Chapter 9
Imbalanced Multi-instance Data

Abstract Class imbalance is widely studied in single-instance learning and refers to the situation where the data observations are unevenly distributed among the possible classes. This phenomenon can present itself in MIL as well. Section 9.1 presents a general introduction to the topic of class imbalance, list the types of solutions to deal with it, and the appropriate performance metrics. In Sect. 9.2, we recall a popular single-instance method addressing class imbalance. We provide a detailed specification of multi-instance class imbalance in Sect. 9.3 and discuss its solutions in Sect. 9.4 on resampling methods and in Sect. 9.5 on custom classification methods. Section 9.6 presents the experimental analysis accompanying this chapter. Some summarizing remarks are listed in Sect. 9.7.

9.1 Introduction

In the presence of class imbalance, the possible classes are unevenly represented in the dataset. Some classes may contain many observations, while others only have very few in comparison. The most common setting is that of a binary or two-class problem, where the instances of the majority class considerably outnumber those of the minority class. Elements of the minority class are usually labeled as positive and those of the majority class as negative. These names indicate that in most applications the minority class is the class of interest. In recent years, the focus of class imbalanced learning has widened to the general setting of multi-class classification, where the number of classes may exceed two. In this situation, there can be a mixture of majority, medium-sized, and minority classes, which automatically yields more challenging learning objectives. A large body of work has been done on the classification of imbalanced data in single-instance problems [18, 21, 26, 31]. Application areas in which class imbalance naturally presents itself include medical diagnosis [8, 19, 20, 22] and bioinformatics [41, 42, 44].

F. Herrera et al., *Multiple Instance Learning*, DOI 10.1007/978-3-319-47759-6_9

9.1.1 Dealing with Class Imbalance

Traditional classifiers tend to lose some of their prediction strength in the presence of class imbalance, because they make the internal assumption of similar class distributions or misclassification costs. By definition, the former is violated for imbalanced data. The latter premise does not hold either, since a higher cost is usually associated with the misclassification of a positive element than with that of a negative observation. As a result, standard classifiers fail, for instance by predicting the majority label over-easily. Specific solutions to handle class imbalance have been proposed in the literature. We can divide these approaches into two general groups:

- **Data-level solutions**: this group consists of preprocessing methods known as *resampling* techniques. They modify the dataset before the application of a classifier, which means that they are independent of the latter. We distinguish between undersampling methods, that remove part of the majority class, oversampling methods, that add new minority elements, and hybrid methods, that combine the previous two approaches. A popular single-instance method, commonly used in comparative studies on class imbalance, is the SMOTE oversampling method [7]. It is described in detail in Sect. 9.2.
- **Customized approaches**: a second group of solutions handling class imbalance is found at the algorithm-level. These methods do not modify the data. We can make a further distinction between three diverse types of methods.
  - The first subgroup consists of cost-sensitive methods (e.g., [11, 43]). These algorithms assign different misclassification costs to the classes and aim to minimize the overall cost. In this way, relatively more focus can be put on the correct classification of minority class elements.
  - Second, we list the methods that focus on the construction of a classification model that is not hindered by the imbalance between classes. Based on imbalance-resistant heuristics, a learner is designed to tackle the imbalance problem.
  - Finally, a third subgroup is formed by custom ensemble techniques (e.g., [16]), that have already been used, possibly in combination with resampling methods, in the classification of single-instance imbalanced data.

Both types of approaches have been proposed in single-instance as well as multi-instance learning. The latter will be discussed in detail in Sect. 9.3.

With respect to multi-class imbalanced learning, decomposition strategies can be used to divide the multi-class problem in a set of binary ones, as done for single-instance methods in e.g., [14]. In that study, the one-versus-one and one-versus-all methods are used to derive binary prediction problems from the multi-class dataset. For each of them, a two-class preprocessing method is applied in conjunction with a classifier. The outcomes of all binary problems are aggregated to yield a single prediction value. Any binary solution can be combined with a decomposition scheme and aggregation method to perform a multi-class imbalanced classification. In single-instance learning, multi-class resampling methods as well as general classifiers to deal

with multi-class imbalance without a decomposition step have also been proposed as well (e.g., [1, 39]).

9.1.2 Evaluation Measures in the Imbalanced Domain

The evaluation of classification performance in the class imbalanced domain warrants metrics that are not sensitive to the skewness in the class distributions. Among the measures listed in Sect. 1.4, the accuracy is the most commonly used in general studies. However, the research community agrees that it is not an appropriate measure to use in the presence of class imbalance, as it can lead to misleading results. As an example, consider a dataset with a 1000 observations, of which 900 belong to the negative class and the remaining 100 to the positive class. When a classifier predicts that each observation is negative, it attains an accuracy of 90 %. This is a high value and it does not in any way reflect the fact that the entire positive class has been misclassified. We conclude that this metric does not provide a faithful representation of the performance of the classifier.

As an alternative to the accuracy, we can use g, the geometric mean of the class-wise accuracies. The general definition is provided in Sect. 1.4. In a two-class setting, which is most common in studies on class imbalance, this reduces to

$$g = \sqrt{\text{TPR} \cdot \text{TNR}},$$

where TPR and TNR, respectively, correspond to the true positive and true negative rates of the classification. By computing the rate of correct predictions for each class separately, none of the classes can be ignored. In particular, in the above example, the value for g is zero, since all positive observations were misclassified. This clearly reflects the incapacity of the classifier.

