLEIns_i > SCUMBLE$ then
8:             $D'_i \leftarrow D_i$                            ▷ Clone the affected instance
9:             $D_i[labels_{IRLbl<=IRMean}] \leftarrow 0$         ▷ Maintain minority labels
10:            $D'_i[labels_{IRLbl>IRMean}] \leftarrow 0$         ▷ Maintain majority labels
11:            $D \leftarrow D + D'_i$
12:        end if
13:    end for
14: end function
```

The main advantage of these methods is that they are classifier independent. They are used as a preprocessing step, even it is possible to combine them, and they do not require a specific multilabel classifier to be used. Therefore, the preprocessed MLDs can be later given as input to any of the MLC algorithms described in previous chapters.

8.3.3 The Ensemble Approach

Ensemble-based techniques are quite common in the MLC field. A significant number of proposals have been already published, as was reported in Chap. 6 devoted to multilabel ensembles. ECC, EPS, RAkEL, and HOMER are among the most popular MLC ensembles, an approach that also has been applied to solve the imbalance problem.

Theoretically, each classifier in an ensemble could introduce a bias toward a different set of labels, easing and making more effective the imbalanced learning task. The following two proposals are headed in this direction:

- **Inverse random undersampling (BR-IRUS)**: The method proposed in [24] is built upon an ensemble of binary classifiers. Several of them are trained for each label, using a subset of the original data with each one. This subset of the instances includes all samples in which the minority label is present, as well as a small portion of the remainder samples. This way, each individual classifier faces a balanced classification task. Joining the predictions given by the classifiers associated with a label, a more defined boundary around the minority label space is generated. The name of the proposal, BR-IRUS, highlights the fact of using the binary relevance transformation.

- **Ensemble of multilabel classifiers (EML)**: Developed by the same authors of the previous work, in [23] the construction of an heterogeneous ensemble of multilabel classifiers to tackle the imbalance problem is introduced. The ensemble is made up of five classifiers. All of them are trained with the same data, but using different algorithms. The underlying MLC classifiers chosen by the authors are RAkEL, ECC, CLR, MLkNN, and IBLR. Several methods for joining the individual predictions are tested, along with different thresholding and weighting schemes width adjustments made through cross-validation.

Usually, the major drawback of ensembles is their computational complexity, since a set with several classifiers has to be trained and their predictions have to be combined. This obstacle is more substantial in the case of EML, as the base classifiers are ensembles by themselves. In addition, these solutions are not classifier independent, being closer to the classifier adaptation approach than to resampling techniques.

8.4 Multilabel Imbalanced Learning in Practice

In the previous sections, most of the published methods aimed to tackle multilabel imbalanced learning have been portrayed. The goal in this section is to experimentally test some of them. Five methods, belonging to different techniques, have been chosen, specifically:

- **Random resampling**: Two algorithms based on random resampling techniques have been applied, ML-RUS and ML-ROS. The former performs undersampling, by removing samples associated with majority labels randomly picking them, while the latter does the opposite, producing clones of instances linked to minority labels.
- **Heuristic resampling**: This group of approaches is also represented by two methods, MLeNN and MLSMOTE. The first one removes instances with majority labels following the ENN rule. The second produces synthetic instances associated with minority labels, generating both features and labelsets from the information in the neighborhood.
- **Ensembles**: The EML (ensemble-based method), just described in the previous section, is also included in the test bed. Unlike the previous four algorithms, EML is not a preprocessing technique but a full classifier by itself, able to face imbalance by combining predictions coming from several classifiers with different biases.

These five methods[3] have been tested using the experimental configuration explained in the following section. Obtained results are presented and discussed in Sect. 8.4.2.

[3]The implementations of these methods can be found in the links section provided in this book repository [7], along with dataset partitions.

Table 8.1 Basic traits of MLDs used in the experimentation

Dataset	n	f	k	Card	Dens	MeanIR
`bibtex`	7 395	1 836	159	2.402	0.015	12.498
`cal500`	502	68	174	26.044	0.150	20.578
`corel5k`	5 000	499	374	3.522	0.009	189.568
`medical`	978	1 449	45	1.245	0.028	89.501
`tmc2007`	28 596	49 060	22	2.158	0.098	15.158

8.4.1 Experimental Configuration

Four out of the five imbalance methods to be tested are preprocessing procedures. Therefore, once they have done their work, producing the rebalanced MLD, the data has to be given to an MLC classifier in order to obtain comparable classification results. A basic BR transformation has been used for this duty, with the C4.5 [20] tree induction algorithm as underlying binary classifier.

