Pattern Recognition 48 (2015) 1623–1637

Contents lists available at ScienceDirect

Pattern Recognition

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

A novel ensemble method for classifying imbalanced data

Zhongbin Sun [a], Qinbao Song [a,*], Xiaoyan Zhu [a], Heli Sun [a], Baowen Xu [b], Yuming Zhou [b]

[a] Dept. of Computer Science & Technology, Xi'an Jiaotong University, China 710049
[b] Dept. of Computer Science & Technology, Nanjing University, China 210093

ARTICLE INFO

Article history:
Received 2 January 2014
Received in revised form
18 November 2014
Accepted 22 November 2014
Available online 29 November 2014

Keywords:
Imbalanced data
Classification
Ensemble learning

ABSTRACT

The class imbalance problems have been reported to severely hinder classification performance of many standard learning algorithms, and have attracted a great deal of attention from researchers of different fields. Therefore, a number of methods, such as sampling methods, cost-sensitive learning methods, and bagging and boosting based ensemble methods, have been proposed to solve these problems. However, these conventional class imbalance handling methods might suffer from the loss of potentially useful information, unexpected mistakes or increasing the likelihood of overfitting because they may alter the original data distribution. Thus we propose a novel ensemble method, which firstly converts an imbalanced data set into multiple balanced ones and then builds a number of classifiers on these multiple data with a specific classification algorithm. Finally, the classification results of these classifiers for new data are combined by a specific ensemble rule. In the empirical study, different class imbalance data handling methods including three conventional sampling methods, one cost-sensitive learning method, six Bagging and Boosting based ensemble methods, our previous method EM1vs1 and two fuzzy-rule based classification methods were compared with our method. The experimental results on 46 imbalanced data sets show that our proposed method is usually superior to the conventional imbalance data handling methods when solving the highly imbalanced problems.

1. Introduction

Class imbalance data refers to at least one of its classes is usually outnumbered by the other classes. The class imbalance problems have been reported to occur in a wide variety of real-world domains, such as facial age estimation [1], detecting oil spills from satellite images [2], anomaly detection [3], identifying fraudulent credit card transactions [4], software defect prediction [5], and image annotation [6]. Therefore, researchers have paid much attention to the class imbalance problems, and several thematic workshops and conferences were held, such as the Association for the Advancement of Artificial Intelligence (AAAI) 2000 [7], the International Conference on Machine Learning (ICML) 2003 [8], and the ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations 2004 [9].

For a binary class imbalance problem, the instances are usually categorized into majority and minority classes. Generally speaking, the minority class usually represents a concept with greater interest than the majority class. However, it is often outnumbered by the majority class, and sometimes this scenario may be very severe. As most traditional classification algorithms, such as decision trees [10–12], *k*-nearest neighbors [13,14], and RIPPER [15,16], tend to generate models which maximize the overall classification accuracy, and the minority class is usually ignored [17–19]. For example, for a data set where only 1% of the instances belong to the minority class, even if a model classifies all instances as the majority class, it still achieves an overall accuracy of 99%. However, the minority class instances, which we want to accurately classify, are all misclassified by this model though it achieves a very high accuracy. Therefore, a number of methods have been proposed to deal with the imbalanced binary classification problems.

One of the most popular methods for solving the class imbalance problems is sampling [20–23]. However, the most used under-sampling and over-sampling methods alter the original class distribution of imbalanced data by eliminating the majority class instances or increasing the minority class instances. In addition, cost-sensitive learning is also employed to solve the class imbalance problems [24–26]. This method assigns different cost of misclassification errors for different classes, generally high cost for the minority class and low cost for the majority class as the minority class is usually more interesting. Moreover, Bagging [27] and Boosting [28] based ensemble methods are another widely used methods to deal with imbalanced problems [15,16,29,30].

We argue that the above mentioned methods might encounter some unexpected problems when employed to solve the class imbalance problems. For instance, under-sampling methods might

* Corresponding author. Tel.: +86(0)29 82668645; fax: +86(0)29 82668971.
E-mail address: qbsong@mail.xjtu.edu.cn (Q. Song).

http://dx.doi.org/10.1016/j.patcog.2014.11.014
0031-3203/

abandon some potentially useful data which could be very important for a learning process, while over-sampling methods may increase the likelihood of overfitting in the induction process. Furthermore, Bagging and Boosting based ensemble methods may eliminate some useful data as they use sampling methods to obtain balanced data in each of their iteration procedure, and they may suffer from overfitting as well. In a word, these two kinds of methods both may alter the original class distribution of imbalanced data by adding minority class instances or deleting majority class instances. Moreover, for the cost-sensitive learning methods, it is difficult to obtain the accurate misclassification cost, and the different misclassification cost might result in different induction results. So the classification results are not stable.

This paper introduces a novel ensemble method for addressing binary-class imbalance problems. Our proposed method handles the class imbalance problems by converting a imbalanced binary learning process into multiple balanced learning processes, which does not make use of introducing minority class instances or removing majority class instances to get away from the imbalanced data. The proposed method firstly converts the imbalanced data into multiple balanced data using the data balancing methods random splitting (SplitBal) or clustering (ClusterBal). Then multiple classifiers could be built from these balanced data with a specific classification algorithm. Finally, we use a specific ensemble rule to combine the classification results of these classifiers for the new data. Five ensemble rules Max, Min, Product, Majority Vote, and Sum from [31] and another five improved ensemble rules MaxDistance, MinDistance, ProDistance, MajDistance, and SumDistance proposed by us are studied.

In the empirical study, the performance of six different types of classification algorithms Naive Bayes [32], C4.5 [33], RIPPER [34], Random Forest [35], SMO [36], and IBK [37] were evaluated over 46 highly imbalanced data sets. We first studied which ensemble rule performs better for the two data balancing methods ClusterBal and SplitBal. The experimental results show that for both of the two data balancing methods, ensemble rule MaxDistance performs better than other nine ensemble rules. Then we studied which combination of data balancing method and ensemble rule performs better with these six classification algorithms, and found that the two best combinations ClusterBal+MaxDistance and SplitBal+MaxDistance perform differently for different classification algorithms. After that, we compared our method with the conventional external imbalance data handling methods, including random under-sampling [11], random over-sampling [11], SMOTE [22], MetaCost [38], Bagging [27], Boosting [28], EasyEnsemble [19], SMOTEBoost [15], RUSBoost [16], UnderBagging [39] and our previous method EM1vs1 [40] which deals with the class imbalance problems in software defect prediction. Finally we compared our method with two representative internal imbalance data handling methods Chi3-GTS and Chi5-GTS [41]. The experimental results show that our method is usually more effective than these conventional external and internal imbalance data handling methods over the 46 highly imbalanced data sets.

The remainder part of this paper is organized as follows: Section 2 introduces the related work. In Section 3, we present our proposed method. Section 4 is devoted to the experiments, discusses the detailed experimental setup and analyzes the corresponding results. Finally, in Section 5 we make our concluding remarks.

2. Related work

The imbalance problem is one of the top 10 challenging problems in data mining [42]. It occurs in many real-world domains [40,43,44], and has drawn a significant amount of attention in the fields of data mining and machine learning. Apart from the class imbalance characteristic, there might be other data characteristics that could affect the performance of traditional classification algorithms for imbalanced problems [45–47], such as dataset shift [48], class overlapping [49], and small disjuncts [50]. However, in this study we focus on handling the class imbalance characteristic as it is the primary and intuitive data characteristic in imbalanced data, and other unintuitive data characteristics that may affect the performance of imbalanced data learning will be our future study.

Up to now, a variety of methods has been proposed for dealing with class imbalance problems. These methods can be broadly divided into two categories, namely external methods and internal methods [51]. Internal methods modify existing classification algorithms for reducing their sensitiveness to class imbalance [41,52–54], while external methods preprocess the training data to make them balanced. As the external methods have the advantage of independence on the underlying classification algorithms and our method could be regarded as a kind of external method, in this section we will pay more attention to the external methods, including the sampling methods [11,13,21,55–59], Bagging and Boosting based ensemble methods [16,19,60–64], and cost-sensitive learning methods [18,24–26,65,66].

Sampling methods, which are employed to balance class distribution in imbalanced data sets, can be divided into two groups: under-sampling and over-sampling. The under-sampling methods eliminate the majority class instances while the over-sampling methods increase the minority class instances for the purpose of obtaining a desirable rate of class distribution. Japkowicz [55] mainly discussed two strategies: under-sampling and resampling. She noted that both the sampling approaches were effective, and furthermore she also observed that using the sophisticated sampling techniques does not provide any clear advantage in solving the class imbalance problem. Moreover, Mani [13] also indicated that the random under-sampling strategy usually outperformed some other complicated under-sampling strategies. Kubat et al. [56] proposed a heuristic under-sampling strategy named One-Sided Selection for eliminating the majority class instances that are either borderline or noisy. In addition to the under-sampling strategies, the over-sampling strategies are also widely used for dealing with the class imbalance problem. Chawla et al. [22] proposed an over-sampling approach in which the minority class is over-sampled by creating "synthetic" instances rather than by over-sampling with replacement. Han et al. [67] proposed the borderline-SMOTE to over-sample the minority class instances near the borderline. Batista et al. [11] and Xie at al. [68] both showed that over-sampling usually perform better than under-sampling. Estabrooks et al. [58] and Barandela et al. [69] both suggested that a combination of over-sampling and under-sampling might be more effective to solve the class imbalance problem. However, we argue that the sampling methods alter the original data class distribution of imbalanced data and then lead to some unexpected mistakes. For example, over-sampling might lead to overfitting and under-sampling may drop some potentially useful information.

Apart from the sampling strategies, Bagging and Boosting based ensemble methods have also been widely applied to the class imbalance problem. Seiffert et al. [63] conducted a comprehensive study comparing sampling methods with boosting for improving the performance of decision trees model built for identifying the software defective modules. Their results showed that sampling methods were effective in improving the performance of such models while boosting outperformed even the best data sampling methods. Chawla et al. [15] proposed a novel approach SMOTEBoost for learning from imbalanced datasets on the basis of the SMOTE algorithm and the boosting procedure. Seiffert et al. [16] presented a different hybrid ensemble methods named RUSBoost, which combined the random under-sampling strategy with the boosting procedure. Their empirical results showed that RUSBoost performed comparably to SMOTEBoost while being a faster and simpler technique. Liu et al. [19] proposed the

EasyEnsemble method which combines bagging, random under-sampling and AdaBoost [28]. However, we argue that for each iteration of original bagging and boosting methods, it may still suffer from the class imbalance problem as the sampled subset in a given iteration has the similar class distribution with the original data set. In addition, for the Bagging and Boosting based ensemble methods which usually combine the sampling methods with Bagging and Boosting procedure, they may alter the original data class distribution as they use sampling methods to increase the minority instances or eliminate the majority class instances.

Changing class distribution is not the only way to improve classifier performance when learning from the imbalanced data. A different approach to handling the class imbalance problem is to define fixed and unequal misclassification costs between classes. Generally, data mining and machine learning algorithms assume that all errors in classification are equally important. However, the cost of misclassification errors is often higher when the errors are associated with the minority class in contrast with the majority class. Thus the cost-sensitive learning algorithm focuses on the costs associated with the misclassified instances [70]. Weiss et al. [24] found that the cost-sensitive learning method performs as well as under-sampling and over-sampling methods and additionally they found that no sampling method is a clear winner over the other. Seiffert et al. [26] conducted an empirical study for comparing four data sampling methods and two cost-sensitive learning strategies. However, they noted that random under-sampling method tended to perform best of all the used methods. Ting [65] proposed an instance-weighting method to construct cost-sensitive trees. He concluded that the proposed method was simpler and more effective. Furthermore, another conventional cost-sensitive learning method is the Metacost algorithm [38], which employs bagging to determine optimal class labels for training data, relabeling instances accordingly. However, it may be quite difficult to determine the accurate misclassification cost when applying the cost-sensitive learning method to solve the imbalanced problems.