A second measure that is commonly used in research on class imbalance, is the Area Under the ROC-curve (AUC, [3]), defined for two-class problems. A ROC-curve models the trade-off between true positive and false positive classifier predictions. It was originally defined for probabilistic classifiers, that use a threshold value θ on the positive class probability. When the estimated probability is higher than θ, the sample is classified as positive. In the other case, the negative class is predicted. By varying θ, different true positive and false positive rates are obtained. Each represents a point in ROC-space and together they form the ROC-curve. The area under it gathers the information represented by the curve in a single value. It can be computed by using the procedure described in [13]. A detailed description of ROC-curves and AUC computations can also be found in [33].

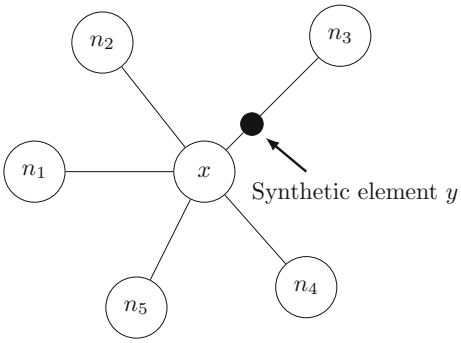

Fig. 9.1 Illustration of the construction of one artificial element in a single-instance dataset by SMOTE, between the seed x and a randomly selected instance from among its nearest neighbors n_i in the minority class

9.2 Single-Instance SMOTE

As noted above, one way to deal with class imbalance is to resample the dataset. In single-instance learning, a popular resampling method is the Synthetic Minority Oversampling Technique (SMOTE) proposed by Chawla et al. [7]. We recall this algorithm here, as several multi-instance resampling proposals (Sect. 9.4) are based on it.

SMOTE is an oversampling method, that increases the size of the minority class by adding artificial new instances to it. Synthetic instances are constructed by selecting one of the existing minority elements as seed and introducing a new instance at a random position on the line segment connecting this seed element to one of its k nearest minority class neighbors, as illustrated in Fig. 9.1. The pseudo-code of this method can be found in Algorithm 19. We present the version that yields a perfectly balanced dataset, that is, positive instances are created until their class reaches the size of the negative class. This is the most commonly used setting, although the amount of oversampling can also be controlled by a parameter, as presented in the original proposal [7]. The value k is commonly set to 5. Step 8 of Algorithm 19 creates an artificial instance. Existing data samples are interpreted as vectors and a new element is introduced at a random point on the line segment between them. The user should take care to use an appropriate interpolation scheme for numeric and categorical attributes.

9.3 Multi-instance Class Imbalance

We continue with a discussion on the presence of class imbalance in multi-instance data. Compared to the single-instance setting, a very limited amount of work has been done on multi-instance class imbalance. However, interest has been raised in the past few years and it forms a promising area of future research.

Algorithm 19 SMOTE algorithm

Input: Imbalanced single-instance dataset T, with positive class P and negative class N. Number of neighbors k
Output: A perfectly balanced dataset T^*

$d \leftarrow |N| - |P|$ ▷ amount of artificial instances
$A \leftarrow \emptyset$ ▷ initialize set of artificial instances
for $i = 1, \ldots, d$ **do**
  $x \leftarrow$ random positive instance from T
  $\{n_1, n_2, \ldots, n_k\} \leftarrow k$ nearest neighbors of x in P
  $n \leftarrow$ random selection from $\{n_1, n_2, \ldots, n_k\}$
  $\alpha \leftarrow$ random value drawn from $[0, 1]$
  $y \leftarrow x + \alpha \cdot (n - x)$ ▷ generate a new instance by interpolation
  Label y as positive
  $A \leftarrow A \cup \{y\}$
end for
$T^* \leftarrow T \cup A$

9.3.1 Problem Description

The nature of class imbalance can be more complex in MIL than it is in single-instance learning. As argued by Wang et al. [35], the imbalance can occur at two different levels, namely that of the instances and that of the bags. Instance-level imbalance means that the number of positive instances in positive bags is relatively low compared to the number of negative instances they contain. In the standard MIL hypothesis, only one positive instance is required in a positive bag, such that imbalance at the level of instances can easily present itself. Imbalance can also occur at the bag-level, which is the direct generalization of class imbalance in single-instance learning and fits in the general description given in Sect. 9.1. It means that the number of positive bags is small compared to the number of negative bags.

The effects that these two types of imbalance have on multi-instance classifiers have been described by e.g., Mera et al. [23]. In the presence of instance-level imbalance, instance-based methods (Chap. 4) are likely to be biased toward the majority class, since the number of actual positive instances is low compared to actual negative instances. Mapping-based methods (Sect. 5.3) can also be hindered by this type of imbalance, when the mapping step is sensitive to instance imbalance. The example given in [23] to illustrate this situation is the averaging function: if the number of positive instances is relatively small in a bag, their information will be mostly lost when averaging over all instances. Bag-level imbalance results in a bias of bag-based classifiers (Sect. 5.2) toward the majority class.

The work of Wang et al. [35] showed that multi-instance imbalance is most pronounced at the level of the bags. Keeping this in mind, we define class imbalance for multi-instance problems at the bag-level. It is reflected in a (possibly considerably) larger number of bags of the majority class compared to the number of bags of the minority class. The degree of imbalance is measured by the so-called *imbalance ratio* (IR), which is defined as the ratio of the number of bags of the majority class over

the number of bags of the minority class. Class imbalance results in more difficulty to recognize the minority class, leading to a high number of misclassifications of these bags. The study of [35] references the application of [37] concerning the object detection of mines based on sonar images. This is an imbalanced problem, since mines occur far less often than other objects (e.g., rocks).

