Wireless Pers Commun (2018) 102:937–950
https://doi.org/10.1007/s11277-017-5117-z

Imbalanced Data Processing Model for Software Defect Prediction

Lijuan Zhou[1] · Ran Li[1] · Shudong Zhang[1] · Hua Wang[1]

Published online: 14 December 2017

Abstract In the field of software engineering, software defect prediction is the hotspot of the researches which can effectively guarantee the quality during software development. However, the problem of class imbalanced datasets will affect the accuracy of overall classification of software defect prediction, which is the key issue to be solved urgently today. In order to better solve this problem, this paper proposes a model named ASRA which combines attribute selection, sampling technologies and ensemble algorithm. The model adopts the Chi square test of attribute selection and then utilizes the combined sampling technique which includes SMOTE over-sampling and under-sampling to remove the redundant attributes and make the datasets balance. Afterwards, the model ASRA is eventually established by ensemble algorithm named Adaboost with basic classifier J48 decision tree. The data used in the experiments comes from UCI datasets. It can draw the conclusion that the effect of software defect prediction classification which using this model is improved and better than before by comparing the precision P, F-measure and AUC values from the results of the experiments.

Keywords Software defect prediction · Class imbalance · Attribute selection · Sampling · Ensemble algorithm

1 Introduction

The quality of the software is one of the most important performance indicators in the software engineering. Therefore, the research in the field of software defect prediction is increasing day by day, and the reliability of the software will be a certain increase with the accurate prediction of software defects. The key to software defect prediction [1] is how to improve the accuracy of it, which is how to identify the modules of software defect

✉ Ran Li
1281276560@qq.com

[1] College of Information Engineering, Capital Normal University, Beijing, China

classification. If the modules which have software defects are divided into fault-prone modules and none-fault-prone modules, and then how to accurately find the existence of fault-prone modules in the software products is going to be a research focus nowadays. What software defect prediction can do is to allocate the limited resources to the fault-prone modules and achieve the optimal estimation of the classification, so as to find out the software defects.

There are numerous researches on learning about software defect prediction. For instance, Lessmann et al. [2] conducted a comparative study on the traditional methods of software defect prediction to demonstrate the accuracy of different methods, including Neural Networks, ensemble algorithms and SVM; Wang et al. [3] proposed a model of C4.5 decision tree based on Spearman correlation coefficient which be better used in software defect detection; Czibula et al. [4] focused on relational association rules to identify the module in the software is defective or not; Turhan et al. [5] used datasets from NASA to apply weighted Bayesian method to improve the accuracy of the classification in software defect prediction.

In the current researches of software defect prediction, the problem of how to solve the imbalance of datasets has become a hotspot. The number of fault-prone modules of the dataset is always much less than the number of none-fault-prone modules in most software systems, that is the fault-prone modules are called "few classes" [6], and the none-fault-prone modules are called "most classes". When software defect prediction is made for class imbalanced datasets, the classifier is biased towards higher accuracy for none-fault-prone modules which results in a large number of fault-prone modules being misinterpreted into "most classes". Finally, it will greatly reduce the accuracy of software defect prediction's classification, thus, the overall software development process will be affected to some extent. Based on the above, this paper will study the datasets for the existence of class imbalanced problem and utilize experiments to analyse how to improve the classification accuracy of software defect prediction.

2 Research Background

A detailed account of current related work is given below. It is mainly from the data level of sampling technology and algorithm level to solve the class imbalanced datasets recently, with using a variety of machine learning algorithms [7]. It is generally recognized that sampling technologies mainly includes: SMOTE [8], RUS [9] (random under-sampling) etc. Moreover, lots of articles focus on algorithms such as Boosting [10], Bagging [11]), cost-sensitive learning (e.g. AdaCost [12], MetaCost [13]) to utilize different classifiers to optimize accuracy of the software defect prediction at the algorithm level. With the development of machine learning, scholars have made a lot of researches in the field of software defect prediction and have obtained many valuable academic achievements while proposing many algorithms to solve the imbalance problem. Some particular examples for this are: Thwin et al. [14] proposed and constructed a software defect prediction model based on Generalized Regression Neural Network (GRNN) and Neural Network based on Ward in the meantime; Wang and Yao [15] reviewed the three types of unbalanced learning methods, such as random under-sampling method, cost-sensitive classification method and ensemble algorithm (AdaBoost [16])to analyse their applications in software defect prediction; Liu et al. [17] proposed a two-level cost-sensitive learning method to solve the problem of class imbalanced datasets to be more effective for the first time;

Khoshgoftaar et al. [18] achieved the improved accuracy, settling the problem of highly unbalanced data with the method of attribute selection.