Five MLDs with disparate imbalance levels, `bibtex`, `cal500`, `corel5k`, `medical` and `tmc2007`, have been included in the experimentation. Their basic traits, including the *MeanIR*, are provided in Table 8.1. Each MLD was partitioned with a 2×5 fold cross-validation scheme, as usual. Training partitions were preprocessed with ML-RUS, ML-ROS, MLeNN, and MLSMOTE.

Thus, five versions of each one were used, one without resampling and four more preprocessed by each method. The original version, without resampling, was given as input to the BR classifier to obtain a base evaluation. It was also used with EML, which did not need an independent classifier. The preprocessed versions also served as input to the same BR + C4.5 MLC, with exactly the same configuration parameters.

In Chap. 3, the metrics designed to assess MLC algorithms performance were introduced. Many of them, such as *Hamming Loss*, *Accuracy*, *Precision*, *Recall*, and *F-measure*, have been used in the experiments of previous chapters. To study the behavior of classifiers when working with imbalanced data, as it is done here, it is usual to rely on label-based metrics, instead of sample-based ones. In this case, *F-measure* following the macro- and microaveraging strategies are the metrics obtained to assess the results. *MacroFM* (Macro-F-measure) assigns the same weight to all labels, while *MicroFM* (Micro-F-measure) is heavily influenced by the frequencies of each label. Therefore, the former is usually used to assess the performance with respect to minority labels, and the latter to obtain a more general view of the classifier's behavior.

8.4.2 Classification Results

Classification results assessed with *MacroFM* are shown in Fig. 8.3. Each group of bars corresponds to an MLD, with each bar depicting the performance of a method.

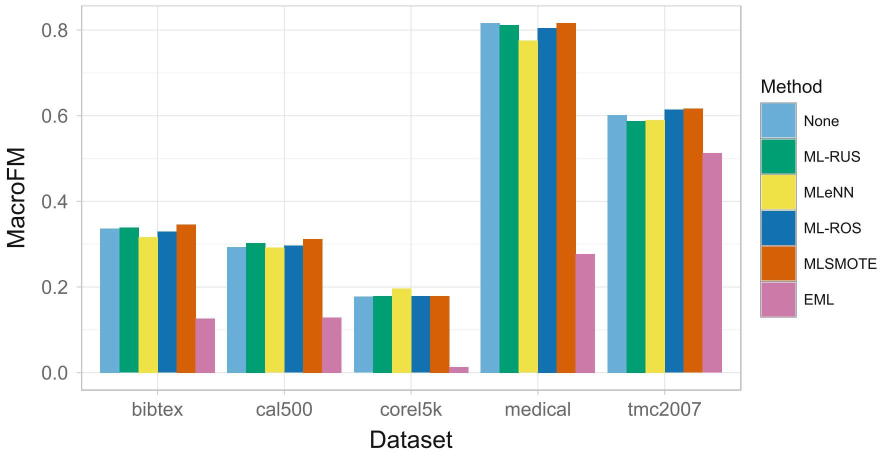

Fig. 8.3 Classification results assessed with the Macro-FMeasure metric

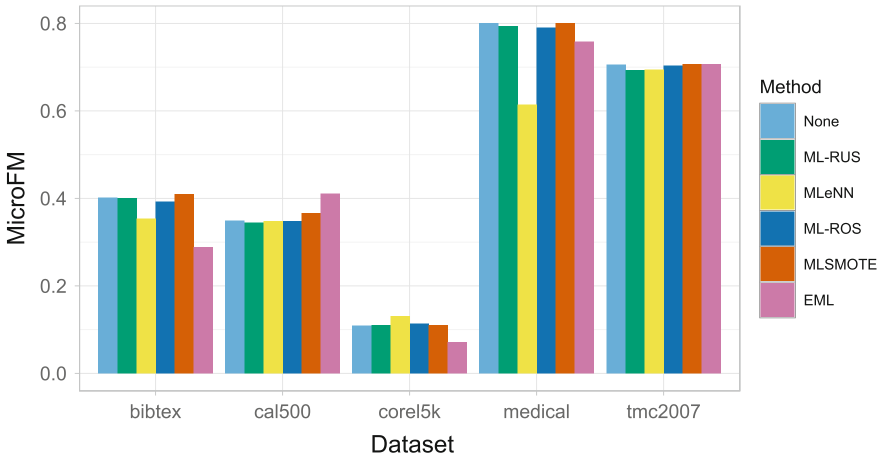

Fig. 8.4 Classification results assessed with the Micro-FMeasure metric

The left-most bar shows the value for base results, those obtained without any special imbalance treatment.