All these three kinds of external methods mentioned above deal with the class imbalance data characteristic in the imbalanced problem. Our proposed method addresses the potential shortcomings of these conventional imbalance data handling methods mentioned above by converting the imbalanced binary problem into multiple balanced problems that no longer suffer from the class-imbalance curse and does not alter the original data class distribution, thus it is quite different from the conventional imbalance data handling methods. In addition, the newly proposed method differs greatly from our previous ensemble method EM1vs1 for solving the class imbalance problem in software defect prediction [40]. EM1vs1 firstly splits the majority class into several non-overlapping subsets and each subset is then assigned a new class label, which has the similar number of instances to the minority class. Thus the imbalanced binary-class data is converted into balanced multi-class data. Then we use a popular coding scheme one-against-one [71] to learn the multi-class data by building individual classifier for each pair of classes. Finally the classification results of all the built classifiers are combined with the Hastie and Tibshirani's method [72]. Thus EM1vs1 and the newly proposed method are two different problem solution schemes: EM1vs1 turns the imbalanced binary-class learning problem into balanced multi-class learning problem while the newly proposed method turns the imbalanced binary-class learning problem into multiple balanced binary-class learning problems.

3. Proposed method

3.1. Overview of the method

Many traditional classification algorithms have shown poor performance for the class imbalance problems [11,13,16]. The classifiers built with these algorithms for such problems usually ignore the minority class as these classification algorithms tend to maximize the overall classification accuracy [10,18]. Thus they may be inaccurate for the class imbalance problems. Bagging is a kind of ensemble learning method that could solve this problem as the ensemble learning could improve the performance if base classifiers are accurate and diverse [27]. However, for each base classifier in the Bagging, it is also inaccurate as it still deals with the class imbalance data which is sampled proportionally from the original imbalanced data. Thus, some special sampling methods are combined with Bagging to solve this problem, including UnderBagging [39], OverBagging and SMOTEBagging [73]. These Bagging based methods first use sampling to balance data for each base classifier, and then combine the classification results of these base classifiers with specific ensemble rules. However, these Bagging based methods may suffer from the problems that sampling methods always encounter, such as altering the original data distribution and overfitting. Furthermore their ensemble rules usually lose sight of the relationship between new data and historical data. That is, new data can be more closer to a specific class of historical data in spatial distribution. Therefore for the purpose of overcoming the shortcomings of these Bagging based ensemble methods, we propose a new ensemble method for dealing with class imbalance problems.

Our proposed ensemble method addresses a class imbalance problem by converting it into several balanced problems, which includes three components: *Data Balancing*, *Modeling* and *Classifying*. Fig. 1 shows the details.

In our method, the majority class instances are firstly divided into several bins. Each bin has the similar number of instances to that of the minority class, and is combined with the instances of minority class. So several balanced data sets are obtained (*Data Balancing*). Afterwards, each balanced data set is employed to build a binary classifier with a specific learning algorithm (*Modeling*). Finally, these binary classifiers are combined into an ensemble classifier to classify new data (*Classifying*). Since the *Modeling* component is nothing but directly apply a specific classification algorithm to each of the balanced data, we will particularly introduce the two components *Data Balancing* and *Classifying* in the following sections.

3.2. Data balancing

Imbalanced binary data has been certified to jeopardize the traditional classification learning algorithms, and the minority class is usually misclassified by such classification models [58]. Thus methods to balance the skewed data, such as under-sampling and over-sampling, have been seriously taken into consideration. However, under-sampling may drop some potentially useful information [74] while over-sampling is likely to lead overfitting [69]. Therefore, it is reasonable to convert imbalanced data into several balanced ones without introducing extra information or removing the original information.

As is known to all, in an imbalanced data set, the majority class instances usually outnumber the minority class instances. If the number of majority class instances can be reduced and then combined with the minority class instances, balanced data will be obtained. This requires us to split the majority class instances into several bins, and each bin has similar number of instances with minority class. Considering that some instances of the majority class may be more similar than others [10,12], so it is rational to split the majority class into several subclasses with fewer number of instances. For this purpose, cluster analysis methods can be used. When using clustering as the data balancing method, a specific clustering algorithm is firstly applied to the majority class instances and multiple clusters are then obtained. Finally, each of these clusters is combined with minority class, so a

new balanced data is constructed. By this way, we obtain multiple balanced data sets. However, clustering algorithms cannot guarantee each cluster has similar number of instances, so sometimes they still result in new imbalanced data.

On the other hand, think of as the members of a class they should share some properties with each other, and the difference among them is much less. This means if we split the majority class instances into several bins, each instance can be put into anyone bin. Now that this is the case, we can obtain bins whose number of instances are comparable to that of the minority class. So no bins with too many or too few instances will occur, and the shortcoming of clustering method is avoided. In our proposed method, this is achieved by randomly splitting the majority class instances into multiple bins, where each bin has almost the same number of instances as that of the minority class. Then each bin is combined with the instances of minority class to construct a new balanced data. Finally, multiple balanced data are obtained.

In the following sections, we refer to the clustering and the random splitting based data balancing methods as ClusterBal and SplitBal, respectively.

3.3. Classifying

After *Modeling*, multiple classifiers could be built with the multiple balanced data obtained from *Data Balancing*. Thus, for a new problem, we will obtain individual classification result from each classifier. Obviously, it is required to combine these classification results together. Kittler et al. has proposed five ensemble rules for combining the multiple classification results of different classifiers, including Max Rule, Min Rule, Product Rule, Majority Vote Rule and Sum Rule [31]. For the purpose of conveniently introducing these ensemble rules, we make the following assumptions.

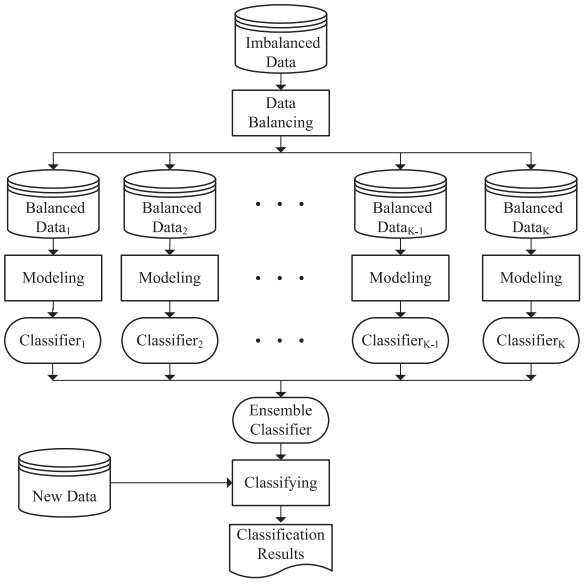

Fig. 1. Proposed ensemble method for handling imbalanced data.

Suppose that there are K binary-class data and for each data, the class labels are C_1 and C_2. Then we could obtain K classifiers by applying a specific classification algorithm to each of these K binary-class data. For the ith classifier ($1 \le i \le K$), it classifies the new data as C_1 with the probability P_{i1} while as C_2 with the probability P_{i2}. Moreover, R_1 and R_2 represent the final ensemble result for the class C_1 and C_2, respectively. Table 1 shows the detailed ensemble strategies and descriptions for the five ensemble rules. For more details about these five ensemble rules, refer to [31].

However, we argue that these five ensemble rules only consider the classification results of constructed classifiers while ignoring the relationship between new data and training data. The relationship is that new data seems more likely to be categorized into the class whose instances are more similar to the new data. Tahir et al. [75] employed a weighting function based on the neighbor distance to learn the weights of multi-label classifiers and we also consider using such distance-based weighting mechanism. Therefore in our current study, we proposed five new ensemble rules on the basis of combining the aforementioned five ensemble rules with the relevance between new data and the training data distribution, in which the relevance is represented as the negative correlation function of distance. In fact, such negative correlation function of distance could be regarded as a weighting function to the classification probability in Table 1.

For the purpose of introducing our ensemble rules, in addition to the aforementioned assumptions, we will use D_{ij} ($1 \le i \le K$, $1 \le j \le 2$) to represent the average distance between new data and the data with class label C_j in the ith data. Table 2 shows the detailed ensemble strategies and descriptions for the our proposed five ensemble rules. Note that for the purpose of being safe from the distance with value 0 in the denominator, we have added the distance with a value of 1. What's more, before computing the distance between two instances, we suggest the researchers that normalization should be employed to deal with each individual feature of the datasets as the features of the datasets may be measured on different scales.

Finally, given the final classification result R_1 and R_2 obtained with the ensemble rules in Table 1 and 2, the new data is classified as C_1 if $R_1 \ge R_2$, otherwise C_2.

Our proposed method is different from the UnderBagging method [39]. The UnderBagging method firstly uses sampling (with or without replacement) from the majority class for constructing several subsets, with each subset having similar size to the minority class. Then classifiers trained with each majority class subset and

Table 1
The strategies and descriptions for Kittler ensemble rules.

Rule	Strategy	Description
Max	$R_1 = \arg\max_{1 \le i \le K} P_{i1}$, $R_2 = \arg\max_{1 \le i \le K} P_{i2}$	Use the maximum classification probability of these K classifiers for each class label
Min	$R_1 = \arg\min_{1 \le i \le K} P_{i1}$, $R_2 = \arg\min_{1 \le i \le K} P_{i2}$	Use the minimum classification probability of these K classifiers for each class label
Product	$R_1 = \prod_{i=1}^{K} P_{i1}$, $R_2 = \prod_{i=1}^{K} P_{i2}$	Use the product of classification probability of these K classifiers for each class label
Majority vote	$R_1 = \sum_{i=1}^{K} f(P_{i1}, P_{i2})$, $R_2 = \sum_{i=1}^{K} f(P_{i2}, P_{i1})$	For the ith classifier, if $P_{i1} \ge P_{i2}$, class C_1 gets a vote, if $P_{i2} \ge P_{i1}$, class C_2 gets a vote
Sum	$R_1 = \sum_{i=1}^{K} P_{i1}$, $R_2 = \sum_{i=1}^{K} P_{i2}$	Use the summation of classification probability of these K classifiers for each class label

The function f(x,y) is defined as follows:

$$f(x,y) = \begin{cases} 1 & x \ge y \\ 0 & x < y \end{cases} \quad (1)$$

Table 2
The strategies and descriptions for our five ensemble rules.

Rule	Strategy	Description
MaxDistance	$R_1 = \arg\max_{1 \le i \le K} \frac{P_{i1}}{D_{i1}+1}$, $R_2 = \arg\max_{1 \le i \le K} \frac{P_{i2}}{D_{i2}+1}$	Use the inverse of average distance to adjust the Max Rule
MinDistance	$R_1 = \arg\min_{1 \le i \le K} \frac{P_{i1}}{D_{i1}+1}$, $R_2 = \arg\min_{1 \le i \le K} \frac{P_{i2}}{D_{i2}+1}$	Use the inverse of average distance to adjust the Min Rule
ProDistance	$R_1 = \prod_{i=1}^{K} \frac{P_{i1}}{D_{i1}+1}$, $R_2 = \prod_{i=1}^{K} \frac{P_{i2}}{D_{i2}+1}$	Use the inverse of average distance to adjust the Product Rule
MajDistance	$R_1 = \sum_{i=1}^{K} \frac{f(P_{i1},P_{i2})}{D_{i1}+1}$, $R_2 = \sum_{i=1}^{K} \frac{f(P_{i2},P_{i1})}{D_{i2}+1}$	Use the inverse of average distance to adjust the Majority Vote Rule
SumDistance	$R_1 = \sum_{i=1}^{K} \frac{P_{i1}}{D_{i1}+1}$, $R_2 = \sum_{i=1}^{K} \frac{P_{i2}}{D_{i2}+1}$	Use the inverse of average distance to adjust the Sum Rule

The function f(x,y) is defined in Table 1.

the minority class are combined with the majority vote ensemble rule. The differences between our method and UnderBagging are obvious: (1) Different data balancing method. We consider two data balancing methods, namely ClusterBal and SplitBal. ClusterBal is apparently different from the sampling data balancing method in UnderBagging. In addition, SplitBal is more simpler and faster than sampling as it does not need to use random seed to sample the instance one by one. (2) Different ensemble rules. UnderBagging only use the majority vote rule to combine the built classifiers, while we not only consider the five ensemble rules (including the majority vote rule) mentioned in [31] but also improve these five ensemble rules (see Table 2).