As is known to all, this paper will concentrate on a new algorithm model and solution, taking the imbalanced datasets of software defect prediction as the research objects. Therefore, we take advantage of sampling technologies in the data level, with attribute selection and ensemble algorithm in the algorithm level to construct a software defect prediction model named ASRA so that it can solve the imbalanced issue in the datasets, promoting the classification accuracy.

3 Methodology

3.1 Ensemble Algorithm Adaboost

The Adaboost algorithm is an iterative algorithm developed by the Boosting algorithm, proposed by Freund and Schapire et al. [19]. Unlike the Boosting algorithm, AdaBoost algorithm does not need to know the lower limit of learning accuracy of weak learning algorithms in advance. As well as, the classification accuracy of strong classifiers depends on the classification accuracy of all weak classifiers, which can dig deeper into the ability of each weak classifier. The generalization of integrated learning is better than a single base classifier generally. The algorithm is designed to fuse the weak classifiers into strong classifiers, which means increasing the previous classifier weight of the wrong sample gradually, and then use the weighted data for training. Simultaneously, the algorithm adds a weak classifier in each iteration until the minimum error rate or the maximum number of iterations is satisfied. The emphases of Adaboost are the sample weight updating and the weak classifier weighted combination. Therefore, it ensures that the maximum value of the classification error rate decreases as the number of training increases which means that the weak classifier with low error rate takes a larger weight in the final classifier, otherwise smaller. The specific algorithm steps are organized as follows:

a. Initialize the sample weight: setting each sample in the training datasets to the same weight about 1/n, where n is the number of samples in the training sets as well as the maximum number of iterations given is N rounds.
b. Perform multiple rounds of iterations: selecting the appropriate threshold and adjusting the weight of the samples according to the classification error rate.
c. Generate a strong classifier: generating a training dataset for the classifier of the next iteration process due to the new sample weights. In consequence, an integrated classifier with better prediction performance is generated after N iterations, that is, the strong classifier.

The algorithm Adaboost has various basic classifiers and commonly uses J48, Id3 or Naïve Bayes. In this paper, we propose the model ASRA which will adopt J48 classifier to experiment the whole datasets, as well as utilizing the J48 classifier as the basic control group to compare the consecutive experimental results.

3.2 Attribute Selection

Attribute selection is one of the hot topics in the field of pattern recognition nowadays. From a point of view, attribute is the key to determine the similarity between samples and the design of classifiers. The meaning of Attribute selection is to remove the irrelevant and

redundant feature attributes from all the features in the datasets, accomplishing the effect of data dimension reduction, selecting the optimal set of attributes according to some evaluation criteria so that it can make the corresponding model and algorithm to have good performance [20]. To achieve the most simple and convenient classification prediction about software defect, we should make a comprehensive choice of attribute selection from some factors (the question scale, the number of samples etc.) There are a lot of definitions for the attribute selection algorithm at present and based on this, we may reasonably arrive at the main steps of attribute selection algorithm:

a. Generate a subset of candidate features: first of all, searching based on different starting points, such as: forward searching, backward searching and so on. Furthermore, generating a subset of searches based on a particular search strategy, such as exhaustive searching, sequence searching, and random searching.
b. Evaluate the features of subsets: when the candidate attribute subset is generated, it is necessary to evaluate them according to its assessment. Meanwhile, models are roughly divided into three categories: wrapper model, embedding model, filter model. The wrapper model uses classification results based on subset-algorithm as feature evaluation basis. The other one embedding model is that the process of attribute selection and the training process of classifiers are completed in the same optimization stage, and so that the two processes are better integrated. This review focuses on the filter model during the attribute selection of data pre-processing. The filter model evaluates the attributes through the characteristics of the data and it has nothing to do with the selection of the classifier or the classification algorithm. Meanwhile, the filter model can quickly rule out a large number of non-critical noise characteristics and can narrow the range of optimization feature subset searching. The steps of the filter model are: ① Using original datasets to do the feature sorting. ② Adopting the attribute selection with the evaluation criteria and generating the subsets. ③ Using classifiers to get the results in the end. As a result, using the filter model during the data pre-processing stage can prevent the selection of classifiers from affecting the experimental results of software defect prediction. The filter model's framework about attribute selection shown in Fig. 1.
c. Stop criteria: The two steps above all go through multiple rounds of iteration until satisfying the standard. The specific evaluation and the application requirements of the software have a significant impact on the setting of the stop criteria. Besides, the most common stop criteria are: the size of the feature subset, the error threshold and the number of operations performed by the algorithm.

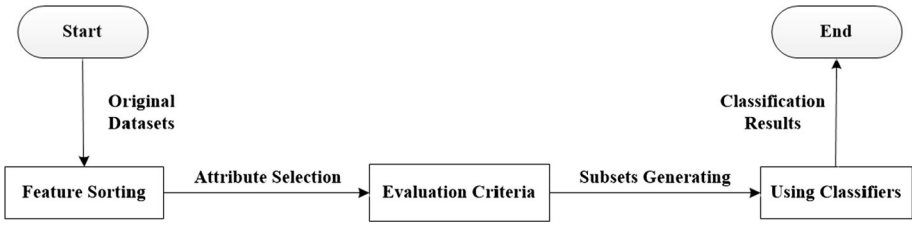

Fig. 1 Framework of attribute selection model based on filter

3.3 Sampling Technologies

3.3.1 Over-Sampling

It is generally agreed that over-sampling means increasing the number of samples to balance the proportions of classes in the dataset. The SMOTE algorithm is one of the most classic over-sampling algorithms, which means linear interpolation is performed between "few classes" of similar samples to generate new data instances. The main idea of SMOTE is to choose the number of k "few classes" from the nearest neighbour distance (k generally has a value of 5) for each class of sample x in the imbalanced datasets, and then to select n samples from x and k nearest neighbour samples randomly. Besides, performing the linear interpolation on each nearest neighbour sample to generate new data samples as synthetic "few classes" which can adjust the proportion of the datasets to balance. Thus, we utilize SMOTE as the over-sampling method to increase the number of samples to balance the datasets. However, its main disadvantage is that the training time will gradually increase as the number of samples grows, decreasing the efficiency of the software defect prediction model to a certain extent. Apart from this, SMOTE also may lead to the classifier learning over-fitting and multiple rules for multiple copies of the same sample will cause that the rules are too specific to accomplish.

3.3.2 Under-Sampling

Under-sampling refers to the removal of the number of samples in order to achieve the balance of classes in the dataset. The practical technologies of it are Tomek-links (remove noise points or boundary points from data), ENN (remove those data that differ from the two samples in the three most recent samples) and Resample. Resample is used widely and its main idea is to remove "most classes" samples randomly and retain the defective samples of software defects completely to solve the problem of imbalanced data in the software defect prediction. Therefore, we use the Resample method as the under-sampling technique in the review. However, there is a disadvantage of under-sampling technique that deleting some data in some classes may result in loss of critical information to some extent.

4 Content of ASRA Model

In order to solve the problem of imbalanced class in most datasets, this paper proposes an algorithm named ASRA model for software defect prediction. We use the WEKA platform to experiment a combination of attribute selection and sampling techniques in the data preprocessing phase. Firstly, this model utilizes the Chi square test of attribute selection algorithm on the datasets of software defect prediction, deleting the redundant and unrelated attributes to achieve dimensionality reduction. We adopt the Chi square method in attribute selection which is a feature weight algorithm commonly used. What's more, the practical meaning of Chi square test is that it is a hypothetical test which is usually used to test the relevance of the two variables and evaluate whether two events are independent or relevant. When describing the degree of deviation between the actual and theoretical values, the larger the Chi square value, the weaker the independence of the two variables, and the stronger the independence of the two variables on the contrary. We assume that the independent variables have N values and there are M values for the dependent variables.

