International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 4, August 2019, pp. 3241~3246
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i4.pp3241-3246

Ensemble learning for software fault prediction problem with imbalanced data

Thanh Tung Khuat, My Hanh Le
Information Technology Faculty, The University of Danang, University of Science and Technology, Vietnam

Article Info

Article history:

Received Oct 10, 2018
Revised Mar 19, 2019
Accepted Apr 3, 2019

Keywords:

Classifier
Data sampling
Ensemble learning
Random under sampling
Software fault prediction

ABSTRACT

Fault prediction problem has a crucial role in the software development process because it contributes to reducing defects and assisting the testing process towards fault-free software components. Therefore, there are a lot of efforts aiming to address this type of issues, in which static code characteristics are usually adopted to construct fault classification models. One of the challenging problems influencing the performance of predictive classifiers is the high imbalance among patterns belonging to different classes. This paper aims to integrate the sampling techniques and common classification techniques to form a useful ensemble model for the software defect prediction problem. The empirical results conducted on the benchmark datasets of software projects have shown the promising performance of our proposal in comparison with individual classifiers.

Corresponding Author:

My Hanh Le,
Information Technology Faculty,
University of Danang, University of Science and Technology,
54 Nguyen Luong Bang, Danang, 550000, Viet Nam.
Email: ltmhanh@dut.udn.vn

1. INTRODUCTION

In the past few years, researchers have put more effort into software property prediction problems such as effort estimation [1], defect classification [2], and software quality prediction [3]. While the risk of defects within the software modules under development are high, testing operations [4] are time-consuming and expensive, and they cannot be performed for entire elements. As a result, accurate prediction of faults in software units might help managers to allocate limited time and precious resources to deploy an efficient software testing process. Along with the advancement of machine learning techniques, various software metrics have been used to construct predictive models for identifying fault-prone software modules such as static code metrics, execution traces, and historical code changes [5]. This work also employs the static code metrics including class-level and method-level ones to build software fault classifiers.

One of the features of software quality datasets is the imbalance between the number of patterns in each class label, where most vulnerable components of the software system may only be sought with a small ratio. Therefore, the quantity of faulty samples in such software datasets is much lower than that of non-defective patterns [6]. Unfortunately, the performance of most conventional classifiers, like support vector machines [7], K-nearest neighbor [8], neural networks [9], and Bayesian network [10], is significantly decreased on the class-imbalance problem. They are usually towards the dominant class and tend to disregard the minority class, and this phenomenon is possible to lead to high false negative rates [11]. To solve this problem, data sampling methods are regularly adopted combined with predictive models. This paper makes use of random undersampling (RUS) to cope with the imbalanced data problem. We first produce the balanced datasets by utilizing the RUS techniques for an original imbalanced dataset. These balanced datasets are then put to various base predictors and finally, a specific ensemble rule is deployed to combine the classification results of these base models.

Theoretical and empirical evidence has indicated that the ensemble method of multiple classifiers may make the ultimate predictive model more accuracy [12]. Nevertheless, there are very few studies applying the ensemble approach to the software defect prediction problem [13]. To the best of our knowledge, this work is the first study on empirical assessment of the influence of sampling on ensemble models concerning imbalanced training datasets for the software fault prediction problem. The principal aim of this paper is to reveal the vital role of the sampling technique to the accuracy of the ensemble classifier on imbalanced data. We use a software defect ensemble predictor consisting of five base classifiers: k-nearest neighbor, Bayesian networks, J48, multilayer perceptron, and support vector machines. The diversity in classification abilities of the base classifiers may contribute to capturing different statistical characteristics of the underlying data. Empirical results are performed on seven software defect datasets from the PROMISE repository [14]. Our main contributions in this paper can be summarized as follows:

- We propose a general method of building an ensemble model of base classifiers for software fault prediction using imbalanced training datasets
- We assess the crucial role of the under-random sample technique on improving the performance of the ensemble models through experimental results on highly imbalanced software fault datasets

The remainder of this paper is outlined as follows: section 2 presents the background knowledge and related work of the random undersampling and ensemble learning. Section 3 discusses our proposed method, while section 4 the analysis of experimental results. The conclusion and future work are given in section 5.