Analogously, Fig. 8.4 presents the results evaluated with the *MicroFM* metric. The structure of the plot is exactly the same. To analyze these data, it would be interesting to know in which cases an imbalance treatment has achieved some improvement over the base results. Another important fact is which one of the applied methods works better.

As can be observed in the two previous plots, undersampling methods seem to behave worse than the oversampling ones. The exception is MLeNN with the `corel5k` MLD, which achieves the best results with the two evaluation metrics.

Table 8.2 Results assessed with *MacroFM* (higher is better)

Dataset	Base	ML-RUS	MLeNN	ML-ROS	MLSMOTE	EML
`bibtex`	0.3368	*0.3383*	0.3170	0.3288	**0.3457**	0.1265
`cal500`	0.2933	*0.3029*	0.2918	*0.2966*	**0.3124**	0.1291
`corel5k`	0.1774	*0.1792*	**0.1966**	*0.1784*	*0.1790*	0.0133
`medical`	0.8165	0.8117	0.7750	0.8046	**0.8165**	0.2770
`tmc2007`	0.6015	0.5878	0.5903	*0.6138*	**0.6165**	0.5122

Table 8.3 Results assessed with *MicroFM* (higher is better)

Dataset	Base	ML-RUS	MLeNN	ML-ROS	MLSMOTE	EML
`bibtex`	0.4021	0.4007	0.3533	0.3927	**0.4097**	0.2888
`cal500`	0.3488	0.3447	0.3478	0.3478	*0.3663*	**0.4106**
`corel5k`	0.1096	*0.1109*	**0.1315**	*0.1135*	*0.1103*	0.0712
`medical`	0.8006	0.7935	0.6149	0.7902	**0.8006**	0.7581
`tmc2007`	0.7063	0.6934	0.6947	0.7038	**0.7071**	*0.7065*

EML, the ensemble-based solution, does not produce good *MacroFM* results, although with *MicroFM* the performance seems to be slightly better, specifically with the `cal500` MLD. Regarding the oversampling methods, MLSMOTE appears as the best performed almost always. In fact, this method accomplishes the best results in many cases.

The *MacroFM* and *MicroFM* raw values are provided in Tables 8.2 and 8.3, respectively. Values highlighted in italics denote an amelioration with respect to results without imbalance treatment. Best values across all methods are emphasized in bold, as usual.

From these values observation it can be stated that EML seldom reaches the performance of the BR + C4.5 base classifier, although it achieves the best *MicroFM* result with the `cal500` MLD. In comparison, MLSMOTE improves base results always for both metrics and manages to get the best performance in seven out of ten configurations. ML-RUS and ML-ROS produce some improvements, as well as a few losses. Lastly, MLeNN seems to work well with the `corel5k` MLD, but its behavior with the other four datasets is not as good.

Overall, it seems that advanced preprocessing techniques, such as the MLSMOTE algorithm, are able to improve MLC results while dealing with imbalanced MLDs.

8.5 Summarizing Comments

Class imbalance is a very usual obstacle while learning a classification model. In this chapter, how label imbalance is present in most MLDs, and some of the specificities, in this field such as label concurrence among imbalanced labels, have been introduced. Several metrics aimed to assess these traits have been explained, and some specialized data visualizations have been provided.

Solutions to deal with imbalanced multilabel data can be grouped into a few categories, including preprocessing methods, algorithm adaptation, and ensembles. A handful of proposals from each category have been described, and some of them have been experimentally tested. According to the results obtained, the resampling techniques deliver certain improvements while maintaining the benefit of being classifier independent.