4. Empirical study

4.1. Data sets

In the present study, we have employed 46 highly imbalanced binary data sets from Keel data set repository [76,77]. These 46 data sets have different number of instances and attributes, and furthermore they also differ in class imbalance ratio (IR[1]).

Table 3 summarizes the properties of each selected imbalanced data sets, including the number of total attributes (# Attr.), the number of total instances (# Ins.), the number of minority class instances (# Min.), the number of majority class instances (# Maj.) and the imbalance ratio (IR). Among the 46 used imbalanced data sets, some are originally binary while others are artificially created from multi-class data sets with the union of one or more classes labeled as the minority (positive) class while the union of the remaining classes labeled as the majority (negative) class. For more details about the employed data sets, refer to url: http://sci2s.ugr.es/keel/imbalanced.php.

4.2. Experimental setup

The experimental study was conducted using the 10-fold cross-validation strategy. That is, each data set was divided into ten folds, and each fold has the similar number of instances. Then for each fold, a learning algorithm was trained on the remaining nine folds and then tested on the current fold. For the purpose of obtaining stable and reliable results, the 10-fold cross-validation strategy was repeated 10 times and each time the ordering of instances was shuffled.

Six different kinds of classification algorithms, including Naive Bayes [32], C4.5 [33], RIPPER [34], Random Forest [35], SMO [36] and IBK [37], were selected as the base classifiers. All these six classification algorithms have been implemented in the Weka learning environment [78]. Default parameters in the Weka software were used in our present study for all the six classification algorithms except the IBK. Particularly, K was set as 5 for the IBK classification algorithm.

The area under the ROC (Receiver Operating Characteristic) curve (AUC) [79,80] was used as our measure of classification performance, as it has been widely used to evaluate the performance of classifiers on the imbalanced data [16,19,23,81]. Two other popular performance metrics for imbalance data learning G-Means and F-Measure were not selected in our present study. He et al. [82] categorize these two metrics into threshold metrics and an important disadvantage of all the threshold metrics assume full knowledge of the conditions under which the classifiers will be deployed. In particular, they should be appointed with an accurate threshold for a special sample distribution of training data and thus these two metrics suffer from the subjective factors. However, AUC is able to overcome this issue as it is insensitive to changes in sample distribution [12,83] and so it is objective. Therefore we select the objective AUC instead of the subjective G-Means and F-Measure as our performance metric for learning from the 46 imbalanced data with different sample distribution.

4.3. Experimental design

The present study consists of six investigations. The first investigation is to explore which ensemble rule performs best with the data balancing method ClusterBal and the second investigation is to explore which ensemble rule performs best with the data balancing method SplitBal. On the basis of the analysis of aforementioned two investigations, the third investigation explores which combination of data balancing method and ensemble rule performs best across all the employed classification algorithms. Then the fourth investigation compares our method with the conventional external imbalance data handling methods when dealing with the imbalanced binary problems using different types of classification algorithms. The five investigation is conducted to compare our method with the internal methods for dealing with the class imbalance problem. Finally, the last investigation is to validate whether the added value 1 to distance in our ensemble rules is reasonable.

(1) *Investigation 1: which ensemble rule performs best with ClusterBal?* For the data balancing method ClusterBal, the simple K-Means clustering algorithm was employed, in which K is set as the ratio of majority class instances over the minority class instances. Our five ensemble rules are employed, including MaxDistance, MinDistance, ProDistance, MajDistance and SumDistance, were compared with the five ensemble rules proposed by Kittler et al. [31], including Max, Min, Product, Majority Vote and Sum.

(2) *Investigation 2: which ensemble rule performs best with SplitBal?* For the data balancing method SplitBal, the number of split bins is set as the ratio of majority class instances over the minority class instances. Additionally, the ten ensemble rules (including our five and Kittler's five) were compared as well.

(3) *Investigation 3: which combination of data balancing method and ensemble rule performs best?* In this investigation, the results from former two investigations are utilized. The first investigation

[1] $\mathrm{IR} = \frac{number\ of\ majority\ class\ instances}{number\ of\ minority\ class\ instances}$

Table 3
Statistic summary of the 46 highly imbalanced data sets in experimental study.

ID	Data	# Attr.	# Ins.	# Min.	# Maj.	IR	ID	Data	# Attr.	# Ins.	# Min.	# Maj.	IR
1	yeast3	9	1484	163	1321	8.10	24	ecoli01vs5	7	240	20	220	11.00
2	ecoli3	8	336	35	301	8.60	25	glass06vs5	10	108	9	99	11.00
3	pageblocks0	11	5472	559	4913	8.79	26	glass0146vs2	10	205	17	188	11.06
4	ecoli034vs5	8	200	20	180	9.00	27	glass2	10	214	17	197	11.59
5	yeast2vs4	9	514	51	463	9.08	28	ecoli0147vs56	7	332	25	307	12.28
6	ecoli067vs35	8	222	22	200	9.09	29	cleveland0vs4	14	177	13	164	12.62
7	ecoli0234vs5	8	202	20	182	9.10	30	ecoli0146vs5	7	280	20	260	13.00
8	glass015vs2	10	172	17	155	9.12	31	shuttlec0vsc4	10	1829	123	1706	13.87
9	yeast0359vs78	9	506	50	456	9.12	32	yeast1vs7	8	459	30	429	14.30
10	yeast0256vs3789	9	1004	99	905	9.14	33	glass4	10	214	13	201	15.46
11	yeast02579vs368	9	1004	99	905	9.14	34	ecoli4	8	336	20	316	15.80
12	ecoli046vs5	7	203	20	183	9.15	35	pageblocks13vs4	11	472	28	444	15.86
13	ecoli01vs235	8	244	24	220	9.17	36	abalone918	9	731	42	689	16.40
14	ecoli0267vs35	8	224	22	202	9.18	37	glass016vs5	10	184	9	175	19.44
15	glass04vs5	10	92	9	83	9.22	38	shuttlec2vsc4	10	129	6	123	20.50
16	ecoli0346vs5	8	205	20	185	9.25	39	yeast1458vs7	9	693	30	663	22.10
17	ecoli0347vs56	8	257	25	232	9.28	40	glass5	10	214	9	205	22.78
18	yeast05679vs4	9	528	51	477	9.35	41	yeast2vs8	9	482	20	462	23.10
19	vowel0	14	988	90	898	9.98	42	yeast4	9	1484	51	1433	28.10
20	ecoli067vs5	7	220	20	200	10.00	43	yeast1289vs7	9	947	30	917	30.57
21	glass016vs2	10	192	17	175	10.29	44	yeast5	9	1484	44	1440	32.73
22	ecoli0147vs2356	8	336	29	307	10.59	45	ecoli0137vs26	8	281	7	274	39.14
23	led7digit02456789vs1	8	443	37	406	10.97	46	yeast6	9	1484	35	1449	41.40

obtains the best ensemble rule with ClusterBal while the second investigation achieves the best ensemble rule with SplitBal. Then we analyzed the results of these two combinations over the six employed classification algorithms to explore which combination of data balancing method and ensemble rule performs best.

(4) *Investigation 4: is our class imbalance data classification method more effective than the external methods?* This investigation was used to explore whether our proposed method is really effective or not by comparing it with 10 conventional external imbalance data handling methods.

Three sampling methods, including random under-sampling (RUS), random over-sampling (ROS) and synthetic minority over-sampling technique (SMOTE) were employed in the present study. In addition, we employed the popular MetaCost [38] algorithm as the cost sensitive learning method. Furthermore, Bagging [27], Boosting [28], and four popular Bagging and Boosting based ensemble methods EasyEnsemble [19], RUSBoost [16], SMOTE-Boost [15] and UnderBagging [39] were also compared in the experiment. Moreover, our previous method EM1vs1 [40] for dealing with the class imbalance problem in software defect prediction was also compared in the present study.

Note that we refer to results of applying the six classification algorithms on the original imbalanced binary data as Orig in the following sections.

(5) *Investigation 5: is our class imbalance data classification method more effective than the internal methods?* Investigation 4 was conducted to validate the effectiveness of our proposed method for dealing with the imbalanced problems by comparing it with the conventional external methods. However, we also need to compare our method with the internal imbalance data handling methods, and thus this investigation was conducted for this purpose. As our method could be regarded as a kind of external method, in this investigation we will only select a few representative internal methods for comparison.

Fernández et al. [41] has improved the behavior of fuzzy rule based classification method for the class imbalance problems by a means of tuning step. Particularly, the 2-tuples based genetic tuning approach was adapted by them for enhancing the performance of fuzzy models. In their study, two learning methods to generate the rule base for the fuzzy rule based classification system were employed, including Chi et al.'s rule generation (Chi) [84] and Ishbuchi and Yamamoto's Fuzzy Hybrid Genetic Based Machine Learning (FH-GBML) [85]. In this section, we compare our proposed method using different classification algorithms with the Chi methods, including Chi3-GTS and Chi5-GTS in [41].

(6) *Investigation 6: is the value 1 added to distance in our ensemble rules reasonable?* Strategies of the five ensemble rules in Table 2 could be simplified into the following form $R=f(P/(D+Const))$, in which R represents the final ensemble results, P stands for the classification probability and D stands for the distance. In addition, the *Const* stands for a constant value specified by the researchers. In current study, we have set *Const* as 1 as many other researchers do to avoid the D with value 0 in denominator. This investigation is conducted to validate whether the *Const* with value 1 in our ensemble rules is reasonable.

In order to conduct such investigation, we intend to perform a series of experiments with various *Const* values in our ensemble rules. Specially, we have respectively selected five different values (including 0.01, 0.1, 1, 10 and 100) as the *Const* value in our ensemble rules.

4.4. Experimental results and analysis

(1) *Results for ensemble rules with ClusterBal.* This investigation explores which ensemble rule performs best with the data balancing method ClusterBal for the ensembles with the six classification algorithms as base classifiers, respectively. Due to space limitation, in the following we will only show the average AUC values of the 46 imbalanced data for each specific classification algorithm. Table 4 shows the average AUC values for each classification algorithm while using different ensemble rules with ClusterBal. Note that for each classification algorithm, the best average AUC value corresponding to a specific ensemble rule is highlighted in boldface.

From Table 4, we observe that the ensemble rule MaxDistance ranks first for five classification algorithms and the ensemble rule MajDistance ranks first for one classification algorithms. For the purpose of providing readers a more explicit picture about the performance difference of different ensemble rules, we rank the 10 different ensemble rules with ClusterBal for each classification algorithm according to the average AUC values of the 46 imbalanced data. Then we summarize the ranks of each ensemble rule

under the six classification algorithms and provide the final rank. Table 5 shows the details.

From Table 5, we observe: (i) For the five ensemble rules proposed by Kittler et al., Sum performs better than the other four ensemble rules, which is identical to the experimental results of [31]. (ii) Four of our proposed rules, including MaxDistance, MinDistance, MajDistance and SumDistance, perform better than their corresponding ensemble rules proposed by Kittler et al. [31]. In addition, our ProDistance ensemble rule ranks equally to its corresponding ensemble rule Product. This means that when using the data balancing method ClusterBal, our ensemble rules are more effective than those of Kittler. (iii) Our proposed ensemble rule MaxDistance performs best among all the 10 ensemble rules with the data balancing method ClusterBal.

(2) *Results for ensemble rules with SplitBal.* This investigation explores which ensemble rule performs best with the data balancing method SplitBal for the ensembles with the six classification algorithms as base classifiers, respectively. Table 6 shows the average AUC values for each classification algorithm while using different ensemble rules with SplitBal.

From Table 6, we observe that for the data balancing method SplitBal, MaxDistance ensemble rule ranks first with four of the six classification algorithms while the ensemble rule SumDistance ranks first for two classification algorithms. Also we rank the 10 different ensemble rules with SplitBal for each classification algorithm according to the average AUC values of the 46 imbalanced data, and then summarize the ranks of each ensemble rule to provide the final rank. Table 7 shows the details.

From Table 7, we observe that: (i) For the data balancing method SplitBal, Sum again performs better than the other four ensemble rules proposed by Kittler et al., which is similar to the results obtained with ClusterBal. (ii) Our proposed five ensemble rules all perform better than their corresponding ensemble rule proposed by Kittler et al. This means that when using SplitBal as the data balancing method, our ensemble rules are usually better than those in [31]. (iii) As well as ClusterBal, for SplitBal, our MaxDistance ensemble rule performs best among the 10 ensemble rules over the six classification algorithms.