Meanwhile, we set the theoretical value of E and the actual value of x_i, thus, the degree of deviation to determine the Chi square value χ^2can be expressed as:

$$\chi^2 = \sum \frac{(x_i - E)^2}{E} \tag{1}$$

After determining the independence of the variables based on the Chi square test, we set the threshold to remove irrelevant and redundant features in the datasets. Additionally, we adopt the SMOTE method to add "few classes" samples and then use the Resample technique to balance the datasets according to different over-sampling rates and under-sampling rates, namely "Combined Sampling" method in this paper. Afterwards, the model classifies the results from the experiments in the light of the J48 classifier of ensemble algorithm Adaboost by ten-fold cross validation until the experiment satisfies the condition of iterations. Consequently, we design experimental control groups to verify the advantages of ASRA model, and the research demonstrates that the accuracy of the software defect prediction model has been improved obviously. The flow chart of the algorithm is shown in Fig. 2.

5 Experimental Work

5.1 Experimental Datasets Selection

In this paper, we have utilized the UCI data collections from the University of California, Irvine. On the basis of these conditions about the number of attributes of the datasets, the total number of samples and the proportion of imbalanced class, we have selected five datasets includes two categories for the software defect prediction. To detail this, there are five datasets: ① Credit-g: called Statlog-German Credit Dataset, which means German credit data that is used to predict whether personal credit is good or bad and the "bad" class for the "few classes". ② Diabetes: called Pima Indians Diabetes Dataset, which is based on medical history to predict Pima Indians nearly 5 years of diabetes incidence and the two categories are "tested-negative" and "tested-positive", using the "tested-positive" as the "few classes". ③ Hepatitis: represents the liver dataset that is used to predict the

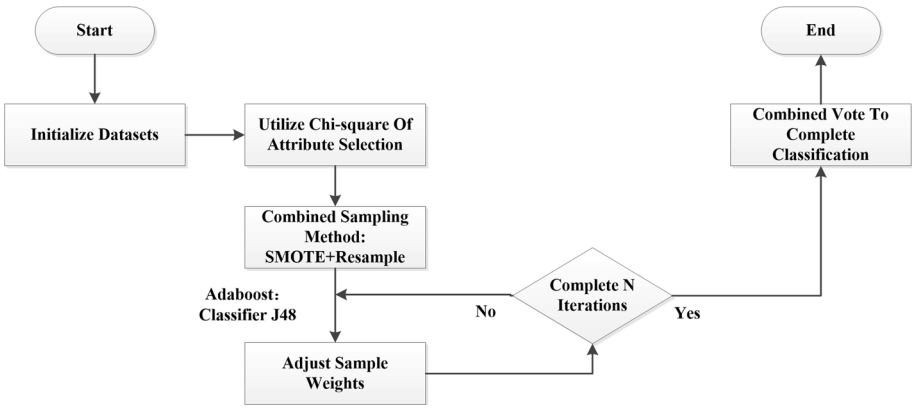

Fig. 2 The flow chart of ASRA algorithm

incidence of hepatitis and two categories are "DIE" and "LIVE", utilizing the "DIE" for the "few classes". ④ Ionosphere: on the behalf of the ionospheric dataset, which predicts the atmospheric structure according to the free ion radar echo in a given ionosphere. It has two categories that are "g" and "b", representing "good" and "bad" and using "b" as the "few classes". ⑤ Sick: called datasets of sick and two categories are "sick" and "negative", using the "sick" as the "few classes". The specific information of the datasets selected in the experiment is shown in Table 1.

5.2 Experimental Evaluation Measures

Regarding to the ASRA model in the paper, this experiment uses the traditional classifier performance evaluation measures. As a matter of fact, we classify the samples of datasets into four categories based on their actual category and classifier predictions: TP (true positive), FP (false positive), TN (true negative), FN (false negative), and TP + FP + FN + TN = Total number of dataset samples which uses "few classes" as positive examples standing for the defective software [21]. At the same time, the classification results can be used as shown in Table 2.