References

1. Charte, F., Rivera, A., del Jesus, M.J., Herrera, F.: Resampling multilabel datasets by decoupling highly imbalanced labels. In: Proceedings of 10th International Conference on Hybrid Artificial Intelligent Systems, HAIS'15, vol. 9121, pp. 489–501. Springer (2015)
2. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: A first approach to deal with imbalance in multi-label datasets. In: Proceedings of 8th International Conference on Hybrid Artificial Intelligent Systems, HAIS'13, vol. 8073, pp. 150–160. Springer (2013)
3. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: Concurrence among imbalanced labels and its influence on multilabel resampling algorithms. In: Proceedings of 9th International Conference on Hybrid Artificial Intelligent Systems, HAIS'14, vol. 8480. Springer (2014)
4. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: MLeNN: a first approach to heuristic multilabel undersampling. In: Proceedings of 15th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL'14, vol. 8669, pp. 1–9. Springer (2014)
5. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: Addressing imbalance in multilabel classification: measures and random resampling algorithms. Neurocomputing **163**, 3–16 (2015)
6. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: MLSMOTE: approaching imbalanced multilabel learning through synthetic instance generation. Knowl. Based Syst. **89**, 385–397 (2015)
7. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: Multilabel Classification. Problem analysis, metrics and techniques book repository. https://github.com/fcharte/SM-MLC
8. Chen, K., Lu, B., Kwok, J.: Efficient classification of multi-label and imbalanced data using min-max modular classifiers. In: Proceedings of IEEE International Joint Conference on Neural Networks, IJCNN'06, pp. 1770–1775 (2006)
9. Dendamrongvit, S., Kubat, M.: Undersampling approach for imbalanced training sets and induction from multi-label text-categorization domains. In: New Frontiers in Applied Data Mining. LNCS, vol. 5669, pp. 40–52. Springer (2010)
10. Fernández, A., López, V., Galar, M., del Jesus, M.J., Herrera, F.: Analysing the classification of imbalanced data-sets with multiple classes: binarization techniques and ad-hoc approaches. Knowl. Based Syst. **42**, 97–110 (2013)
11. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., Herrera, F.: An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes. pattern Recogn. **44**(8), 1761–1776 (2011)

12. Giraldo-Forero, A.F., Jaramillo-Garzón, J.A., Ruiz-Muñoz, J.F., Castellanos-Domínguez, C.G.: Managing imbalanced data sets in multi-label problems: a case study with the SMOTE algorithm. In: Proceedings of 18th Iberoamerican Congress on Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, CIARP'13, vol. 8258, pp. 334–342. Springer (2013)
13. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)
14. He, J., Gu, H., Liu, W.: Imbalanced multi-modal multi-label learning for subcellular localization prediction of human proteins with both single and multiple sites. PloS One **7**(6), 7155 (2012)
15. Li, C., Shi, G.: Improvement of learning algorithm for the multi-instance multi-label RBF neural networks trained with imbalanced samples. J. Inf. Sci. Eng. **29**(4), 765–776 (2013)
16. López, V., Fernández, A., García, S., Palade, V., Herrera, F.: An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf. Sci. **250**, 113–141 (2013)
17. Lu, B., Ito, M.: Task decomposition and module combination based on class relations: a modular neural network for pattern classification. IEEE Trans. Neural Netw. **10**(5), 1244–1256 (1999)
18. Nitesh, V.C., Kevin, W.B., Lawrence, O.H., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
19. Prati, R.C., Batista, G.E., Silva, D.F.: Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. Knowl. Inf. Syst. **45**(1), 247–270 (2015)
20. Quinlan, J.R.: C4.5: Programs for Machine Learning (1993)
21. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman & Hall (2003)
22. Sun, Y., Wong, A.K.C., Kamel, M.S.: Classification of imbalanced data: a review. Int. J. Pattern Recogn. Artif. Intell. **23**(4), 687–719 (2009)
23. Tahir, M.A., Kittler, J., Bouridane, A.: Multilabel classification using heterogeneous ensemble of multi-label classifiers. Pattern Recogn. Lett. **33**(5), 513–523 (2012)
24. Tahir, M.A., Kittler, J., Yan, F.: Inverse random under sampling for class imbalance problem and its application to multi-label classification. Pattern Recogn. **45**(10), 3738–3750 (2012)
25. Tepvorachai, G., Papachristou, C.: Multi-label imbalanced data enrichment process in neural net classifier training. In: Proceedings of IEEE International Joint Conference on Neural Networks, IJCNN'08, pp. 1301–1307. IEEE (2008)
26. Zhang, M., Wang, Z.: MIMLRBF: RBF neural networks for multi-instance multi-label learning. Neurocomputing **72**(16), 3951–3956 (2009)