2. BACKGROUND

2.1. Software fault prediction

Defect prediction is a method of early identification of faults in software modules. It investigates the properties of individual code elements to determine those units being fault-prone or not [15] or to predict the number of faults in each component [16]. While the latter considers software defect prediction as a regression issue, the former approach regards it as a classification problem. This study only deals with the classification viewpoint, which predicts a software module into fault-prone or non-fault-prone. A large number of static code characteristics have been proposed for the software fault prediction ranging from method level metrics such as Lines Of Code-based measures [17], McCabe [18] and Halstead [19] metrics to class level metrics like Chidamber-Kemerer [20] and Conceptual Cohesion of Classes measure [21].

Based on static code metrics, researchers have adopted different methods to construct software fault prediction models. In general, conventional defect prediction approaches consist of four main steps, i.e., construction of training datasets, feature extraction from software defect datasets, development of a predictive model, and the application of the constructed model.

2.2. Class imbalance problem and random under sampling

Class imbalance is an integral attribute of the software defect data, which comprise only a few faulty units and a large number of non-faulty modules [22]. This characteristic has a considerable impact on both the training of a model and the predictive performance since most machine learning algorithms tend to form classifiers maximizing the overall classification accuracy. Consequently, the valuable minority class is usually ignored by such models. For example, given a dataset having only 1% of the faulty components, an overall accuracy of 99% might be easily attained by a binary classifier grouping all data patterns as non-faulty patterns. As a result, the minority defective instances are all misclassified with this simple model. In this case, it outputs a very high accuracy, but it makes no sense. Therefore, the class imbalance problem often diminishes the binary predictors, and further makes these classification models not to predict the minority faulty software units accurately.

Many studies have been introduced to handle the class imbalance problem. A survey of techniques for reducing the negative impact of imbalance on classification performance was proposed by Weiss et al. [23]. Crucial methods for alleviating the influence of class imbalance might be categorized into groups, namely external and internal methods. Internal techniques aim to modify existing machine learning algorithms for reducing their sensitiveness to class imbalance [24], while the external approach tends to form a balanced training dataset. The external approaches are widely used as they are independent of the underlying classification algorithms. Data sampling belongs to the external group. The undersampling technique often eliminates samples of the majority class for obtaining a balanced dataset before training the classifiers. Mani and Zhang [25] pointed out that the random undersampling technique regularly outperforms other complex sampling strategies. Therefore, we use random undersampling in comparison with base classifiers to build an ultimate ensemble model.

3. PROPOSED ENSEMBLE MODEL

In our proposed model, each base classifier is trained on a different balanced dataset formed from the sampling step, and the model includes three components: data balancing, classifiers training, and classifying. The details are shown in Figure 1.