In the actual situation, the goal of the software defect prediction model is to find the defective software to the maximum extent under the existing conditions and to minimize the misclassification of the none-defective software modules into the defective categories. The formula for the precision rate P and the recall rate R is defined as follows:

$$P = \frac{TP}{TP + FP} \tag{2}$$

$$R = \frac{TP}{TP + FN} \tag{3}$$

However, it is not enough to reflect the accuracy of classification about software defect assessments merely by the two kinds of performance evaluations of precision rate P and recall rate R, especially for the class imbalanced datasets. Therefore, the experiment utilizes three evaluation measures (precision rate P, F-measure, AUC-ROC area) to assess the performance of ASRA algorithm. The formula for the F-measure value is defined as follows:

Table 1 Information of the datasets

Dataset	Number of samples	Number of attributes	Number of few classes	Proportion of few classes (%)
Credit-g	1000	21	300	30.0
Diabetes	768	9	268	34.9
Hepatitis	155	20	32	20.6
Ionosphere	351	35	126	35.9
Sick	3772	30	231	6.1

Table 2 Confusion matrix

Actual situation	Predict classification results	
	Predict defective	Predict none-defective
Defective	TP	FN
None-defective	FP	TN

$$F\text{-}measure = \frac{(1+\beta^2) \times P \times R}{(\beta^2 \times P) + R} \tag{4}$$

The F-measure value is based on the weighted sum of the precision rate P and the recall rate R, which is comprehensive of results according to the P and R. Hence, F-measure has a better accuracy for the classification results of the performance evaluation in the software defect prediction. Where β is the relative importance of the P and R, we set the value of β is 1 in this experiment.

ROC [22] represents Receiver Operating Characteristic curve, describing the learning performance evaluation comparisons of different classifiers in the field of machine learning [23]. It sorts the test samples according to the results of different classifiers and calculates them to predict one by one as a positive example. The ROC curve is obtained by plotting the result as a vertical and horizontal coordinate. The abscissa axis is the "false positive rate" (FPR), the ordinate axis is "true positive rate" (TPR). Both are defined as:

$$FPR = \frac{FP}{TN + FP} \tag{5}$$

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

In the actual cases, the AUC value of the lower area surrounded by the ROC curve and the coordinate axis is used as a practical evaluation measure instead of the ROC curve. On the other hand, the AUC value combines the false positives and probabilities of software defect prediction [24]. When the AUC value is larger, the software defect prediction model has better performance. Besides, the AUC value of the stochastic prediction model is 0.5, and the ideal model is 1.

5.3 Experimental Design and Empirical Results

In this paper, the experimental platform uses WEKA (Waikato Environment for Knowledge Analysis), which can be more convenient to use this model for experiments. Firstly, the part of attribute selection in the experiment is Chi square method, reducing the training time by decreasing the number of features with the idea of hypothetical test and setting the threshold to 0. Secondly, the combined sampling technique uses the classical SMOTE method with the neighbourhood value of 5 and the Resample under-sampling method through setting different rates of SMOTE and Resample about each dataset. Moreover, this experiment adjusts the total number of samples to be half of the original dataset to lessen the training time in the end. Furthermore, the model combines with the ensemble algorithm Adaboost, selecting the J48 classifier of the C4.5 decision tree as a base classifier to construct a software defect prediction named ASRA model in the review.

In the data pre-processing stage, we adopt the combined sampling technique. The purpose of using Resample method to reduce the number of experimental samples to be

half of the original datasets is to enhance the efficiency of classification and prediction about software defect. The different rates of SMOTE and Resample corresponding to each imbalanced dataset are calculated according to the ratio above. The rates of the datasets in the data pre-processing phase is set as shown in Table 3.

Besides, all experiments in this paper adopts ten-fold cross validation and takes each dataset to divide into 10 subsets, 9 of which are used as training sets and the one as test dataset to ensure the objectivity and reliability of the final predictive classification test results. This review compares three methods to consider the classification in software defect prediction: one is to classify by J48 classifier only, the other is to classify by data pre-processing stage and the use of J48 classifier, the last method is to adopt ASRA model proposed in the paper. Accordingly, we utilize the precision rate P, F-measure and AUC values to reflect the contrastive results in the all experiments. The experimental designs of ASRA algorithm are divided into the following three categories:

Original: To use of C4.5 decision tree with J48 classifier to predict and classify.
ASR: To perform the attribute selection and then use the sampling techniques, utilizing the J48 classifier to classify afterwards.
ASRA: To perform the attribute selection and then use the combined sampling technique with utilizing the J48 classifier of Adaboost ensemble algorithm to predict the defects.