9.3.2 Solutions for Multi-instance Class Imbalance

To date, we are not aware of any specific proposals to deal with multi-class imbalance in MIL. The existing solutions have been specifically proposed with binary problems in mind and have only been evaluated on this kind of data. However, the decomposition strategies discussed in Sect. 9.1 can be transferred from the single-instance to the multi-instance setting, in conjunction with any of the binary class imbalance solutions described below.

As noted above, compared to the amount of attention that the class imbalance problem has received in single-instance learning, its exploration in MIL has been very limited. Nevertheless, applications inherently prone to class imbalance also present themselves in this domain and custom methods able to deal with the intrinsic challenges of class imbalance are warranted. We discuss the recent developments in Sects. 9.4 and 9.5. As will be clear from Sect. 9.4, the study of multi-instance preprocessing techniques has so far been largely limited to extensions of the SMOTE method recalled in Sect. 9.2. In single-instance learning, this technique has been improved upon by many authors (e.g., [4, 5, 29]) and it is a valid question whether similar improvements can be made in MIL. Extensions of other single-instance solutions, like cost-sensitive support vector machines [32] or alternative ensemble techniques (e.g., [17, 38]), should be further explored as well.

9.4 Multi-instance Resampling Methods

In this section, we discuss the resampling methods that have been proposed to deal with multi-instance class imbalance. We recall the contributions of [23, 24, 35], which, to the best of our knowledge, form the complete set of multi-instance resampling methods dealing with class imbalance.

9.4.1 BagSMOTE, InstanceSMOTE, Bag_oversampling

Wang et al. [35] consider two approaches to deal with class imbalance in MIL: resampling on the one hand and cost-sensitive methods on the other. We describe their resampling methods here, the discussion of the cost-sensitive methods is postponed

to Sect. 9.5. Two extensions of the SMOTE method of [7] are developed. Their first method is called BagSMOTE and creates synthetic minority bags. The procedure is presented in Algorithm 20. Each existing minority bag X leads to the creation of one new bag, that is also labeled with the minority class. For each instance $x \in X$, one of its nearest instances from within all minority bags is selected. A synthetic element is generated between x and its neighbor and added to the new bag.

Algorithm 20 BagSMOTE algorithm

Input: Imbalanced multi-instance dataset $\mathbf{T}$, number of neighbors k
Output: The oversampled dataset $\mathbf{T}^*$
$\mathbf{A} \leftarrow \emptyset$ ▷ initialize set of artificial bags
for each positive bag $X \in \mathbf{T}$ **do**
  $Y \leftarrow \emptyset$ ▷ initialize artificial bag
  for each instance $x \in X$ **do**
    $\{n_1, n_2, \ldots, n_k\} \leftarrow k$ nearest neighbors of x among all instances in all positive bags
    $n \leftarrow$ random selection from $\{n_1, n_2, \ldots, n_k\}$
    $\alpha \leftarrow$ random value drawn from $[0, 1]$
    $y \leftarrow x + \alpha \cdot (n - x)$ ▷ generate a new instance by interpolation
    $Y \leftarrow Y \cup y$
  end for
  Label Y as positive
  $\mathbf{A} \leftarrow \mathbf{A} \cup \{Y\}$
end for
$\mathbf{T}^* \leftarrow \mathbf{T} \cup \mathbf{A}$

The second method is called InstanceSMOTE and is presented in Algorithm 21. It modifies the multi-instance dataset to a single-instance dataset by assigning all instances to the class to which their parent bag belongs. The single-instance SMOTE method is applied to this transformed dataset. Afterwards, the dataset is changed back to its original multi-instance form, by assigning synthetic instances to the same bag from which the corresponding seed was drawn. Clearly, the largest difference between the BagSMOTE and InstanceSMOTE methods is that the first one creates new bags, while the second one inserts new instances in already existing bags.

In their experiments, Wang et al. also include a third alternative, that randomly duplicates minority bags to obtain a better balance between the classes. This method is referred to as Bag_oversampling. The experimental study shows that the BagSMOTE method yields the best results among the three proposals. The more intricate oversampling procedure in BagSMOTE has a better performance than the random procedure in Bag_oversampling. However, Bag_oversampling outperforms InstanceSMOTE, which supports the statement of the authors that multi-instance class imbalance occurs at the level of the bags and that there is where a solution should be applied.

Algorithm 21 InstanceSMOTE algorithm

Input: Imbalanced multi-instance dataset $\mathbf{T}$, number of neighbors k
Output: The oversampled dataset $\mathbf{T}^*$
1: $T \leftarrow \{(x, class(X)) \mid x \in X, X \in \mathbf{T}\}$ ▷ transformation to single-instance data
2: $T^* \leftarrow$ output of Algorithm 19 on T with value k. Store parent bag of seed instance for each artificial element.
3: $\mathbf{T}^* \leftarrow \emptyset$
4: **for each** parent bag $X \in \mathbf{T}$ **do** ▷ transformation to multi-instance data
5: $X^* \leftarrow$ set of all instances $x \in T^*$, either original or artificial, linked to this parent bag
6: Label X^* with the class of X
7: $\mathbf{T}^* \leftarrow \mathbf{T}^* \cup \{X^*\}$
8: **end for**

9.4.2 B-Instances

Mera et al. [24] proposed a preprocessing method to improve the classification of imbalanced multi-instance data by means of an enriched representation of the positive class. In their later work [23], they refer to this method as B-Instances. By means of kernel density estimation [25], they construct a function which estimates the degree to which an instance can be considered as negative. This measure is used to locate likely and unlikely positive elements within positive bags and use them in resampling procedures. The method consists of three main steps:

1. **Oversampling within positive bags**: the set T^+ is constructed containing the most positive (least negative) instance from every positive bag. In their experiments, the authors increase T^+ to include the second most positive instance from each positive bag as well, to improve the performance of the method. When T^+ has been determined, SMOTE is applied to oversample instances within the positive bags. The elements from T^+ are used as seeds. A synthetic instance is generated at a random position on the line segment connecting the seed with one of its k nearest neighbors in the entire dataset. The constructed instance is added to the bag which contained the seed element.
2. **Undersampling within positive bags**: undersampling is applied to the positive bags. To this end, the least positive (most negative) instance in each positive bag is located and added to the set T^-. For every element $x \in T^-$, its k nearest neighbors from among all data instances are determined. When the majority of these neighbors originate from negative bags, x is interpreted as a borderline element and it is decided to remove it.
3. **Undersampling within negative bags**: the third stage consists of removing instances from negative bags, using a similar procedure as in the undersampling of the positive bags. For each instance in a negative bag, its k nearest neighbors in the entire dataset are determined. When the majority of its neighbors belong to positive bags, the instance is a borderline element in the negative bag and is removed.

In the experimental study of [24], their B-Instances preprocessing method was combined with several classifiers and was shown to improve their performance on imbalanced multi-instance datasets. However, no comparison was offered with the methods of [35].

9.4.3 B-Bags

The proposal of [23] uses kernel density estimation as well. It is largely based on the standard MIL hypothesis, in that the method creates new positive bags that contain only one positive instance. The method is called B-Bags. Based on the kernel density estimation procedure, B-Bags aims to determine the most positive instance in the positive bags. It is a bag oversampling method and creates a total number of n new positive bags. For the construction of a synthetic positive bag, the following steps are performed:

1. **Positive instance**: one artificial positive instance is constructed and added to the new bag. Two random positive training bags are selected and, within each of them, the most positive instance is determined based on kernel density estimation. The new instance is obtained via linear interpolation between these two elements.
2. **Negative instances**: the remainder of the bag is filled with negative instances, until the size of the new bag equals the average size of the training bags. The construction of these negative instances also uses the two positive bags selected in the previous step. The most negative instance is determined in the first one. Random negative instances are selected in the second bag and artificial instances are generated by means of interpolation.

Mera et al. stress that the difference with the BagSMOTE algorithm from [35] is that their oversampling step is more informative, because it determines the most positive instances within the positive bags and uses these to generate new positive instances. The experiments of [23] compare B-Bags with their earlier proposal B-Instances and with BagSMOTE, in combination with several multi-instance classifiers, on nine datasets. B-Bags has the highest AUC in four out of the nine datasets and the highest g value in five.

9.5 Customized Multi-instance Approaches

In this section, we discuss the second type of solutions to deal with class imbalance, namely those at the algorithm-level. These are multi-instance classifiers that incorporate some imbalance-resistant heuristics in their internal workings.

9.5.1 Cost-Sensitive Boosting Models

Apart from their preprocessing techniques discussed in Sect. 9.4, cost-sensitive multi-instance classification procedures are introduced by Wang et al. [35, 36] as well. Their algorithms are based on the AdaBoost.M1 boosting scheme [15]. In single-instance learning, this is an iterative method, which trains a classifier in each iteration and reweighs instances based on their classification outcome, to ensure that misclassified elements receive more attention in the next iteration. Its weight update formula is

$$D_{t+1}(i) = \frac{D_t(i)K_t(x_i, y_i)}{Z_t},$$

with

$$K_t(x_i, y_i) = \exp(-\alpha_t y_i h_t(x_i)). \tag{9.1}$$

In these expressions, t is the iteration number and Z_t a normalization factor to ensure that D_{t+1} is a probability distribution. The function h_t refers to a single-instance classifier and $\alpha_t \in \mathbb{R}$ to the coefficient that represents the weight of h_t in the final classification aggregation. As AdaBoost.M1 was proposed as a single-instance learning method, (x_i, y_i) refers to an instance x_i and its outcome y_i. AdaBoost.M1 does not distinguish between classes in these weight update formulas. The cost-sensitive boosting methods proposed in [35, 36] do make this distinction, by introducing class-dependent costs in (9.1). A cost is defined for each class, that is, the methods use one cost value for the positive class and one for the negative class. The ratio of these two values is set in favor of the minority class. As a result, relatively more effort is taken to correctly classify minority bags. The authors note that the real ratio between the class-wise misclassification costs is generally not available. They advise to use the imbalance ratio as cost ratio, as this value can be easily derived from the data. They propose four versions of their algorithm, differing in the places where the cost factors are introduced. Their proposals are similar to the single-instance cost-sensitive boosting algorithms from [30]. The methods are called Ab1, Ab2, Ab3, and Ab4 and there weight update formulas are

Ab1: $K_t(X_i, y_i) = \exp(-C_i\alpha_t y_i h_t(X_i))$
Ab2: $K_t(X_i, y_i) = C_i \exp(-\alpha_t y_i h_t(X_i))$
Ab3: $K_t(X_i, y_i) = C_i \exp(-C_i\alpha_t y_i h_t(X_i))$
Ab4: $K_t(X_i, y_i) = C_i^2 \exp(-C_i^2\alpha_t y_i h_t(X_i))$

In these formulas, X_i refers to a bag and y_i to its outcome. The function h_t corresponds to a multi-instance classifier. The value C_i is the cost associated with the bag X_i. It can take on only two values, either the cost for the positive class or that of the negative class, depending on the bag-label. Bags of the same class are automatically associated with the same cost.