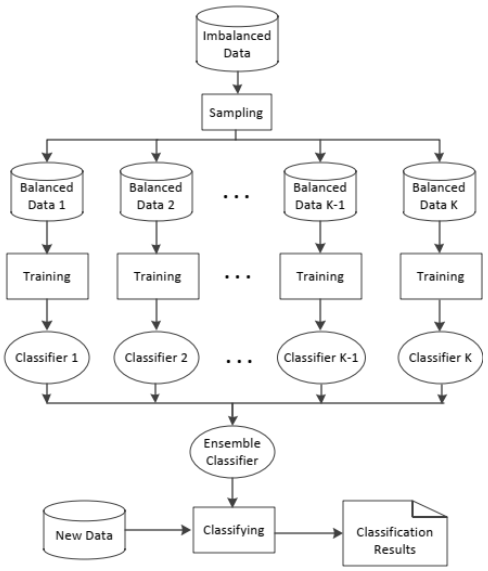

Figure 1. Proposed ensemble classifier

During the training process, the majority class samples in the original imbalanced dataset are split into several bins by adopting the random undersampling method. Each bin includes the equal number of patterns to that of the minority class, and then all minority class patterns are put into each bin to form the balanced training dataset. After that, each base classifier will be trained on a separated balanced dataset by a specific classification algorithm. Finally, the final classifier is built by combining the outcomes of base predictors relied on the majority voting rule. The ensemble model would then be deployed to classify new data. There are various classification techniques possible to be used for base classifiers. The diversity of base predictors might result in the performance improvement of the final ensemble model. In this study, we use five common classification algorithms, including support vector machines (SVM) [7], multilayer perceptron (MLP) [9], Bayesian networks [10], K-nearest neighbor (KNN) [26], and decision tree J48 [27]. Diversity is a crucial factor in the ensemble members' decisions. It can be seen that base learners are trained on different datasets, and this will contribute to the diversity of the final ensemble model formed from the majority voting rule for outcomes of base classifiers.

4. RESULTS AND ANALYSIS

4.1. Empirical evaluation criteria and dataset

Each binary classification issue is associated with four possible prediction cases, i.e., true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). As for the software defect prediction, if a sample is classified as "faulty" and is actually "faulty", it is a true positive; if a non-faulty pattern is misclassified as "faulty", it is a fault positive. In a similar way, true negative shows that the non-faulty sample is predicted to "non-faulty," while fault negative indicates an error situation where a buggy program unit is incorrectly grouped as "non-buggy". Based on these four variables, measures such as Precision, Recall, and F1-score are computed as follows:

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

$$F1 - score = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$$

To evaluate the effectiveness of the ensemble classifier, we conducted experiments on a collection of seven highly imbalanced binary datasets from the PROMISE repository of software defect databases [14] These seven open source datasets have the different number of patterns, features, and the class imbalance ratio. Table 1 shows the attributes of each selected imbalanced dataset, including the total number of attributes (#Attr.), the number of patterns (# Pats.), the number of defective components (# Defect), the number of non-defective units (# Non-defect), the ratio of faulty modules to all modules in each dataset (% Defect). All seven software systems have been written in Java programming language. Each instance in these datasets represents a single Java class. The feature set of each dataset consists of 20 software metrics such as complexity, coupling, cohesion, size and defect proneness characteristics of a Java class.

Table 1. Summary of seven highly imbalanced datasets

Dataset	# Attr.	# Pats	# Defect	# Non-defect	%Defect
Ant 1.7	20	745	166	579	22.28%
Camel 1.6	20	965	188	777	19.48%
Ivy 2.0	20	352	40	312	11.36%
Poi 2.0	20	314	37	277	11.78%
Tomcat	20	858	77	781	8.97%
Xalan 2.4	20	723	110	613	15.21%
Synapse 1.2	20	256	86	170	33.59%

4.2. Experimental results

4.2.1. Comparison of the ensemble models with and without using random undersampling

This part is to uncover if the undersampling-based ensemble model can handle the class imbalance problem more efficient compared with one without using the undersampling technique. Non-sampling ensemble means that base classifiers are trained on the entire original imbalanced dataset. Table 2 shows the average results of F1-score over 30 execution times for the non-sampling and undersampling ensemble models. In the table, the best value of each dataset is highlighted in bold.

From Table 2, it is observed that the integration of the random undersampling method with ensemble learning outperforms the ensemble classifier without using the sampling technique in all imbalanced datasets, especially for the *poi 2.0* dataset. In this dataset, the ensemble predictor trained on the original imbalanced data outputs the F1-score value being completely inaccurate, while the ensemble model using the random undersampling algorithm significantly enhances the accuracy of F1-score. These results indicate that sampling technique contributes to the considerable improvement of the accuracy of the ensemble classifier regarding the class imbalance training datasets.