The experimental datasets of the above three groups are shown in Tables 4, 5 and 6, and the comparison charts of the experimental performances are shown in Figs. 3, 4 and 5.

The above-mentioned Tables 4, 5 and 6 and Figs. 3, 4 and 5 show the three methods of evaluating the effect of the classification prediction performance using the precision P, F-measure and AUC values respectively. From the above experimental results can be found, the ASRA algorithm has the highest F-measure value and AUC value on these five datasets in the paper. In addition to the hepatitis dataset, the precision rate P on the other datasets reaches the highest value. After analysing the experimental results, we found that the reason of the phenomenon is the small number of hepatitis datasets, the model proposed in this paper can not accurately obtain the dataset sample characteristics for classification prediction when the Adaboost algorithm is adopted so that resulting in the data under-fitting and making the experimental results less than the performance of other data sets. However, we get the better experimental results from the ASRA model than the other groups (original and ASR) in generally so that we can see the accuracy of classification about the software defect prediction has been improved.

In short, it can be seen from this that the classification about software defect prediction of ASRA model is superior to its contrast groups. The ASRA model uses different algorithms of machine learning and sampling technologies to solve the problem of class

Table 3 The information of the rates for SMOTE and resample

Dataset	Rate of SMOTE (%)	Rate of resample (%)	Original number	Present number
Credit-g	55.60	42.88	1000	499
Diabetes	24.38	46.10	768	384
Hepatitis	156	37.7	155	76
Ionosphere	19.05	46.79	351	175
Sick	855.27	30.81	3772	1770

Table 4 Precision rate P of three designs in the five datasets

Dataset	Original	ASR	ASRA
Credit-g	0.511	0.723	0.728
Diabetes	0.632	0.808	0.811
Hepatitis	0.667	0.783	0.778
Ionosphere	0.929	0.953	0.969
Sick	0.919	0.953	0.970

Table 5 F-measure of three designs in the five datasets

Dataset	Original	ASR	ASRA
Credit-g	0.442	0.661	0.728
Diabetes	0.614	0.743	0.814
Hepatitis	0.528	0.783	0.840
Ionosphere	0.874	0.917	0.940
Sick	0.901	0.964	0.976

Table 6 AUC of three designs in the five datasets

Dataset	Original	ASR	ASRA
Credit-g	0.639	0.734	0.857
Diabetes	0.751	0.852	0.923
Hepatitis	0.708	0.802	0.970
Ionosphere	0.892	0.921	0.959
Sick	0.951	0.984	0.995

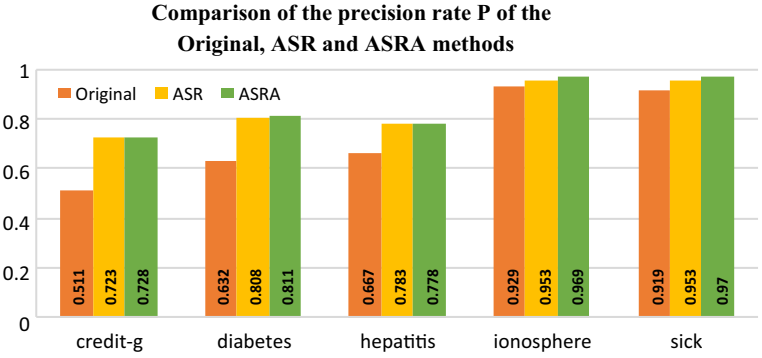

Fig. 3 Comparison of the precision rate P of the original, ASR and ASRA methods

imbalanced datasets, decreasing the imbalance of the overall datasets to raise the accuracy of classification.