The cost-sensitive boosting schemes are experimentally shown to outperform BagSMOTE in [35]. Based on their experimental work in [36], Wang et al. put Ab3

forward as best performing version among their proposed cost-sensitive boosting algorithms.

9.5.2 *Fuzzy Rough Multi-instance Classifiers*

In the recent contribution of Vluymans et al. [34], an algorithm-level solution to multi-instance class imbalance was proposed as an extension of the single-instance classifier from [27] that was developed for two-class imbalanced data. Two classifier families, one instance-based and one bag-based are developed. Both use fuzzy rough set theory [12], a mathematical concept that models vague and incomplete information.

To classify a new bag, these methods determine its membership degree to the fuzzy rough lower approximation of the two classes. This value is a number between 0 and 1. For a bag X and a class C, it expresses the degree to which the similarity of a training bag B with X implies the affinity of B with class C. When X has a high membership to the fuzzy rough lower approximation of C, training bags similar to X are likely to belong to class C. This information is used in the prediction step. An unseen bag is assigned to the class for which its membership degree to the lower approximation is highest.

The two classifier families in [34] differ from each other in the way they compute the lower approximation values for a bag. The instance-based methods first determine these values for the instances in the bag. These computations rely on a definition of similarity between instances, an affinity degree of instances with bags and of instances with classes. In a second phase, the instance-based values are aggregated to the bag level. The bag-based algorithms on the other hand directly derive the information from the bag as a whole, using an appropriate metric to measure the similarity between bags and the affinity of bags with classes.

Within the two families, classifiers differ from each other in the way they measure instance or bag similarity as well as in their aggregation procedures. The best performing representatives of the two families, referred to as FRI (Fuzzy Rough Instance-based multi-instance classifier) and FRB (Fuzzy Rough Bag-based multi-instance classifier) in [34], are experimentally shown to outperform the cost-sensitive boosting methods described in Sect. 9.5.1 as well as BagSMOTE in combination with the MITI classifier [6].

9.6 Experimental Analysis

In this section, we present an experimental comparison of methods dealing with multi-instance class imbalance. These experiments are performed on datasets coming from various application areas. We include both resampling methods and custom multi-instance classifiers. The experimental setup is specified in Sects. 9.6.1 and 9.6.2 presents the results.

Table 9.1 Description of the class imbalanced multi-instance datasets used in this comparison

Dataset	# Bags	# Inst	# Feat	IR
Bonds	160	3558	16	3.57
Chains	152	4630	24	4.63
Corel1	2000	7947	9	19
Corel2	2000	7947	9	19
Thio	193	26611	8	6.72
WIR-2	113	3423	298	4.38
WIR-5	113	3423	303	3.71

9.6.1 Setup

The datasets used in this experimental study are described in Table 9.1. We list the total number of bags and instances as well as the number of features. The degree of class imbalance is represented by the IR of the dataset. These datasets originate from different application domains. We use the same versions and partitions as in the experimental study of [34]. Bonds and Chains are bioinformatics datasets that originally appeared in [28] and were also used in the previous chapters. In this section, we use the imbalanced versions of [36]. Thio (Thioredoxin) is a bioinformatics dataset as well, while the two Corel datasets correspond to image recognition problems and were used in e.g., [9]. Finally, the two WIR datasets relate to the web index recommendation problem and were originally introduced in [45].

We use the fivefold cross validation procedure described in Sect. 1.4. In the comparison, we include the BagSMOTE (B-SMT), Bag_Oversampling (B-Over), and B-Bags resampling methods as well as all custom classifiers discussed in Sect. 9.5. We use the parameter settings recommended by the authors of the original proposals. BagSMOTE, Bag_Oversampling, and B-Bags are combined with the tree-based classifier MITI [6]. This classifier is used internally in the cost-sensitive boosting methods from Sect. 9.5.1 as well. We evaluate the performance of the classifiers by means of the AUC and g metrics.

9.6.2 Results and Discussion

In this section, we list the full results of the selected algorithms on all datasets and interpret them accordingly. We divide the main discussion in two parts, related to the evaluation by the AUC and the g value respectively. We also compare the performance of the resampling methods in combination with three different classifiers.

Table 9.2 Experimental AUC results for all methods

Dataset	B-SMT	B-Over	B-Bags	Ab1	Ab2	Ab3	Ab4	FRI	FRB
Bonds	0.7371	0.7680	0.6800	0.6857	0.7183	**0.8373**	0.7297	0.7345	0.6435
Chains	0.6692	0.6797	0.5726	0.6446	0.6970	0.7855	0.7736	**0.7913**	0.6081
Corel1	0.7797	0.6016	0.8074	0.7512	0.6931	0.8746	0.7960	**0.8751**	0.7469
Corel2	0.7392	0.6005	0.7437	0.7703	0.5197	0.7490	0.7131	**0.8444**	0.8211
Thio	0.5298	0.6169	0.4979	0.6571	0.5000	0.6437	0.6304	**0.7076**	0.6414
WIR-2	0.7893	0.7580	0.7547	0.8196	0.6674	0.8043	0.7384	0.8323	**0.8665**
WIR-5	0.6538	0.7011	0.7067	0.6695	0.5440	0.7278	0.7125	**0.8984**	0.8783
Mean	0.6997	0.6751	0.6804	0.7140	0.6199	0.7746	0.7277	**0.8120**	0.7437

9.6.2.1 Evaluation by AUC

The AUC values are listed in Table 9.2. For each dataset, we print the results of the best performing method in bold. The table shows that, on average, the FRI method from [34] attains the highest result. Furthermore, this method dominates the table by yielding the highest AUC value in five out of seven datasets. In the remaining two datasets, either Ab3 or FRB yield the best result.