Table 2. Average F1-score values of imbalanced datasets for the ensemble models

Dataset	Non-sampling ensemble	UnderSampling ensemble
Ant 1.7	0.5278	0.6261
Camel 1.6	0.2321	0.4413
Ivy 2.0	0.2759	0.3937
Poi 2.0	0	0.3354
Tomcat	0.2222	0.3899
Xalan 2.4	0.2535	0.4535
Synapse 1.2	0.5333	0.6487

4.2.2. Comparison of the ensemble model and its base classifiers

The purpose of this experiment is to validate whether the ensemble model using the random undersampling lead to better average F1-score values compared to their base classifiers. Table 3 describes the results of the ensemble model and its base classifiers when trained on the original imbalanced data and balanced datasets. The best results for each dataset are highlighted in bold.

Generally, base predictors trained on balanced data output much better average F1-score results over all datasets compared to those trained on original imbalanced datasets, especially kNN, MLP, and SVM. It is easy to observe that several classifiers such as J48, SVM, and kNN are very sensitive to imbalanced data, and they generate incredibly inaccurate F1-score values. When adopting the original imbalanced dataset to train models, the ensemble model cannot outperform all base classifiers on all experimental datasets. However, the random undersampling technique assists the ensemble classifier to perform better than their base learners on all datasets. It is concluded that the use of random undersampling contributes to the significant improvement of the performance of base classifiers and the final ensemble model. Obtained results have shown the critical role of balanced training datasets on the accuracy of binary classification algorithms.

Table 3. Average F1-score over datasets of the ensemble model and its base classifiers

Type	Classifier	Dataset						
		Ant 1.7	Camel 1.6	Ivy 2.0	Poi 2.0	Tomcat	Xalan 2.4	Synapse 1.2
Original imbalanced dataset	Ensemble	0.5278	0.2321	0.2759	0	0.2222	0.2535	0.5333
	kNN	0.4961	0.2171	0.2667	0.0833	0.1154	0.1918	0.5556
	BN	0.6095	0.2449	0.3256	0	0.3634	0.4348	0.5385
	J48	0.5576	0.2857	0.3125	0	0.3077	0.2632	0.557
	MLP	0.4493	0.3172	0.1935	0.0667	0.274	0.3059	0.5789
	SVM	0.3186	0.0208	0.1905	0.0833	0	0.0351	0.4706
Random under sampling	Ensemble	**0.6261**	**0.4413**	**0.3937**	**0.3354**	**0.3899**	**0.4535**	**0.6487**
	kNN	0.5382	0.4021	0.2695	0.3191	0.3455	0.3865	0.5943
	BN	0.6059	0.2439	0.3709	0.3206	0.3686	0.4067	0.6105
	J48	0.5833	0.3804	0.3534	0	0.3418	0.3984	0.559
	MLP	0.5015	0.3954	0.2934	0.3184	0.3393	0.3807	0.5833
	SVM	0.5884	0.3776	0.3227	0.3078	0.3538	0.3931	0.5924

5. CONCLUSION

This paper showed the efficiency of integrating the random undersampling to the ensemble learning on the imbalanced software defect datasets. Experimental outcomes pointed out that balanced training datasets allow the significant enhancement of performance of both the ensemble model and base classifiers. As a result, the combination of the sampling technique and ensemble learning contributes to forming a promising classifier for the software fault prediction problem. The ensemble model in this paper adopts only a simple majority voting rule. Therefore, we intend to produce a variety of ensemble classifiers using different rules in the future. Moreover, several other sampling methods such as oversampling techniques and evolving sampling strategies will be applied to binary classification models.

ACKNOWLEDGEMENTS

This work was supported by University of Danang, University of Science and Technology, code number of Project: T2018-02-50, and Ministry of Education and Training Vietnam for the research project in the period 2019–2020, code number of Project: B 2019-DNA-03.