(3) *Results for the comparison of ClusterBal and SplitBal.* From the former two investigations, we have known that, for both of the two data balancing methods ClusterBal and SplitBal, MaxDistance ensemble rule performs best among the 10 ensemble rules. Thus in this investigation, we compare the combination ClusterBal+MaxDistance to the combination SplitBal+MaxDistance, and find out which combination of data balancing method and ensemble rule performs best.

For each data set, Table 8 provides the detailed AUC values for these two combinations with the six different classification algorithms. Note that the last 'Avg' row stands for the average AUC values of the 46 data sets and furthermore the better average AUC values of the two combinations are highlighted in boldface for a specific classification algorithm.

From Table 8, we observe that the combination ClusterBal+MaxDistance performs better than the combination SplitBal+MaxDistance in terms of average AUC values for the classification algorithms Naive Bayes and SMO. However, for the other four classification algorithms C4.5, RIPPER, Random Forest and IBK, combination SplitBal+MaxDistance performs better.

For the purpose of formally confirming that whether or not the above performance difference is significant, we conducted the Wilcoxon signed rank test [86] at the significance level 0.05. For the classification algorithms Naive Bayes and SMO, the alternative hypotheses are that the combination ClusterBal+MaxDistance performs better than the combination SplitBal+MaxDistance, while for the other four classification algorithms, the alternative hypotheses are that the combination SplitBal+MaxDistance is better than the combination ClusterBal+MaxDistance.

The p-values are 0.0069, 0.0012, 0.0157, 0.0003, 0.0272 and 0.1866, which are corresponding to the hypotheses of classification algorithms Naive Bayes, C4.5, RIPPER, Random Forest, SMO and IBK, respectively. Five of the six p-values are less than the significance level 0.05 except the one 0.1866 corresponding to the hypothesis of classification algorithm IBK. This means that when using the classification algorithms Naive Bayes and SMO, the combination ClusterBal+MaxDistance performs significantly better than the combination SplitBal+MaxDistance, while the combination SplitBal+MaxDistance is significantly better than the

Table 4
Average AUC values of 46 imbalanced data for different ensemble rules with ClusterBal.

Base	Kittler's ensemble rules [31]					Our ensemble rules				
Classifier	Max	Min	Product	Majority	Sum	MaxDistance	MinDistance	ProDistance	MajDistance	SumDistance
Naive Bayes	0.8713	0.5874	0.6100	0.6031	0.8824	**0.8937**	0.5880	0.6110	0.8843	0.8828
C4.5	0.8271	0.7784	0.7784	0.6335	0.8778	**0.9014**	0.7784	0.7784	0.8896	0.8795
RIPPER	0.8223	0.5423	0.5423	0.6409	0.8795	**0.9001**	0.5423	0.5423	0.8927	0.8834
Random Forest	0.9112	0.8542	0.8542	0.6633	0.9205	**0.9224**	0.8542	0.8542	0.9169	0.9222
SMO	0.7806	0.7793	0.7793	0.7232	0.8850	0.8910	0.7793	0.7793	**0.8925**	0.8850
IBK	0.9025	0.9032	0.8667	0.7177	0.9053	**0.9123**	0.9104	0.8665	0.9004	0.9051

Table 5
Rank of different ensemble rules with ClusterBal.

Base	Kittler's ensemble rules [31]					Our ensemble rules				
Classifier	Max	Min	Product	Majority	Sum	MaxDistance	MinDistance	ProDistance	MajDistance	SumDistance
Naive Bayes	5	10	7	8	4	1	9	6	2	3
C4.5	5	6	6	10	4	1	6	6	2	3
RIPPER	5	7	7	6	4	1	7	7	2	3
Random Forest	5	6	6	10	3	1	6	6	4	2
SMO	5	6	6	10	3	2	6	6	1	3
IBK	6	5	8	10	3	1	2	9	7	4
Sum	31	40	40	54	21	7	36	40	18	18
Rank	5	7	7	10	4	1	6	7	2	2

Table 6
Average AUC values of 46 imbalanced data for different ensemble rules with SplitBal.

Base	Kittler's ensemble rules [31]					Our ensemble rules				
Classifier	Max	Min	Product	Majority	Sum	MaxDistance	MinDistance	ProDistance	MajDistance	SumDistance
Naive Bayes	0.8776	0.8567	0.8601	0.8300	0.8787	**0.8811**	0.8577	0.8601	0.8679	0.8793
C4.5	0.8706	0.8041	0.8040	0.8558	0.9107	**0.9146**	0.8041	0.8040	0.9110	0.9127
RIPPER	0.8722	0.6225	0.6330	0.8346	0.8897	**0.9050**	0.6225	0.6336	0.8957	0.8951
Random Forest	0.9274	0.8999	0.9007	0.8730	0.9318	0.9309	0.9002	0.9009	0.9243	**0.9329**
SMO	0.8618	0.8207	0.8207	0.8420	0.8790	**0.8810**	0.8207	0.8207	0.8800	0.8790
IBK	0.9092	0.9086	0.9069	0.8391	0.9113	0.9141	0.9125	0.9090	0.8913	**0.9143**

Table 7
Rank of different ensemble rules with SplitBal.

Base	Kittler's ensemble rules [31]					Our ensemble rules				
classifier	Max	Min	Product	Majority	Sum	MaxDistance	MinDistance	ProDistance	MajDistance	SumDistance
Naive Bayes	4	9	6	10	3	1	8	6	5	2
C4.5	5	7	9	6	4	1	7	9	3	2
RIPPER	5	9	8	6	4	1	9	7	2	3
Random Forest	4	9	7	10	2	3	8	6	5	1
SMO	5	7	7	6	3	1	7	7	2	3
IBK	5	7	8	10	4	2	3	6	9	1
Sum	28	48	45	48	20	9	42	41	26	12
Rank	4	9	8	9	3	1	7	6	5	2

combination ClusterBal+MaxDistance with the three classification algorithms C4.5, RIPPER and Random Forest. In addition, when using IBK as the classification algorithm, these two combinations are equivalent.

(4) *Results for the comparison of our method with the external methods.* This investigation explores whether or not our proposed method is more effective at dealing with the imbalanced problems by comparing our method with the conventional external imbalance data handling methods with different classification algorithms as base classifiers. From the former investigation we knew that the best two combinations (ClusterBal+MaxDistance and SplitBal+MaxDistance) of data balancing method and ensemble rule perform differently with different classification algorithms. Thus in this investigation we compare the results of the both combinations with results of conventional imbalance data handling methods. For space limitations, for each classification algorithm, we only provide the average AUC values of the 46 imbalanced data sets instead of the individual AUC value for each imbalanced data set.

Table 9 shows the detailed average AUC values of the 46 imbalanced data sets for different imbalance data handling methods for the ensembles with the six classification algorithms as base classifiers.

From Table 9 we observe that, for each of the six classification algorithms, the two combinations always achieve the best average AUC value among all the imbalance data handling methods. Particularly, the combination ClusterBal+MaxDistance performs best for classification algorithms Naive Bayes and SMO, while for the three classification algorithms J48, Ripper and IBK, the combination SplitBal+MaxDistance performs best.

For the purpose of providing the readers a big picture about the performance difference of these 14 methods, we rank these 14 methods for each specific classification algorithm according to the average AUC values, summarize these ranks over the six classification algorithms and give the final rank. Table 10 shows the details.

From Table 10, we observe that the two combinations SplitBal+MaxDistance and ClusterBal+MaxDistance rank first and second, respectively. This means that with the six classification algorithms as base classifiers, our proposed method could not only deal with the class imbalance problems but also perform better than the conventional external imbalance data handling methods.

In order to statistically confirm this conclusion, for each classification algorithm, the Friedman test [87], which is based on the ranked performance rather than the actual performance estimates and therefore is less susceptible to outliers, was conducted with the significance level 0.05 to compare these different imbalance data handling methods over multiple data sets. The six p-values are all less than 0.05, which indicates that the performance differences among the 14 imbalance data handling methods are not random and therefore confirms the differences of average AUC values are significant. After that, a post hoc test was performed to identify which particular methods significantly perform best, as suggested by Demsar [88]. This was accomplished by applying the Nemenyi test [89] at the significance level $\alpha = 0.05$. Fig. 2 includes six subfigures, each of which shows the results of the Nemenyi test for a specific classification algorithm.

Each subfigure of Fig. 2 plots the imbalance data handling methods against average performance ranks, where all methods are sorted according to their ranks. The '*' denotes the respective average rank of each method and the line segment to the right of each method represents its critical difference, which means the methods whose '*' on the right end of the line are outperformed significantly. The critical difference is highlighted with a vertical dotted line in two cases. The left vertical line is associated with the best method, and all methods right to this line perform significantly worse than this method. The right vertical line is associated with the worst method, and all methods left to this line perform significantly better than it.

From Fig. 2 we observe that (i) For each classification algorithm, the two combinations perform significantly better than the Orig method. This means that our proposed method could improve the performance of Orig method significantly, which indicates that our method could be used to deal with the class imbalance problems. (ii) For the three sampling methods (RUS, ROS and SMOTE) and the cost sensitive learning method MetaCost, the two combinations outperform them significantly for all the six classification algorithms. This means that our proposed method is more effective than the four conventional

Table 8
AUC values for the two combinations using different classification algorithms.