Among the resampling methods, BagSMOTE gives the highest average value. It outperforms Bag_Oversampling, which was demonstrated in the original proposal [35] as well. Contrary to the observations in [23], B-Bags does not clearly outperform BagSMOTE. The former yields a higher AUC value than the latter in only three out of the seven datasets.

With respect to the custom learners for imbalanced multi-instance data, the first matter that is evident from Table 9.2 is that they generally provide better classification results than the resampling methods. Only the boosting method Ab2 is excluded from this observation, yielding an average AUC value considerably inferior to those of the three resampling algorithms. Our experiments confirm the finding of [36], that Ab3 is the best performing alternative for the cost-sensitive learners. Comparing the two methods from [34], FRI can be preferred over FRB. As noted above, FRI also stands out as best performing overall method when the classification performance is evaluated by the AUC.

The results of Table 9.2 are visually presented in Fig. 9.2. We have selected the two oversampling methods BagSMOTE and B-Bags and the classification methods Ab3 and FRI and plot their performance on all datasets.

9.6.2.2 Evaluation by g

The results for this evaluation can be found in Table 9.3 and Fig. 9.3. The conclusions are less clear-cut than for the AUC evaluation. The FRI method still yields the highest average result, but is the best performing algorithm in only two out of the

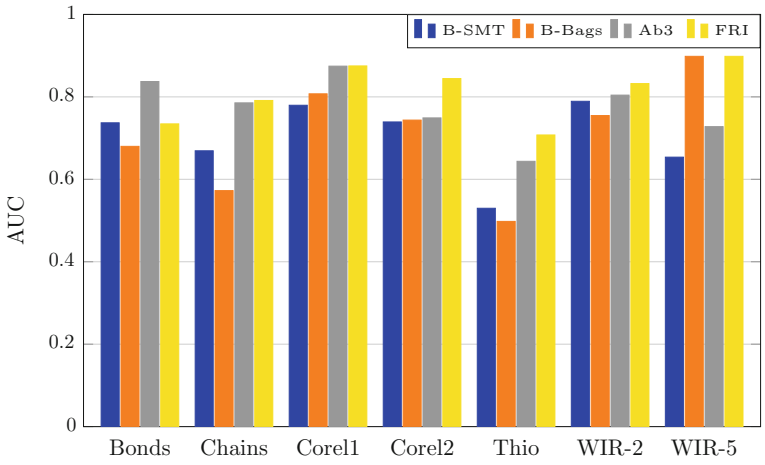

Fig. 9.2 Classification results of the four selected methods, measured with the AUC

Table 9.3 Experimental *g* results for all methods

Dataset	B-SMT	B-Over	B-Bags	Ab1	Ab2	Ab3	Ab4	FRI	FRB
Bonds	0.7026	0.7494	0.6197	0.4074	0.4743	**0.7715**	0.4071	0.6296	0.6583
Chains	0.5988	0.6228	0.4216	0.0000	0.2711	**0.7465**	0.0000	0.4163	0.4820
Corel1	0.7766	0.4808	**0.8074**	0.1995	0.2819	0.5358	0.0000	0.7846	0.6079
Corel2	0.7325	0.4876	0.7429	0.4527	0.0000	0.2115	0.0000	**0.8016**	0.7526
Thio	0.2440	0.5127	0.1852	0.2673	0.0000	0.0000	0.0000	**0.6612**	0.5521
WIR-2	0.7809	0.7563	0.7477	0.7323	0.3018	0.5898	0.0000	0.7198	**0.8140**
WIR-5	0.5914	0.6577	0.6616	0.4082	0.2041	0.6665	0.0000	0.7551	**0.7879**
Mean	0.6324	0.6096	0.5980	0.3525	0.2190	0.5031	0.0582	**0.6812**	0.6650

seven datasets. Both Ab3 and FRB are each dominant in two datasets as well. The oversampling method B-Bags attains the best result in the seventh dataset.

As for the AUC, we can conclude that the best results are generally obtained by custom classifiers rather than oversampling methods. However, we do note that the cost-sensitive boosting methods do not perform well. Ab3 is still the best version among them, but it does not perform at the same level as the resampling methods or FRI and FRB. Among the resampling methods, BagSMOTE attains the highest average result, followed by Bag_Oversampling. B-Bags performs best in three out of seven datasets, a number that is not sufficient to support the conclusion of [23] stating that this method can be preferred over the others. Naturally, as we fixed the classification algorithm (MITI) executed after the resampling methods, these observations may differ when another classifier is selected, as presented in the next paragraph.

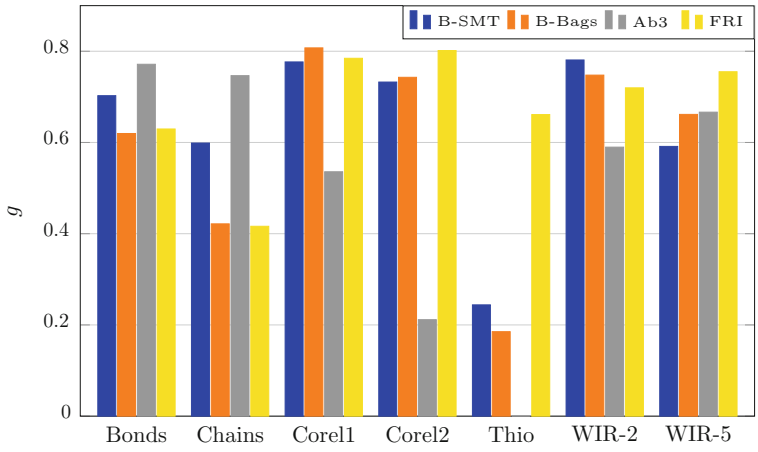

Fig. 9.3 Classification results of the four selected methods, measured with g

9.6.2.3 Resampling Methods

Resampling methods are performed in the preprocessing phase, which means that they are independent of the classification step. In particular, they can be combined with any multi-instance classifier and their effect on the performance of the latter may depend on the selected method. Some classifiers may benefit more from resampling than others.