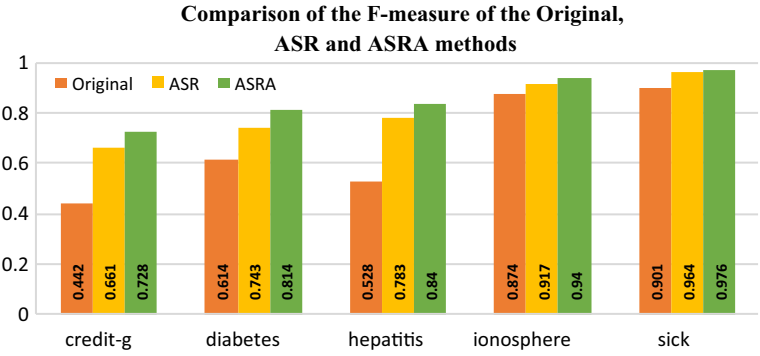

Fig. 4 Comparison of the F-measure of the original, ASR and ASRA methods

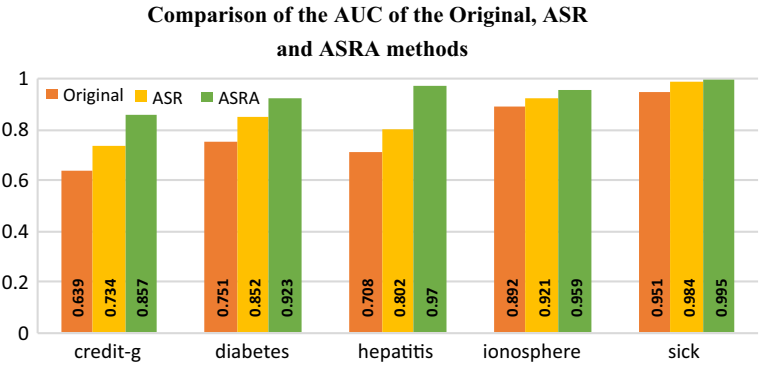

Fig. 5 Comparison of the AUC of the original, ASR and ASRA methods

6 Conclusion

In this paper, we constructed the ASRA model with machine learning which combined with attribute selection, SMOTE and Resample technique to handle the imbalanced datasets and utilized the ensemble algorithm Adaboost to predict the classification of software defect by J48 basic classifier. Moreover, we got that the ASRA model has the highest F-measure value and AUC value on the selected five class imbalanced datasets of UCI, and the precision rate P also had high values from the experiments, which means that the software defect prediction performance of the ASRA model is improved obviously and the model we constructed is more effective and pratical in software defect prediction.

For the further work, we plan to extend our model with other approaches, considering to integrate distinct algorithms in software defect prediction with machine learning. Furthermore, we intend to apply more algorithms such as Naive Bayes, Bagging to experiment the predictions and utilize more different classifiers not only the J48 that may be relevant to the classification results. We will also use more methods of attribute selection to verify the feature is relevant or redundant, adopting different assessment such as information gain, mutual information. Consequently, we hope that the model ASRA proposed in the paper will have valuable meaning for the future developments and advancements in the field of software defect prediction.

Acknowledgements The authors would like to thank the editors and all of the reviewers through the development for this work. Special thanks are according to the authors and experts from the UCI Machine Learning Repository who made the numerous datasets that were used in the experimental chapters in this paper. "The computer application technology" Beijing municipal key construction of the discipline. The authors acknowledge the support of Capital Normal University during the thesis is completed and the National Nature Science Foundation (Grant: 61601310).