	Naive Bayes		C4.5		RIPPER		Random Forest		SMO		IBK	
Data	ClusterBal	SplitBal	ClusterBal	SplitBal	ClusterBal	SplitBal	ClusterBal	SplitBal	ClusterBal	SplitBal	ClusterBal	SplitBal
yeast3	0.9617	**0.9666**	0.9604	**0.9666**	0.9645	**0.9679**	0.9693	**0.9743**	**0.9519**	0.9377	0.9627	**0.9716**
ecoli3	**0.9305**	0.9286	**0.9512**	0.9252	**0.9495**	0.9289	**0.9403**	0.9306	**0.9508**	0.9082	0.9433	**0.9469**
pageblocks0	0.9197	**0.9280**	0.9663	**0.9858**	0.9705	**0.9857**	0.9846	**0.9910**	**0.8959**	0.8600	0.9650	**0.9710**
ecoli034vs5	**0.9438**	0.8703	0.9338	**0.9566**	0.9437	**0.9519**	0.9644	**0.9786**	**0.9397**	0.9170	0.9415	**0.9718**
yeast2vs4	0.9249	**0.9282**	0.9282	**0.9749**	0.9284	**0.9711**	0.9644	**0.9826**	**0.9125**	0.9081	0.9450	**0.9730**
ecoli067vs35	**0.9170**	0.8959	**0.9193**	0.9004	**0.9125**	0.9027	**0.9262**	0.9208	**0.9193**	0.8931	0.9220	**0.9365**
ecoli0234vs5	**0.9399**	0.8835	0.9465	**0.9599**	0.9448	**0.9647**	0.9636	**0.9721**	**0.9393**	0.9171	0.9437	**0.9695**
glass015vs2	0.5885	**0.6511**	0.6611	**0.7284**	0.6411	**0.6865**	0.7594	**0.8067**	0.5571	**0.5828**	**0.6954**	0.6915
yeast0359vs78	**0.7835**	0.7676	0.7647	**0.7789**	0.7638	**0.7740**	0.7986	**0.8014**	**0.7712**	0.7667	**0.8060**	0.7749
yeast0256vs3789	0.7940	**0.8281**	0.8051	**0.8487**	0.8087	**0.8300**	0.8358	**0.8613**	0.8019	**0.8175**	**0.8413**	0.8340
yeast02579vs368	**0.9279**	0.9269	**0.9256**	0.9217	0.9266	**0.9281**	0.9419	**0.9501**	**0.9273**	0.9203	**0.9454**	0.9429
ecoli046vs5	**0.9441**	0.8692	0.9386	**0.9573**	0.9406	**0.9627**	0.9742	**0.9745**	**0.9459**	0.9171	0.9538	**0.9700**
ecoli01vs235	**0.9348**	0.9123	0.9340	**0.9430**	0.9323	**0.9516**	0.9312	**0.9567**	**0.9376**	0.9072	0.9368	**0.9675**
ecoli0267vs35	**0.9122**	0.8990	**0.9071**	0.9046	**0.8998**	0.8885	0.9055	**0.9300**	**0.9033**	0.8859	0.9049	**0.9337**
glass04vs5	**0.9967**	0.9928	**0.9945**	0.9901	**0.9976**	0.9938	0.9999	**1.0000**	0.9959	**0.9963**	**0.9912**	0.9809
ecoli0346vs5	**0.9472**	0.8613	0.9449	**0.9545**	0.9416	**0.9505**	0.9629	**0.9694**	**0.9586**	0.9001	**0.9717**	0.9679
ecoli0347vs56	**0.9539**	0.9045	**0.9574**	0.9367	**0.9468**	0.9400	**0.9661**	0.9577	**0.9584**	0.9277	0.9582	**0.9611**
yeast05679vs4	**0.8412**	0.8257	0.8516	**0.8732**	0.8535	**0.8621**	0.8768	**0.8962**	**0.8361**	0.8275	0.8607	**0.8859**
vowel0	**0.9822**	0.9797	**0.9961**	0.9914	**0.9910**	0.9879	**0.9993**	0.9989	**0.9903**	0.9558	**1.0000**	0.9992
ecoli067vs5	**0.9031**	0.8570	0.9016	**0.9327**	0.9022	**0.9249**	0.9371	**0.9494**	0.9067	**0.9085**	0.9096	**0.9503**
glass016vs2	0.6369	**0.6679**	0.6943	**0.7710**	0.6997	**0.7111**	0.7865	**0.8364**	0.6252	**0.6650**	0.7484	**0.7564**
ecoli0147vs2356	**0.9291**	0.8961	0.9329	**0.9354**	0.9276	**0.9368**	0.9320	**0.9369**	**0.9356**	0.8985	0.9434	**0.9475**
led7digit02456789vs1	**0.9616**	0.9522	**0.9614**	0.9508	**0.9664**	0.9588	**0.9605**	0.9576	**0.9638**	0.9463	0.9633	**0.9650**
ecoli01vs5	**0.9614**	0.8742	0.9544	**0.9799**	0.9567	**0.9781**	0.9772	**0.9853**	**0.9586**	0.9097	0.9623	**0.9778**
glass06vs5	**0.9874**	0.9763	**0.9957**	0.9952	0.9868	**0.9888**	**0.9988**	**0.9988**	0.9851	**0.9907**	**0.9943**	0.9919
glass0146vs2	0.6567	**0.6736**	0.7303	**0.7598**	**0.7358**	0.7164	0.7965	**0.8396**	0.6498	**0.6886**	0.7386	**0.7494**
glass2	**0.7202**	0.7063	0.7480	**0.7852**	**0.7355**	0.7264	0.8274	**0.8412**	**0.7099**	0.6856	**0.7795**	0.7288
ecoli0147vs56	**0.9635**	0.9236	**0.9643**	0.9635	**0.9580**	0.9575	**0.9696**	0.9673	**0.9657**	0.9325	0.9642	**0.9665**
cleveland0vs4	**0.9590**	0.9474	0.9591	**0.9713**	0.9476	**0.9618**	**0.9552**	0.9528	0.9626	**0.9668**	0.9762	**0.9831**
ecoli0146vs5	**0.9506**	0.8840	0.9421	**0.9657**	0.9421	**0.9599**	**0.9786**	0.9735	**0.9496**	0.9016	0.9551	**0.9718**
shuttlec0vsc4	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9984	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
yeast1vs7	0.7835	**0.8090**	0.7708	**0.8038**	0.7769	**0.8132**	0.8058	**0.8233**	0.7601	**0.8228**	0.8063	**0.8236**
glass4	**0.9361**	0.9243	**0.9742**	0.9393	**0.9790**	0.9561	**0.9801**	0.9693	**0.9616**	0.9462	**0.9593**	0.9384
ecoli4	**0.9973**	0.9919	**0.9936**	0.9921	**0.9929**	0.9761	**0.9933**	0.9888	**0.9951**	0.9899	0.9938	**0.9956**
pageblocks13vs4	**0.9873**	0.9486	0.9976	**0.9988**	**0.9987**	0.9924	**0.9999**	0.9998	**0.9883**	0.8234	**0.9990**	0.9948
abalone918	**0.7681**	0.7397	**0.7714**	0.7673	0.7613	**0.7874**	0.8054	**0.8209**	0.7512	**0.7578**	**0.7631**	0.7561
glass016vs5	**0.9895**	0.9796	**0.9954**	0.9890	**0.9923**	0.9855	0.9950	**0.9968**	0.9857	**0.9874**	**0.9870**	0.9737
shuttlec2vsc4	0.9955	**1.0000**	**1.0000**	**1.0000**	**0.9989**	0.8860	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
yeast1458vs7	0.6638	**0.6861**	0.6686	**0.6965**	0.6712	**0.6754**	**0.7163**	0.7101	0.6657	**0.6978**	**0.7240**	0.7133
glass5	**0.9889**	0.9835	**0.9970**	0.9889	**0.9899**	0.9846	0.9945	**0.9966**	0.9834	**0.9892**	**0.9836**	0.9604
yeast2vs8	**0.8034**	0.7893	0.7935	**0.8190**	0.8012	**0.8021**	0.8394	**0.8431**	0.7840	**0.8131**	**0.8195**	0.7640
yeast4	**0.8673**	0.8595	0.8834	**0.9166**	0.8800	**0.9173**	0.9031	**0.9320**	0.8685	**0.8729**	0.9012	**0.9232**
yeast1289vs7	0.7235	**0.7565**	0.7242	**0.7816**	0.7214	**0.7496**	0.7528	**0.7772**	0.7020	**0.7613**	**0.7721**	0.7631
yeast5	**0.9879**	0.9865	0.9892	**0.9900**	**0.9910**	0.9881	**0.9914**	0.9911	**0.9873**	0.9687	**0.9922**	0.9893
ecoli0137vs26	0.9152	**0.9503**	0.9053	**0.9535**	0.9080	**0.9419**	0.9438	**0.9517**	0.9175	**0.9458**	**0.9112**	0.8908
yeast6	0.9289	**0.9465**	**0.9299**	0.9251	0.9256	**0.9263**	0.9247	**0.9294**	**0.9296**	0.9106	**0.9291**	0.9261
Avg	**0.8937**	0.8811	0.9014	**0.9146**	0.9001	**0.9050**	0.9224	**0.9309**	**0.8910**	0.8810	0.9123	**0.9141**

The 'ClusterBal' columns stand for the results of combination ClusterBal+MaxDistance while the 'SplitBal' columns represent the results of combination SplitBal+MaxDistance.

Table 9
Average AUC values of 46 imbalanced data for different imbalance data handling methods.

Base	Conventional imbalance data handling methods												Our methods	
Classifier	Orig	RUS	ROS	SMOTE	MetaCost	Bagging	Boosting	EasyEnsemble	RUSBoost	SMOTEBoost	UnderBagging	EM1vs1	ClusterBal	SplitBal
Naive Bayes	0.8687	0.8587	0.8691	0.8646	0.8423	0.8759	0.8566	0.8790	0.8670	0.8630	0.8780	0.8651	**0.8937**	0.8811
J48	0.7984	0.8350	0.8147	0.8499	0.8364	0.8841	0.8999	0.9019	0.8985	0.9093	0.9083	0.9096	0.9014	**0.9146**
Ripper	0.7764	0.8067	0.8080	0.8423	0.8286	0.8556	0.8940	0.8941	0.8803	0.9009	0.8882	0.8919	0.9001	**0.9050**
Random Forest	0.8998	0.8979	0.8996	0.9095	0.9060	0.9305	0.9068	0.9215	0.9232	0.9254	0.9313	**0.9317**	0.9224	0.9309
SMO	0.6726	0.8317	0.8456	0.8485	0.6608	0.7190	0.8779	0.8865	0.8581	0.8668	0.8771	0.8753	**0.8910**	0.8810
IBK	0.8837	0.8819	0.8774	0.8973	0.8816	0.8917	0.8738	0.8889	0.8835	0.9001	0.9092	0.9077	0.9123	**0.9141**

The 'ClusterBal' columns stand for the results of combination ClusterBal+MaxDistance while the 'SplitBal' columns represent the results of combination SplitBal+MaxDistance.

imbalance data handling methods RUS, ROS, SMOTE, and MetaCost when dealing with the class imbalance problems. (iii) For the basic Bagging and Boosting methods, with the three classification algorithms C4.5, RIPPER and SMO, our two combinations both perform significantly better than Bagging while with the other three classification algorithms, Boosting is significantly outperformed by our two

Table 10
Rank of different imbalance data handling methods over the six classification algorithms.

Base	Conventional imbalance data handling methods												Our methods	
Classifier	Orig	RUS	ROS	SMOTE	MetaCost	Bagging	Boosting	EasyEnsemble	RUSBoost	SMOTEBoost	UnderBagging	EM1vs1	ClusterBal	SplitBal
Naive Bayes	7	12	6	10	14	5	13	3	8	11	4	9	1	2
J48	14	12	13	10	11	9	7	5	8	3	4	2	6	1
Ripper	14	13	12	10	11	9	5	4	8	2	7	6	3	1
Random Forest	12	14	13	9	11	4	10	8	6	5	2	1	7	3
SMO	13	11	10	9	14	12	4	2	8	7	5	6	1	3
IBK	9	11	13	6	12	7	14	8	10	5	3	4	2	1
Sum	69	73	67	54	73	46	53	30	48	33	25	28	20	11
Rank	12	13	11	10	13	7	9	5	8	6	3	4	2	1

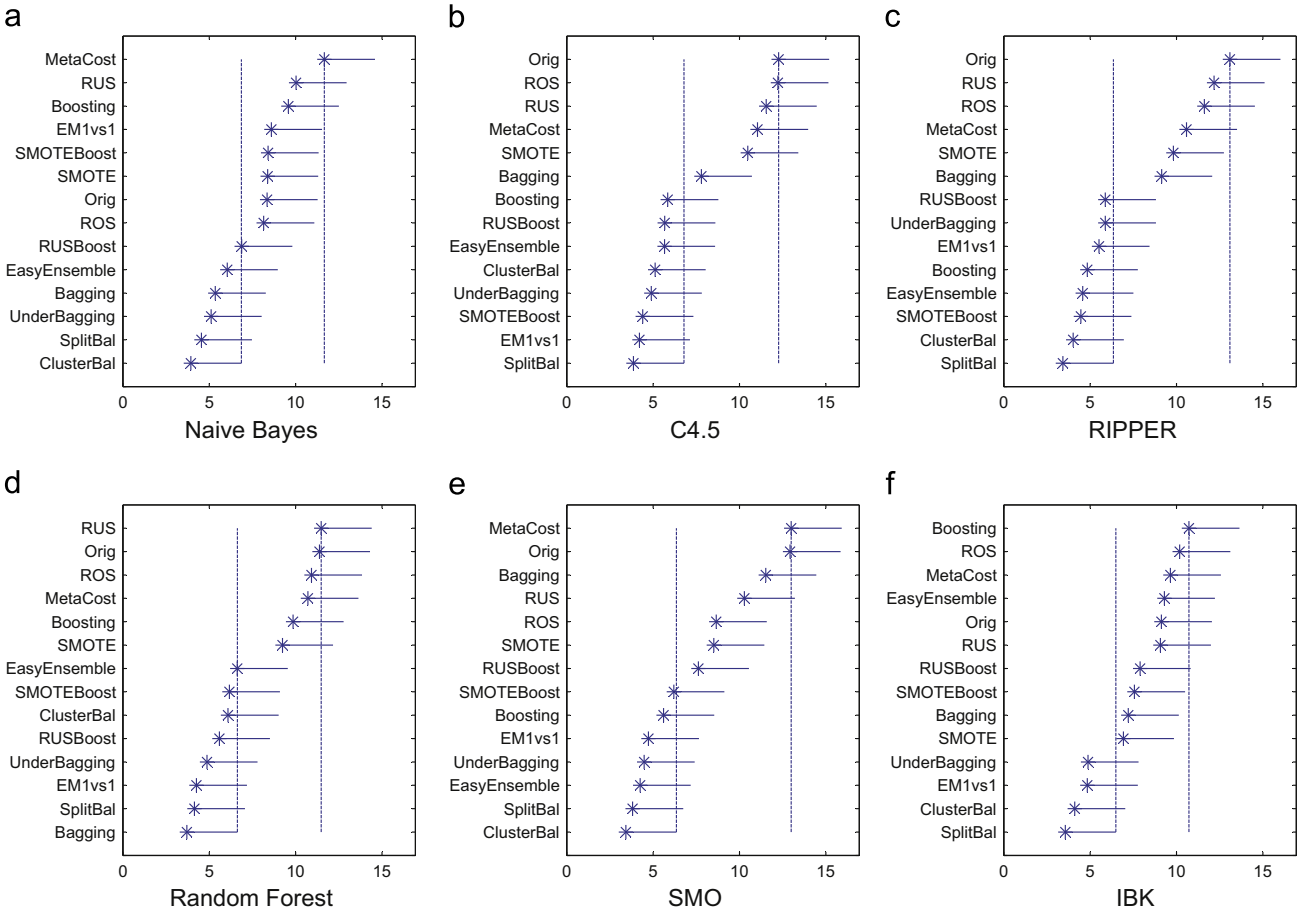

Fig. 2. Results of the pairwise comparisons of the 14 models using Nemenyi post hoc test with $\alpha = 0.05$.