In Table 9.4 we evaluate the performance of the three oversampling methods in conjunction with three different classifiers. The results for MITI were presented above. We also include the CitationKNN method from [40], using two references and four citers. The third method is the MILES algorithm with C4.5 from [9].

We compare the AUC and g values of these classifiers before (column 'None') and after preprocessing by any of the three resampling methods. The results for MITI were discussed above and we could put forward BagSMOTE as the preferred oversampling method for this classifier on this group of datasets. We also note the clear improvement in the two metrics of all resampling methods on the application of MITI without preprocessing. Clearly, this classifier greatly benefits from resampling. For CitationKNN, the AUC values of all resampling methods, as well as that of the classifier without preprocessing, are close together. For the evaluation by g on the other hand, the benefits of resampling are very clear. The highest g value is obtained by the simple Bag_Oversampling method. This method also yields the best results for MILES, for both evaluation metrics.

Table 9.4 Results of the resampling methods combined with different classifiers, taken as averages over the seven datasets

	AUC				*g*			
Classifier	None	B-SMT	B-Over	B-Bags	None	B-SMT	B-Over	B-Bags
MITI	0.5866	0.6997	0.6751	0.6804	0.4880	0.6324	0.6096	0.5980
CitationKNN	0.7082	0.7098	0.7084	0.7033	0.1430	0.2905	0.4238	0.2863
MILES	0.5702	0.5829	0.5937	0.5809	0.2666	0.2648	0.4154	0.2927

9.7 Summarizing Comments

In this chapter, we have discussed the phenomenon of class imbalance and the challenges it poses to both single-instance and multi-instance classification. The skewed distribution of the data observations among the classes inhibits the recognition of underrepresented classes. In single-instance learning, many solution methods to deal with class imbalance have been proposed over the past decades. A main distinction can be made between data-level and algorithm-level solutions. The former modify the dataset, e.g., by creating new minority class samples, and are independent of the classification step. The latter are custom classifiers, that use imbalance-resistant heuristics internally.

In MIL, comparatively less work has been done on this subject. We have described the existing methods in this chapter, including resampling methods and custom classifiers. We have compared these algorithms in an experimental study based on appropriate metrics for the setting of imbalanced classes. Our experiments indicate that, in imbalanced MIL, custom classifiers generally yield better results than resampling algorithms.

References

1. Abdi, L., Hashemi, S.: To combat multi-class imbalanced problems by means of over-sampling techniques. IEEE Trans. Knowl. Data Eng. **28**(1), 238–251 (2016)
2. Amores, J.: Multiple instance classification: review, taxonomy and comparative study. Artif. Intel. **201**, 81–105 (2013)
3. Bamber, D.: The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. J. Math. Psychol. **12**(4), 387–415 (1975)
4. Barua, S., Islam, M., Yao, X., Murase, K.: MWMOTE-majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans. Knowl. Data Eng. **26**(2), 405–425 (2014)
5. Batista, G., Prati, R., Monard, M.: A study of the behavior of several methods for balancing machine learning training data. ACM Sigkdd Explor. Newsl. **6**(1), 20–29 (2004)
6. Blockeel, H., Page, D., Srinivasan, A.: Multi-instance tree learning. In: De Raedt, L., Wrobel, S. (eds.) Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), pp. 57–64. ACM, New York (2005)

7. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intel. Res. **16**(1), 321–357 (2002)
8. Chen, Y.: An empirical study of a hybrid imbalanced-class DT-RST classification procedure to elucidate therapeutic effects in uremia patients. Med. Biol. Eng. Comput. 1–19 (2016)
9. Chen, Y., Bi, J., Wang, J.Z.: MILES: multiple-instance learning via embedded instance selection. IEEE Trans. Pattern Anal. **28**(12), 1931–1947 (2006)
10. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artif. Intel. **89**(1–2), 31–71 (1997)
11. Domingos, P.: Metacost: a general method for making classifiers cost-sensitive. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 155–164. ACM, New York (1999)
12. Dubois, D., Prade, H.: Rough fuzzy sets and fuzzy rough sets. Int. J. Gen. Syst. **17**(2–3), 191–209 (1990)
13. Fawcett, T.: An introduction to ROC analysis. Pattern Recognit. Lett. **27**(8), 861–874 (2006)
14. 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)
15. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Saitta, L. (ed.) Proceedings of the 13th International Conference on Machine Learning (ICML 1996), pp. 148–156. Morgan Kaufmann Publishers, San Francisco (1996)
16. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., Herrera, F.: A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans. Syst. Man Cybern. C **42**(4), 463–484 (2012)
17. Galar, M., Fernández, A., Barrenechea, E., Herrera, F.: EUSBoost: enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. Pattern Recognit. **46**(12), 3460–3471 (2013)
18. He, H., Garcia, E.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)
19. Kharbat, F., Bull, L., Odeh, M.: Mining breast cancer data with XCS. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 2066–2073. ACM, New York (2007)
20. Lee, Y., Hu, P., Cheng, T., Huang, T., Chuang, W.: A preclustering-based ensemble learning technique for acute appendicitis diagnoses. Artif. Intel. Med. **58**(2), 115–124 (2013)
21. 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)
22. Mena, L., Gonzalez, J.: Machine learning for imbalanced datasets: application in medical diagnostic. In: Sutcliffe, G., Goebel, R. (eds.) Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference (Flairs 2006), pp. 574–579. The AAAI Press, Menlo Park (2006)
23. Mera, C., Arrieta, J., Orozco-Alzate, M., Branch, J.: A bag oversampling approach for class imbalance in multiple instance learning. In: Parto, A., Kittler, J. (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 724–731. Springer, Switzerland (2015)
24. Mera, C., Orozco-Alzate, M., Branch, J.: Improving representation of the positive class in imbalanced multiple-instance learning. In: Campilho, A., Kamel, M. (eds.) Image Analysis and Recognition, pp. 266–273. Springer, Switzerland (2014)
25. Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Stat. **33**(3), 1065–1076 (1962)
26. Prati, R., Batista, G., Silva, D.: Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. Knowl. Inf. Syst. **45**(1), 247–270 (2015)
27. Ramentol, E., Vluymans, S., Verbiest, N., Caballero, Y., Bello, R., Cornelis, C., Herrera, F.: IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification. IEEE Trans. Fuzzy Syst. **23**(5), 1622–1637 (2015)

28. Reutemann, P.: Development of a propositionalization toolbox. Master thesis, Albert Ludwigs University of Freiburg, Germany (2004)
29. Sáez, J.A., Luengo, J., Stefanowski, J., Herrera, F.: SMOTE-IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. Inf. Sci. **291**, 184–203 (2015)
30. Sun, Y., Kamel, M., Wong, A., Wang, Y.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recognit. **40**(12), 3358–3378 (2007)
31. Sun, Y., Wong, A., Kamel, M.: Classification of imbalanced data: a review. Int. J. Pattern Recognit. **23**(4), 687–719 (2009)
32. Veropoulos, K., Campbell, C., Cristianini, N.: Controlling the sensitivity of support vector machines. In: Dean, T. (ed.) Proceedings of the 16th International Joint Conference on AI, pp. 55–60. Morgan Kaufmann Publishers, San Francisco (1999)
33. Vluymans, S.: Instance selection for imbalanced data. Master thesis, Ghent University, Belgium (2014)
34. Vluymans, S., Sánchez Tarragó, D., Saeys, Y., Cornelis, C., Herrera, F.: Fuzzy rough classifiers for class imbalanced multi-instance data. Pattern Recognit. **53**, 36–45 (2016)
35. Wang, X., Liu, X., Japkowicz, N., Matwin, S.: Resampling and cost-sensitive methods for imbalanced multi-instance learning. In: Wei, D., Washio, T., Xiong, H., Karypis, G., Thuraisingham, B., Cook, D., Wu, X. (eds.) Proceedings of the 2013 IEEE 13th International Conference on Data Mining Workshops (ICDMW), pp. 808–816. IEEE, Los Alamitos (2013)
36. Wang, X., Matwin, S., Japkowicz, N., Liu, X.: Cost-sensitive boosting algorithms for imbalanced multi-instance datasets. In: Zaïne, O., Zilles, S. (eds.) Advances in Artificial Intelligence, pp. 174–186. Springer, Berlin (2013)
37. Wang, X., Shao, H., Japkowicz, N., Matwin, S., Liu, X., Bourque, A., Nguyen, B.: Using SVM with adaptively asymmetric misclassification costs for mine-like objects detection. In: Wani, M., Khoshgoftaar, T., Zhu, X., Seliya, N. (eds.) Proceedings of the 11th International Conference on Machine Learning and Applications (ICMLA 2012), pp. 78–82. IEEE, Los Alamitos (2012)
38. Wang, S., Yao, X.: Diversity analysis on imbalanced data sets by using ensemble models. In: Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Data Mining (CIDM'09), pp. 324–331. IEEE, Los Alamitos (2009)
39. Wang, S., Yao, X.: Multiclass imbalance problems: Analysis and potential solutions. IEEE Trans. Syst. Man Cybern. B **42**(4), 1119–1130 (2012)
40. Wang, J., Zucker, J.: Solving multiple-instance problem: A lazy learning approach. In: Langley, P. (ed.) Proceedings of the 17th International Conference on Machine Learning (ICML 2000), pp. 1119–1125. Morgan Kaufmann Publishers, San Francisco (2000)
41. Yu, H., Ni, J., Zhao, J.: ACOSampling: an ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data. Neurocomputing **101**, 309–318 (2013)
42. Yu, H., Hong, S., Yang, X., Ni, J., Dan, Y., Qin, B.: Recognition of multiple imbalanced cancer types based on DNA microarray data using ensemble classifiers. BioMed Res. Int. **2013**, 1–13 (2013)
43. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: Wu, X., Tuzhilin, A., Shavlik, J. (eds.) Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), pp. 435–442. IEEE, Los Alamitos (2003)
44. Zhao, X., Li, X., Chen, L., Aihara, K.: Protein classification with imbalanced data. Proteins Struct. Funct. Bioinform. **70**(4), 1125–1132 (2008)
45. Zhou, Z., Jiang, K., Li, M.: Multi-instance learning based web mining. Appl. Intel. **22**(2), 135–147 (2005)