References

1. Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering, 38*(6), 1276–1304.
2. Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering, 34*(4), 485–496.
3. Wang, J., Shen, B., & Chen, Y. (2012). Compressed C4.5 models for software defect prediction. In *International conference on quality software* (Vol. 430, pp. 13–16). IEEE.
4. Czibula, G., Marian, Z., & Czibula, I. G. (2014). Software defect prediction using relational association rule mining. *Information Sciences, 264*(183), 260–278.
5. Turhan, B., & Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study ☆. *Data & Knowledge Engineering, 68*(2), 278–290.
6. Weiss, G. M. (2004). Mining with rarity: A unifying framework. *ACM SIGKDD Explorations Newsletter, 6*(1), 7–19.
7. Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing Journal, 27*(C), 504–518.
8. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*(1), 321–357.
9. Tahir, M. A., Kittler, J., & Yan, F. (2012). Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognition, 45*(10), 3738–3750.
10. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews, 42*(4), 463–484.
11. Rodriguez, D., Herraiz, I., Harrison, R., Dolado, J., & Riquelme, J. C. (2014). Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In *ACM international conference on evaluation and assessment in software engineering* (pp. 1–10).
12. Fan, W., Stolfo, S. J., Zhang, J., & Chan, P. K. (1999). AdaCost: Misclassification cost-sensitive boosting. In *Sixteenth international conference on machine learning* (Vol. 33, pp. 97–105). Morgan Kaufmann Publishers Inc.
13. Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. In *ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 155–164). ACM.
14. Quah, T. S., & Thwin, M. M. T. (2003). Application of neural networks for software quality prediction using object-oriented metrics. In *International conference on software maintenance* (Vol. 76, pp. 116). IEEE Computer Society.
15. Wang, S., & Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability, 62*(2), 434–443.
16. Rätsch, G., Onoda, T., & Müller, K. R. (2001). Soft margins for Adaboost. *Machine Learning, 42*(3), 287–320.
17. Liu, M., Miao, L., & Zhang, D. (2014). Two-stage cost-sensitive learning for software defect prediction. *IEEE Transactions on Reliability, 63*(2), 676–686.
18. Khoshgoftaar, T. M., Gao, K., & Hulse, J. V. (2012). Feature selection for highly imbalanced software measurement data. *Recent trends in information reuse and integration* (pp. 167–189). Vienna: Springer.
19. Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In: *Computational learning theory* (Vol. 55(1), pp. 119–139) Berlin, Heidelberg: Springer.
20. Khoshgoftaar, T. M., & Gao, K. (2009). Feature selection with imbalanced data for software defect prediction. In *International conference on machine learning and applications* (pp. 235–240). IEEE Computer Society.

21. Mandal, P., & Ami, A. S. (2015). Selecting best attributes for software defect prediction. In *IEEE international wie conference on electrical and computer engineering* (pp. 110–113). IEEE.
22. Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861–874.
23. Fawcett, T. (2003). ROC graphs: Notes and practical considerations for data mining researchers. *Machine Learning, 31*(8), 1–38.
24. Huang, J., & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering, 17*(3), 299–310.

Lijuan Zhou Professor/Capital Normal University/Beijing, China. 1987–1991: Bachelor of computer software, Heilongjiang University. 1995–1998: Master of computer science and technology, Harbin University of Science and Technology. 1999–2004: Ph.D. of computer science and technology, Harbin Engineering University. Presided over the national science and technology support program project, the Beijing Municipal Commission of Education Technology Development Plan, the provincial key scientific and technological projects, natural science funds, science and technology research projects and other research projects more than 20. Research interests: Intelligence Analysis, Data mining and Data analysis, Business Intelligence and Artificial Neural Network.

Ran Li Master degree candidate/Capital Normal University/Beijing, China. 2012–2016: Bachelor of Information management and information system, Capital Normal University. 2016–present: Master degree candidate of computer applied technology, Capital Normal University. Research interests: Data mining and Data analysis, Machine Learning and Software Engineering.

Shudong Zhang Professor/Capital Normal University/Beijing, China. Bachelor of Engineering degree-Beijing Institute of Technology in 1993. Master of Science degree-China Academy of Engineering Physics in 1996. Ph.D. degree in engineering-Beijing Institute of Technology in 2005. 2005–2007: Worked as a postdoctoral fellow at the Chinese Academy of Sciences Research work, research direction for the community broadband network and sensor network system. 2007–present: assigned to Capital Normal University School of Information Engineering work. Research interests: Network and distributed systems, Computer software.

Hua Wang Associate Professor/Capital Normal University/Beijing, China. 1987: the Soviet Union sent to study. 1993: in Ukraine Kiev Institute of Technology received a doctorate returned. 1993.7–1996.1: engaged in postdoctoral research in the computer science department of Beijing University of Aeronautics and Astronautics. Worked in the enterprise for many years and participated in and presided over a number of ministerial-level projects, mainly engaged in the research and implementation of application systems. Research interests: Data Mining, Software Engineering.