combinations. (iv) For comparison between our methods and the three Bagging and Boosting based ensemble methods EasyEnsemble, RUSBoost and SMOTEBoost, different results could be obtained with different classification algorithms. Specially, for IBK, the two combinations both perform significantly better than these three ensemble methods. However, for C4.5, RIPPER and Random Forest, the two combinations are not significantly better than any of the three ensemble methods. In addition, the two combinations both perform significantly better than SMOTEBoost for Naive Bayes while for classification algorithm SMO, the two combinations both outperform RUSBoost significantly. All in all, our proposed method could be a good alternative for Bagging and Boosting based ensemble methods when dealing with the class imbalance problems, and it usually performs better than these conventional ensemble methods for some specific classification algorithms. (v) For comparison with UnderBagging, our proposed method performs better than it across the six employed classification algorithms, though not significantly. (vi) For our former method EM1vs1 [40], the new proposed combinations both perform significantly better than it with the classification algorithm Naive Bayes. However, for the other five classification algorithms, these three methods are equivalent when dealing with the class imbalance problems.

From the above results we can know that our proposed method is able to improve the performance of classifiers over originally binary-class imbalance data, and it usually performs better than the conventional external imbalance data handling methods.

(5) *Results for the comparison of our method with the fuzzy rule based methods.* From Table 8 we could know that both of our two data balancing methods ClusterBal and SplitBal perform best with the classification algorithm Random Forest while perform worst with the classification algorithm SMO when using the MaxDistance

Table 11
The AUC of Chi3-GTS and Chi5-GTS vs. the best and worst AUC of our method.

	Fuzzy rule based methods		ClusterBal+MaxDistance		SplitBal+MaxDistance	
Data	Chi3-GTS	Chi5-GTS	Random forest	SMO	Random forest	SMO
yeast3	0.5000	0.6179	0.9693	0.9519	**0.9743**	0.9377
ecoli3	0.5126	0.7608	0.9403	**0.9508**	0.9306	0.9082
pageblocks0	0.7028	0.7458	0.9846	0.8959	**0.9910**	0.8600
ecoli034vs5	0.8194	0.8278	0.9644	0.9397	**0.9786**	0.9170
yeast2vs4	0.6160	0.7297	0.9644	0.9125	**0.9826**	0.9081
ecoli067vs35	0.7575	0.7950	**0.9262**	0.9193	0.9208	0.8931
ecoli0234vs5	0.7945	0.8253	0.9636	0.9393	**0.9721**	0.9171
glass015vs2	0.4871	0.4734	0.7594	0.5571	**0.8067**	0.5828
yeast0359vs78	0.5967	0.5714	0.7986	0.7712	**0.8014**	0.7667
yeast0256vs3789	0.5966	0.6861	0.8358	0.8019	**0.8613**	0.8175
yeast02579vs368	0.6535	0.8698	0.9419	0.9273	**0.9501**	0.9203
ecoli046vs5	0.8000	0.8308	0.9742	0.9459	**0.9745**	0.9171
ecoli01vs235	0.7627	0.7714	0.9312	0.9376	**0.9567**	0.9072
ecoli0267vs35	0.7875	0.8101	0.9055	0.9033	**0.9300**	0.8859
glass04vs5	0.5316	0.6456	0.9999	0.9959	**1.0000**	0.9963
ecoli0346vs5	0.7973	0.8480	0.9629	0.9586	**0.9694**	0.9001
ecoli0347vs56	0.8156	0.8476	**0.9661**	0.9584	0.9577	0.9277
yeast05679vs4	0.4969	0.5716	0.8768	0.8361	**0.8962**	0.8275
vowel0	0.9278	0.9772	**0.9993**	0.9903	0.9989	0.9558
ecoli067vs5	0.7975	0.7875	0.9371	0.9067	**0.9494**	0.9085
glass016vs2	0.4857	0.4457	0.7865	0.6252	**0.8364**	0.6650
ecoli0147vs2356	0.7201	0.7772	0.9320	0.9356	**0.9369**	0.8985
led7digit02456789vs1	0.8257	0.8257	0.9605	**0.9638**	0.9576	0.9463
ecoli01vs5	0.8205	0.8318	0.9772	0.9586	**0.9853**	0.9097
glass06vs5	0.6300	0.7500	**0.9988**	0.9851	**0.9988**	0.9907
glass0146vs2	0.4920	0.4601	0.7965	0.6498	**0.8396**	0.6886
glass2	0.4822	0.4569	0.8274	0.7099	**0.8412**	0.6856
ecoli0147vs56	0.7767	0.8205	**0.9696**	0.9657	0.9673	0.9325
cleveland0vs4	0.3505	0.1151	0.9552	0.9626	0.9528	**0.9668**
ecoli0146vs5	0.7481	0.8308	**0.9786**	0.9496	0.9735	0.9016
shuttlec0vsc4	0.9912	0.9831	**1.0000**	**1.0000**	**1.0000**	**1.0000**
yeast1vs7	0.4977	0.5147	0.8058	0.7601	**0.8233**	0.8228
glass4	0.5567	0.7576	**0.9801**	0.9616	0.9693	0.9462
ecoli4	0.7234	0.8671	0.9933	**0.9951**	0.9888	0.9899
pageblocks13vs4	0.6922	0.6888	**0.9999**	0.9883	0.9998	0.8234
abalone918	0.5000	0.5111	0.8054	0.7512	**0.8209**	0.7578
glass016vs5	0.4857	0.7043	0.9950	0.9857	**0.9968**	0.9874
shuttlec2vsc4	0.9338	0.8338	**1.0000**	**1.0000**	**1.0000**	**1.0000**
yeast1458vs7	0.5000	0.4932	**0.7163**	0.6657	0.7101	0.6978
glass5	0.4927	0.7085	0.9945	0.9834	**0.9966**	0.9892
yeast2vs8	0.7478	0.7446	0.8394	0.7840	**0.8431**	0.8131
yeast4	0.5000	0.5248	0.9031	0.8685	**0.9320**	0.8729
yeast1289vs7	0.5000	0.4945	0.7528	0.7020	**0.7772**	0.7613
yeast5	0.5233	0.5965	**0.9914**	0.9873	0.9911	0.9687
ecoli0137vs26	0.8445	0.8336	0.9438	0.9175	**0.9517**	0.9458
yeast6	0.5000	0.5955	0.9247	0.9296	**0.9294**	0.9106
Avg	0.6538	0.6991	0.9224	0.8910	**0.9309**	0.8810

ensemble rule. Thus in this section we will compare the results of the fuzzy rule based methods with the best and worst results of our methods. Table 11 shows the classification results of the 46 datasets in terms of AUC. Note that for each specific data set, the best AUC value was highlighted in boldface.

From Table 11, we find that our two combinations with Random Forest and SMO all perform better than the Chi3-GTS and Chi5-GTS in terms of the average AUC values over the 46 employed datasets.

For the purpose of formally confirming whether the performance difference is significant, we conducted the popular Wilcoxon signed rank test at the significance level 0.05. The alternative hypotheses are that for each of the two classification algorithms Random Forest and SMO, the two combinations both perform better than the two fuzzy rule based classification methods Chi3-GTS and Chi5-GTS. All the corresponding p-values are less than the significance level 0.05, which means that our proposed method with Random Forest and SMO perform significantly better than the two fuzzy rule based methods. This further indicates that with the other four classification algorithms, our method outperforms these two fuzzy rule based methods significantly.

(6) *The impact of different Const*[2] *values on the classification performance.* In former sections, our ensemble rules with *Const* value 1 have been employed to combine with the two data balancing methods SplitBal and ClusterBal, and the proposed method has shown better classification performance than the conventional imbalance data handling methods. This section is conducted to validate that whether the *Const* value with 1 in our ensemble rules is reasonable. Particularly, we have performed a series of experiments with five different values as the *Const* value, including 0.01, 0.1, 1, 10 and 100.

Figs. 3 and 4 respectively show the impact of different *Const* values on the classification performance when combining our five ensemble rules with the two data balancing methods SplitBal and ClusterBal. Note that in each subfigure of these two figures, the *X*-axis stands for various *Const* values in our ensemble rules and the *Y*-axis represents the classification performance in terms of the average AUC value.

[2] *Const* is defined in Section 4.3.(6)

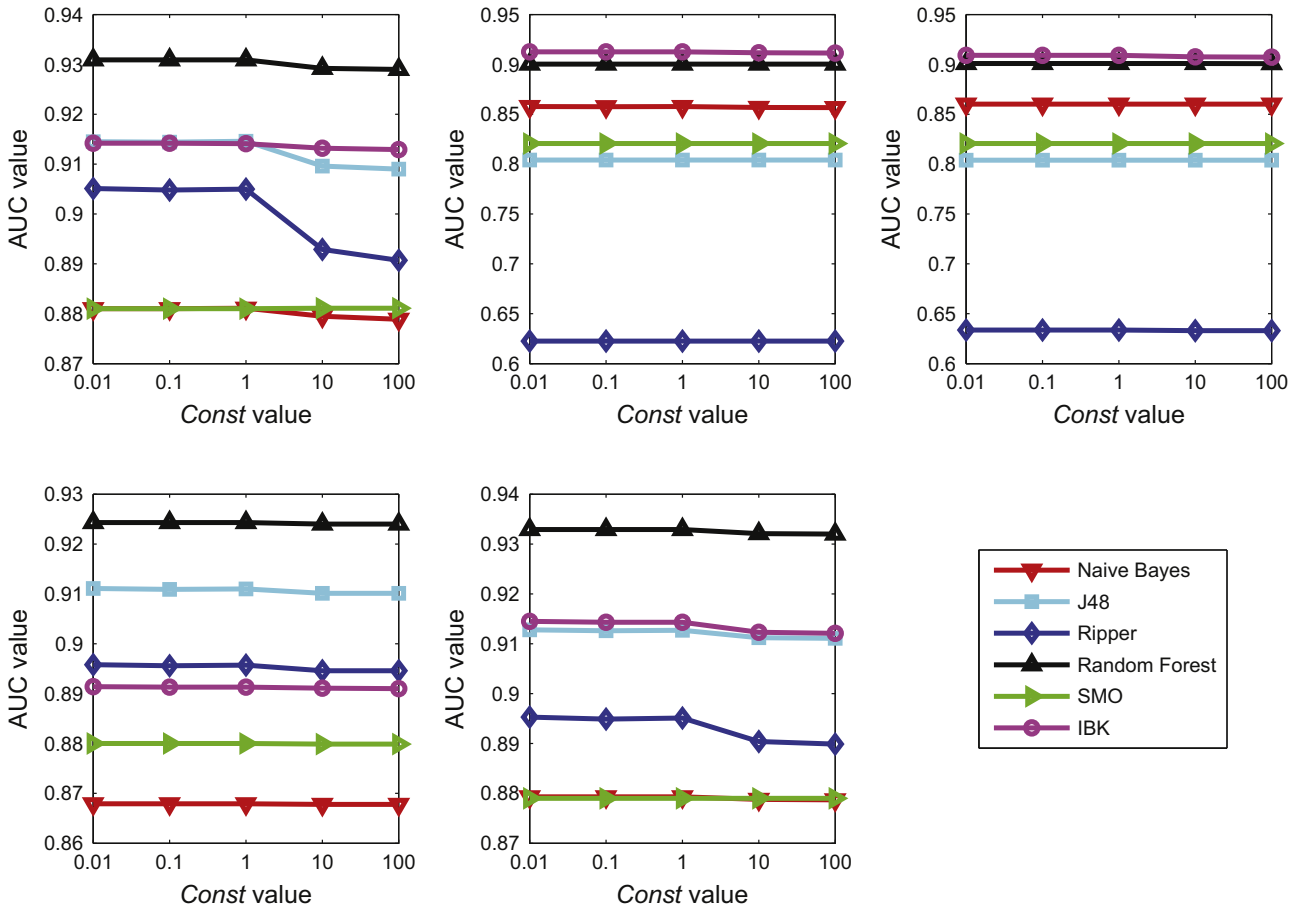

Fig. 3. The impact of various *Const* values on AUC value when combining our five ensemble rules with SplitBal.