REFERENCES

[1] T. T. Khuat and M. H. Le, "A Novel Hybrid ABC-PSO Algorithm for Effort Estimation of Software Projects Using Agile Methodologies," *Journal of Intelligent Systems*, vol. 17, no. 3, pp. 489-506, 2017.
[2] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, vol. 58, pp. 388-402, 2015.
[3] X. Yuan, T. M. Khoshgoftaar, E. B. Allen, and K. Ganesan, "An application of fuzzy clustering to software quality prediction," in *Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, pp. 85-90, 2000.
[4] T. M. H. Le, T. B. Nguyen, and T. T. Khuat, "Survey on Mutation-based Test Data Generation," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 5, no. 5, pp. 1164-1173, 2015.
[5] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering, journal article*, vol. 17, no. 4, pp. 531-577, 2012.
[6] Z. Sun, Q. Song, and X. Zhu, "Using Coding-Based Ensemble Learning to Improve Software Defect Prediction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1806-1817, 2012.
[7] R. Akbani, S. Kwek, and N. Japkowicz, "Applying Support Vector Machines to Imbalanced Datasets," in *Proceedings of the 15th European Conference on Machine Learning*, pp. 39-50, 2004.
[8] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263-1284, 2009.
[9] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429-449, 2002.
[10] N. Bouguila, W. Jian Han, and A. B. Hamza, "A Bayesian approach for software quality prediction," in *Proceedings of the 4th International IEEE Conference Intelligent Systems*, pp. 1149-1154, 2008.
[11] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358-3378, 2007.
[12] L. Rokach, "Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography," *Computational Statistics & Data Analysis*, vol. 53, no. 12, pp. 4046-4072, 2009.
[13] T. Wang, W. Li, H. Shi, and Z. Liu, "Software Defect Prediction Based on Classifiers Ensemble," *Journal of Information and Computational Science*, vol. 8, no. 16, pp. 4241-4254, 2012.
[14] T. Menzies, R. Krishna, and D. Pryor. "The Promise Repository of Empirical Software Engineering Data," [Online]. Available: http://openscience.us/repo.

[15] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2-13, 2007.
[16] T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Predicting the location and number of faults in large software systems," *IEEE Transactions on Software Engineering*, vol. 31, no. 4, pp. 340-355, 2005.
[17] N. E. Fenton and M. Neil, "Software metrics: successes, failures and new directions," *Journal of Systems and Software*, vol. 47, no. 2, pp. 149-157, 1999.
[18] T. J. McCabe, "A Complexity Measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308-320, 1976.
[19] D. N. Card and W. W. Agresti, "Measuring software design complexity," *Journal of Systems and Software*, vol. 8, no. 3, pp. 185-197, 1988.
[20] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994.
[21] A. Marcus, D. Poshyvanyk, and R. Ferenc, "Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented Systems," *IEEE Transactions on Software Engineering*, vol. 34, no. 2, pp. 287-300, 2008.
[22] D. Bowes, T. Hall, and D. Gray, "DConfusion: a technique to allow cross study performance evaluation of fault prediction studies," *Automated Software Engineering, journal article*. vol. 21, no. 2, pp. 287-313, 2014.
[23] G. M. Weiss, "Mining with rarity: a unifying framework," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7-19, 2004.
[24] L. Gonzalez-Abril, H. Nuñez, C. Angulo and F. Velasco, "GSVM: An SVM for handling imbalanced accuracy between classes inbi-classification problems," *Applied Soft Computing*, vol. 17, no. Supplement C, pp. 23-31, 2014.
[25] I. Mani and J. Zhang, "KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction," in *Proceedings of International Conference on Machine Learning*, 2003.
[26] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2012.
[27] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 20-29, 2004.

BIOGRAPHIES OF AUTHORS

Thanh Tung Khuat completed the B.S degree in Software Engineering from University of Science and Technology, Danang, Vietnam, in 2014. Currently, he is working towards the Ph.D. degree at the Advanced Analytics Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. His research interests include machine learning, knowledge discovery, evolutionary computation, intelligent optimization techniques and applications in software engineering.

My Hanh Le is currently a lecturer of the Information Technology Faculty, University of Science and Technology, Danang, Vietnam. She gained M.Sc. degree in 2004 and Ph.D. degree in Computer Science from The University of Danang in 2016. Her research interests are about software testing and more generally application of heuristic techniques to problems in software engineering.