From Fig. 3(a), we observe that for combining the ensemble rule MaxDistance with data balancing method SplitBal, the average AUC value remains unchanged when changing the *Const* value from 0.01 to 1 for all the six employed classification algorithms. However, when changing the *Const* value from 1 to 100, two classification algorithms J48 and Ripper show obvious decrease in average AUC value while the other four classification algorithms still keep unchanged.

From Fig. 3(b)–(d), it could be observed that for combining each of the three ensemble rules (MinDistance, ProDistance and MajDistance) with data balancing method SplitBal, there is no obvious change in average AUC value for all the six employed classification algorithms when changing the *Const* value from 0.01 to 100. This means that changing the *Const* value has little effect on these three ensemble rules with data balancing method SplitBal.

From Fig. 3(e), we observe that for combining the ensemble rule SumDistance with data balancing method SplitBal, when changing the *Const* value from 0.01 to 1, the average AUC value remains unchanged for all the six employed classification algorithms. Furthermore, when changing the *Const* value from 1 to 100, the average AUC value remains unchanged for the five classification algorithms Naive Bayes, J48, Random Forest, SMO and IBK. However, for the classification algorithm Ripper, it shows obvious decrease in average AUC value when changing the *Const* value from 1 to 100.

From Fig. 4(a), we could observe that for the combination of ensemble rule MaxDistance and data balancing method ClusterBal, no obvious change could be observed in average AUC value when changing the *Const* value from 0.01 to 1 for all the six employed classification algorithms. However, when changing the *Const* value from 1 to 100, the average AUC values of four classification algorithms (including Naive Bayes, J48, Ripper and SMO) show clear decrease.

From Figs. 4(b) and 4(c), no obvious change in average AUC value could be observed for all the six employed classification algorithms when changing the *Const* value from 0.01 to 100. This means that when combining the ensemble rules MinDistance and ProDistance with data balancing method ClusterBal, the impact of the *Const* value could be negligible.

From Figs. 4(d) and 4(e), we observe that when changing the *Const* value from 0.01 to 1, no obvious change could be observed for all the six employed classification algorithms when combining the two ensemble rules MajDistance and SumDistance with data balancing method ClusterBal. However, when changing the *Const* value from 1 to 100, classification algorithm Ripper shows clear decrease in average AUC value while the other five classification algorithms remain unchanged.

In summary, we could conclude as follows: (1) When changing *Const* value from 0.01 to 100, in most cases the average AUC value remains unchanged for combining our five ensemble rules with the two data balancing methods SplitBal and ClusterBal. It means that in such cases the impact of the *Const* value on classification performance could be negligible and we could select a random number as the *Const* value. This may be the reason that many researchers have selected 1 as the added constant value in denominator. (2) In some cases, changing the *Const* value from 0.01 to 1 has little effect on the classification performance while the classification performance decreases obviously when changing the *Const* value from 1 to 100. This means that in these cases, the numbers less than 1 could be randomly selected as the *Const* value while the numbers greater than 1 should be taken seriously,

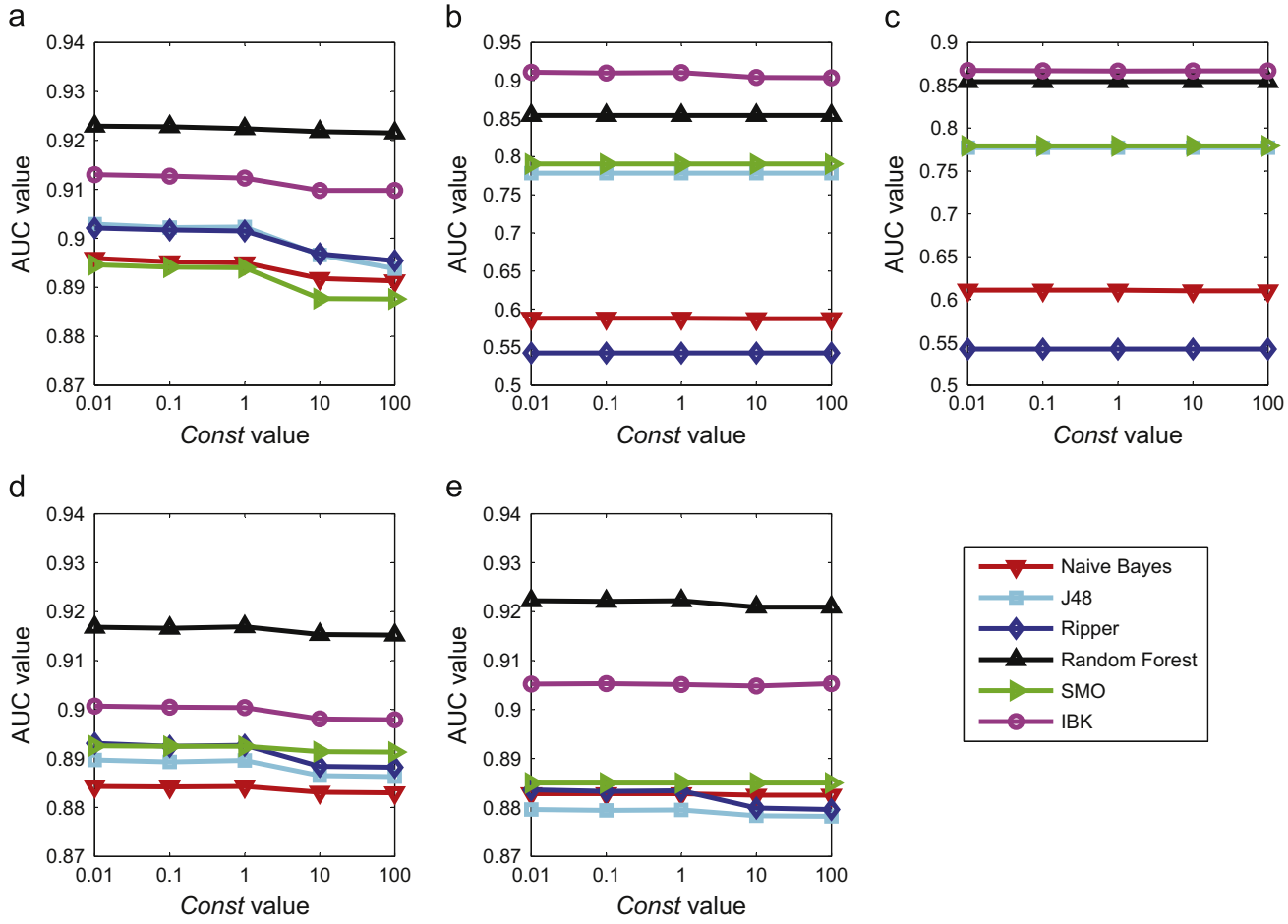

Fig. 4. The impact of various *Const* values on AUC value when combining our five ensemble rules with ClusterBal.

especially for very large numbers. Therefore, on the basis of the above two conclusions, it is reasonable that we select 1 as the *Const* value added to distance in our ensemble rules.

5. Conclusion

In this paper, we have presented an ensemble method for dealing with the binary-class imbalance problems. Different from the conventional sampling methods, cost-sensitive learning methods, and Bagging and Boosting based ensemble methods, the proposed method does not change the original class distribution, and does not suffer from information loss or unexpected mistakes that may be caused by these conventional methods via increasing the minority class instances or decreasing the majority ones.

The proposed method firstly converts the imbalanced binary-class data into multiple balanced binary-class data. This is achieved by applying random splitting or clustering to the majority class instances. After that, a specific classification algorithm is applied to the multiple balanced binary-class data to build multiple classifiers. Finally, the classification results of these binary classifiers for a new data are combined with a specific ensemble rule. Five new ensemble rules including MaxDistance, MinDistance, ProDistance, MajDistance and SumDistance, which depict the relationship between a new problem and the historical data, are presented.

An empirical study has been conducted as well. The experimental results show that (i) It is reasonable that we select 1 as the added constant value to the distance in our ensemble rules. (ii) For both of the two data balancing methods ClusterBal and SplitBal, our ensemble rule MaxDistance performs best. (iii) The data balancing method and ensemble rule combination ClusterBal+MaxDistance is better for Naive Bayes and SMO while the combination SplitBal+MaxDistance prefers C4.5, RIPPER and Random Forest. (iv) Our proposed method is able to handle the binary-class imbalance problems, and it usually performs more effectively than the conventional methods, including the internal and external imbalance data handling methods.

Conflict of interest

None declared.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grants 61373046 and 61210004.

References

[1] W. Chao, J. Liu, J. Ding, Facial age estimation based on label-sensitive learning and age-oriented regression, Pattern Recognit. 46 (2013) 628–641.
[2] M. Kubat, R.C. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Mach. Learn. 30 (1998) 195–215.
[3] W. Khreich, E. Granger, A. Miri, R. Sabourin, Adaptive roc-based ensembles of hmms applied to anomaly detection, Pattern Recognit. 45 (2012) 208–230.
[4] T. Fawcett, F. Provost, Adaptive fraud detection, Data Min. Knowl. Discov. 1 (1997) 291–316.

[5] L. Pelayo, S. Dick, Applying novel resampling strategies to software defect prediction, in: Annual Meeting of the North American on Fuzzy Information Processing Society, pp. 69–72.
[6] D. Zhang, M.M. Islam, G. Lu, A review on automatic image annotation techniques, Pattern Recognit. 45 (2012) 346–362.
[7] N. Japkowicz (Ed.), Learning from imbalanced data sets, AAAI Workshop on Learning from Imbalanced Data Sets, 2000. Technical Report WS-00-05.
[8] N. J. Nitesh Chawla, A. Kolcz (Eds.), Workshop on learning from imbalanced data sets II, in: International Conference on Machine Learning, 2003.
[9] N. Chawla, N. Japkowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, ACM SIGKDD Explor. Newsl. 6 (2004) 1–6.
[10] G.M. Weiss, Mining with rarity: a unifying framework, ACM SIGKDD Explor. Newsl. 6 (2004) 7–19.
[11] G. Batista, R. Prati, M. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newsl. 6 (2004) 20–29.
[12] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (2009) 1263–1284.
[13] I. Mani, I. Zhang, KNN approach to unbalanced data distributions: a case study involving information extraction, in: Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets II.
[14] W. Liu, S. Chawla, Class confidence weighted knn algorithms for imbalanced data sets, Adv. Knowl. Disc. Data Min. 6635 (2011) 345–356.
[15] N. Chawla, A. Lazarevic, L. Hall, K. Bowyer, Smoteboost: Improving prediction of the minority class in boosting, in: Knowledge Discovery in Databases: PKDD 2003, 2003, pp. 107–119.
[16] C. Seiffert, T. Khoshgoftaar, J. van Hulse, A. Napolitano, Rusboost: A hybrid approach to alleviating class imbalance, IEEE Trans. Syst. Man Cybern. A: Syst. Hum. 40 (2010) 185–197.
[17] F. Provost, Machine learning from imbalanced data sets 101, Technical Report, AAAI Workshop on Learning from Imbalanced Data Sets, 2000.
[18] Y. Sun, M. Kamel, A. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognit. 40 (2007) 3358–3378.
[19] X. Liu, J. Wu, Z. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Trans. Syst. Man Cybern. B: Cybern. 39 (2009) 539–550.
[20] R. Barandela, J.S. Sánchez, V. Garcıa, E. Rangel, Strategies for learning in class imbalance problems, Pattern Recognit. 36 (2003) 849–851.
[21] M.A. Tahir, J. Kittler, F. Yan, Inverse random under sampling for class imbalance problem and its application to multi-label classification, Pattern Recognit. 45 (2012) 3738–3750.
[22] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[23] S. García, F. Herrera, Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy, Evol. Comput. 17 (2009) 275–306.
[24] G. Weiss, K. McCarthy, B. Zabar, Cost-sensitive learning vs. sampling: which is best for handling unbalanced classes with unequal error costs, in: Proceedings of the International Conference on Data Mining, pp. 35–41.
[25] Z. Zhou, X. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Trans. Knowl. Data Eng. 18 (2006) 63–77.
[26] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, A. Napolitano, A comparative study of data sampling and cost sensitive learning, in: Proceedings of IEEE International Conference on Data Mining Workshops, pp. 46–52.
[27] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[28] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the International Conference on Machine Learning, pp. 148–156.
[29] H. Guo, H. Viktor, Learning from imbalanced data sets with boosting and data generation: the databoost-im approach, ACM SIGKDD Explor. Newsl. 6 (2004) 30–39.
[30] S. Hido, H. Kashima, Y. Takahashi, Roughly balanced bagging for imbalanced data, Stat. Anal. Data Min. 2 (2009) 412–426.
[31] J. Kittler, M. Hatef, R.P. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 226–239.
[32] G. H. John, P. Langley, Estimating continuous distributions in bayesian classifiers, in: Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 338–345.
[33] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.
[34] W.W. Cohen, Fast effective rule induction, in: Twelfth International Conference on Machine Learning, pp. 115–123.
[35] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[36] J. Platt, Machines using sequential minimal optimization, in: Advances in Kernel Methods – Support Vector Learning, MIT Press, 1998.
[37] D. Aha, D. Kibler, Instance-based learning algorithms, Mach. Learn. 6 (1991) 37–66.
[38] P. Domingos, Metacost: a general method for making classifiers cost-sensitive, in: Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 155–164.
[39] R. Barandela, R.M. Valdovinos, J.S. Sánchez, New applications of ensembles of classifiers, Pattern Anal. Appl. 6 (2003) 245–256.
[40] Z. Sun, Q. Song, X. Zhu, Using coding-based ensemble learning to improve software defect prediction, IEEE Trans. Syst. Man Cybern. C: Appl. Rev. 42 (2012) 1806–1817.
[41] A. Fernández, M.J. del Jesus, F. Herrera, On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets, Inf. Sci. 180 (2010) 1268–1291.
[42] Q. Yang, X. Wu, 10 challenging problems in data mining research, Int. J. Inf. Technol. Decis. Mak. 5 (2006) 597–604.
[43] I. Brown, C. Mues, An experimental comparison of classification algorithms for imbalanced credit scoring data sets, Expert Syst. Appl. 39 (2012) 3446–3453.
[44] D. Thammasiri, D. Delen, P. Meesad, N. Kasap, A critical assessment of imbalanced class distribution problem: the case of predicting freshmen student attrition, Expert Syst. Appl. 41 (2014) 321–330.
[45] V. López, A. Fernández, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics, Expert Syst. Appl. 39 (2012) 6585–6608.
[46] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inf. Sci. 250 (2013) 113–141.
[47] N. Tomašev, D. Mladenić, Class imbalance and the curse of minority hubs, Knowl.-Based Syst. 53 (2013) 157–172.
[48] V. López, A. Fernández, F. Herrera, On the importance of the validation technique for classification with imbalanced datasets: addressing covariate shift when data is skewed, Inf. Sci. 257 (2014) 1–13.
[49] R. Alejo, R.M. Valdovinos, V. García, J. Pacheco-Sanchez, A hybrid method to face class overlap and class imbalance on neural networks and multi-class scenarios, Pattern Recognit. Lett. 34 (2013) 380–388.
[50] T. Jo, N. Japkowicz, Class imbalances versus small disjuncts, ACM SIGKDD Explor. Newsl. 6 (2004) 40–49.
[51] K. Satyasree, J. Murthy, An exhaustive literature review on class imbalance problem, Int. J. Emerg. Trends Technol. Comput. Sci. 2 (2013) 109–118.
[52] D. Rodríguez, R. Ruiz, J.C. Riquelme, J.S. Aguilar-Ruiz, Searching for rules to detect defective modules: A subgroup discovery approach, Inf. Sci. 191 (2012) 14–30.
[53] L. Gonzalez-Abril, C. Angulo, F. Velasco, Gsvm: an svm for handling imbalanced accuracy between classes inbi-classification problems, Appl. Soft Comput. 17 (2014) 23–31.
[54] J. Derrac, I. Triguero, C.J. Carmona, F. Herrera, et al., Evolutionary-based selection of generalized instances for imbalanced classification, Knowl.-Based Syst. 25 (2012) 3–12.
[55] N. Japkowicz, The class imbalance problem: Significance and strategies, in: Proceedings of the 2000 International Conference on Artificial Intelligence, pp. 111–117.
[56] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: one-sided selection, in: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 179–186.
[57] J. Van Hulse, T. Khoshgoftaar, A. Napolitano, Experimental perspectives on learning from imbalanced data, in: Proceedings of the Twenty-fourth International Conference on Machine learning, pp. 935–942.
[58] A. Estabrooks, T. Jo, N. Japkowicz, A multiple resampling method for learning from imbalanced data sets, Comput. Intell. 20 (2004) 18–36.
[59] V. García, J.S. Sánchez, R. Martín-Félez, R.A. Mollineda, Surrounding neighborhood-based smote for learning from imbalanced data sets, Prog. Artif. Intell. 1 (2012) 347–362.
[60] S. Chen, H. He, E.A. Garcia, Ramoboost: ranked minority oversampling in boosting, IEEE Trans. Neural Netw. 21 (2010) 1624–1642.
[61] T. Khoshgoftaar, J. van Hulse, A. Napolitano, Comparing boosting and bagging techniques with noisy and imbalanced data, IEEE Trans. Syst. Man Cybern. A Syst. Hum. 41 (2011) 552–568.
[62] T. Khoshgoftaar, E. Geleyn, L. Nguyen, Empirical case studies of combining software quality classification models, in: Proceedings of the Third International Conference on Quality Software, pp. 40–49.
[63] C. Seiffert, T. Khoshgoftaar, J. van Hulse, Improving software-quality predictions with data sampling and boosting, IEEE Trans. Syst. Man Cybern. A Syst. Hum. 39 (2009) 1283–1294.
[64] V. Nikulin, G. J. McLachlan, S. K. Ng, Ensemble approach for the classification of imbalanced data, in: AI 2009: Advances in Artificial Intelligence, Springer, 2009, pp. 291–300.
[65] K. Ting, An instance-weighting method to induce cost-sensitive trees, IEEE Trans. Knowl. Data Eng. 14 (2002) 659–665.
[66] J. Zheng, Cost-sensitive boosting neural networks for software defect prediction, Expert Syst. Appl. 37 (2010) 4537–4543.
[67] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, in: Advances in Intelligent Computing, 2005, pp. 878–887.
[68] J. Xie, Z. Qiu, The effect of imbalanced data sets on lda: a theoretical and empirical analysis, Pattern Recognit. 40 (2007) 557–562.
[69] R. Barandela, R. Valdovinos, J. Sánchez, F. Ferri, The imbalanced training sample problem: under or over sampling? Struct. Syntactic Stat. Pattern Recognit. 3138 (2004) 806–814.
[70] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pp. 973–978.
[71] S. Knerr, L. Personnaz, G. Dreyfus, Single-layer learning revisited: a stepwise procedure for building and training a neural network, Neuro-comput. Algorithms Arch. Appl. 68 (1990) 41–50.
[72] T. Hastie, R. Tibshirani, Classification by pairwise coupling, Ann. Stat. 26 (1998) 451–471.

[73] S. Wang, X. Yao, Diversity analysis on imbalanced data sets by using ensemble models, in: IEEE Symposium on Computational Intelligence and Data Mining, 2009., IEEE, pp. 324–331.
[74] A. Liu, J. Ghosh, C. Martin, Generative oversampling for mining imbalanced datasets, in: Proceedings of the 2007 International Conference on Data Mining, pp. 25–28.
[75] M.A. Tahir, J. Kittler, A. Bouridane, Multilabel classification using heterogeneous ensemble of multi-label classifiers, Pattern Recognit. Lett. 33 (2012) 513–523.
[76] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, et al., Keel: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. – Fusion Found. Methodol. Appl. 13 (2009) 307–318.
[77] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Logic Soft Comput. 17 (2011) 255–287.
[78] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, The weka data mining software: an update, ACM SIGKDD Explor. Newsl. 11 (2009) 10–18.
[79] A.P. Bradley, The use of the area under the roc curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (1997) 1145–1159.
[80] J. Huang, C.X. Ling, Using auc and accuracy in evaluating learning algorithms, IEEE Trans. Knowl. Data Eng. 17 (2005) 299–310.
[81] J.A. Sáez, J. Luengo, F. Herrera, Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification, Pattern Recognit. 46 (2013) 355–364.
[82] H. He, Y. Ma, Imbalanced learning: foundations, Algorithms, and Applications, John Wiley & Sons, Hoboken, New Jersey, 2013.
[83] T. Fawcett, An introduction to roc analysis, Pattern Recognit. Lett. 27 (2006) 861–874.
[84] Z. Chi, H. Yan, T. Pham, Fuzzy algorithms: with applications to image processing and pattern recognition, vol. 10, World Scientific, 5 Toh Tuck Link, Singapore 596224, 1996.
[85] H. Ishibuchi, T. Yamamoto, T. Nakashima, Hybridization of fuzzy gbml approaches for pattern classification problems, IEEE Trans. Syst. Man Cybern. B Cybern. 35 (2005) 359–365.
[86] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (1945) 80–83.
[87] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (1937) 675–701.
[88] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[89] P. Nemenyi, Distribution-free multiple comparisons (Ph.D. thesis), Princeton, 1963.

Zhongbin Sun received the B.S. degree in computer science from the Xi'an Jiaotong University, Xi'an, China, in 2010. He is currently a doctor student in the Department of Computer Science and Technology, Xi'an Jiaotong University. His research focuses on software engineering and data mining.

Qinbao Song received the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2001. He is currently a Professor of software technology in the Department of Computer Science and Technology, Xi'an Jiaotong University, where he is also the Deputy Director of the Department of Computer Science and Technology. He is also with the State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China. He has authored or coauthored more than 80 referred papers in the areas of machine learning and software engineering. He is a board member of the Open Software Engineering Journal. His current research interests include data mining/machine learning, empirical software engineering, and trustworthy software.

Xiaoyan Zhu received her Ph.D. degree in computer science and technology from Xi'an Jiaotong University, China, in 2013. She is currently a University Lecturer of software technology in the Department of Computer Science and Technology, Xi'an Jiaotong University. Her current research interests include software engineering and data mining.

Heli Sun received the Ph.D. degree in computer science from Xi'an Jiaotong University, Shaanxi, China, in 2011. She is an assistant professor in the Department of Computer Science and Technology at Xi'an Jiaotong University. Her research interests include data mining and information retrieval.

Baowen Xu received the B.S., M.S., and Ph.D. degrees in computer science from Wuhan University, Huazhong University of Science and Technology, and Beihang University, respectively. He is a professor in the Department of Computer Science and Technology at Nanjing University. His main research interests are programming languages, software testing, software maintenance, and software metrics. He is a member of the IEEE and the IEEE Computer Society.

Yuming Zhou received the Ph.D. degree in computer science from Southeast University in 2003. From January 2003 to December 2004, he was a researcher at Tsinghua University. From February 2005 to February 2008, he was a researcher at The Hong Kong Polytechnic University. He is currently a professor in the Department of Computer Science and Technology at Nanjing University. His main research interests are software metrics, software evolution, and program understanding and analysis.