PAPER

The Performance Stability of Defect Prediction Models with Class Imbalance: An Empirical Study

Qiao YU[†], ***Student Member***, **Shujuan JIANG**[†a)], ***and*** **Yanmei ZHANG**[†], ***Nonmembers***

SUMMARY Class imbalance has drawn much attention of researchers in software defect prediction. In practice, the performance of defect prediction models may be affected by the class imbalance problem. In this paper, we present an approach to evaluating the performance stability of defect prediction models on imbalanced datasets. First, random sampling is applied to convert the original imbalanced dataset into a set of new datasets with different levels of imbalance ratio. Second, typical prediction models are selected to make predictions on these new constructed datasets, and Coefficient of Variation ($C{\cdot}V$) is used to evaluate the performance stability of different models. Finally, an empirical study is designed to evaluate the performance stability of six prediction models, which are widely used in software defect prediction. The results show that the performance of C4.5 is unstable on imbalanced datasets, and the performance of Naive Bayes and Random Forest are more stable than other models.

key words: *class imbalance, software defect prediction, prediction models, performance stability, imbalance ratio*

1. Introduction

Class imbalance has gradually become a crucial problem in the fields of machine learning and data mining in recent years. Generally, class imbalance refers to the situation where the number of samples from one class is much higher than that from other class [1]. Class imbalance is a common problem in various areas, such as text classification [2], medical diagnosis [3] and software defect prediction [4], [5].

Software defect prediction can be regarded as a binary classification problem, which aims to divide software modules into defective modules or non-defective modules. Defective modules are usually the minority class, and non-defective modules are the majority class. In this case, correctly classifying the minority class is more important than that of the majority class. However, many prediction models usually focus on maximizing the overall accuracy, which may reduce the accuracy of minority class [6]. For example, suppose that a dataset contains 1000 samples, including only 10 defective samples and 990 non-defective samples. If one prediction model classifies all samples as non-defective samples, its overall accuracy could up to 99.0%, but the accuracy of minority class is 0%. We find that this prediction model may be ineffective on imbalanced datasets [7].

At present, many solutions have been proposed to solve the class imbalance problem, such as sampling, cost-sensitive learning and ensemble learning [8]. However, most of researchers often take no account of the impact of class imbalance, and they do not take any measures to deal with this problem in empirical studies. Therefore, in order to achieve better performance, we should choose the more stable and efficient prediction models for experiments. We think about which prediction models are more stable in terms of their performance with class imbalance? Based on this question, we conduct an empirical study to evaluate the performance stability of different prediction models in software defect prediction.

In this paper, we present an approach to evaluating the performance stability of defect prediction models with class imbalance. First of all, we apply random sampling to convert the original imbalanced dataset into a set of new datasets with different levels of imbalance ratio. Then, we select typical prediction models to make predictions on these new constructed datasets, and we use Coefficient of Variation ($C{\cdot}V$) [9] to evaluate the performance stability of different models. Finally, we design experiments to evaluate the performance stability of six prediction models, which are widely used in software defect prediction. The results show that the performance of C4.5 [10] obviously decreases on most datasets with the increasing of imbalance ratio, which indicates that its performance is unstable on imbalanced datasets. In contrast, the performance of Naive Bayes [11] and Random Forest [12] are more stable than other models. Based on our work, we can be aware of the performance stability of different prediction models with class imbalance, and we can choose reasonable prediction models in empirical studies and practical applications.

The main contributions of this paper are concluded as follows:

- An approach is proposed to evaluate the performance stability of defect prediction models with class imbalance.
- An empirical study is designed to evaluate the performance stability of six prediction models. The results show that the performance of C4.5 is unstable on imbalanced datasets, and the performance of Naive Bayes and Random Forest are more stable than other models.

The remainder of this paper is organized as follows. Section 2 summarizes the related work of class imbalance. Section 3 describes our approach in detail, and Sect. 4 gives an empirical study to show its validity. In Sect. 5, we draw

Manuscript received May 17, 2016.
Manuscript revised September 28, 2016.
Manuscript publicized November 4, 2016.

[†]The authors are with the School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China.
a) E-mail: shjjiang@cumt.edu.cn (Corresponding author)

DOI: 10.1587/transinf.2016EDP7204

our conclusions and discuss the following work.

2. Related Work

In recent years, class imbalance has drawn much attention of researchers in software defect prediction. In order to explore the impact of class imbalance on software defect prediction and its related techniques, many researchers have carried out a large number of empirical studies.

For example, Japkowicz and Stephen [13] showed that class imbalance could affect the performance of decision tree model. Wang et al. [1] provided an empirical study on the stability of feature selection techniques. The experimental results showed that many factors could affect the stability of feature selection, such as class balance, feature subset size and perturbation level. Galinac Grbac et al. [14] investigated the stability of different techniques in relation to levels of data imbalance. They also indicated that feature selection was unstable with higher level of data imbalance.

In addition, Wang and Yao [5] investigated the performance of resampling, cost-sensitive and ensemble algorithms with class imbalance. The results showed that AdaBoost.NC, one of ensemble algorithms, achieved the best performance. Seiffert et al. [15] investigated the combined effects of class imbalance and noise on the performance of sampling techniques. Ryu et al. [16] investigated the impact of class imbalance on cross-project defect prediction (CPDP), and they proposed a value-cognitive boosting approach for CPDP. The experimental results showed that this approach outperformed both existing CPDP approaches and class imbalance approaches.

There are many solutions to the class imbalance problem, such as sampling, cost-sensitive learning and ensemble learning. For example, Tahir et al. [17] proposed an inverse random under-sampling to address the class imbalance problem, which could also be applied to improve the accuracy of multi-class problem. Qian et al. [18] presented a resampling ensemble algorithm for imbalanced datasets, which applied over-sampling for small classes and under-sampling for large classes. However, under-sampling may lead to the information loss, and over-sampling may result in overfitting problem.

Cost-sensitive learning aims to set different misclassification costs for different classes. Traditional prediction models consider that the misclassification costs of different classes are the same [19]. In practice, the cost of misclassifying minority class is much higher than that of misclassifying majority class. Ensemble learning is proposed to gather multiple prediction results to improve the performance of one single model. Generally, the ensemble model outperforms one single model. Ensemble learning is not specifically proposed for solving the class imbalance problem, but it has obtained improvements in addressing this problem [20]. Moreover, Rodriguez et al. [21] also indicated that ensemble approaches outperformed the sampling and cost-sensitive approaches when dealing with the class imbalance problem in software defect prediction.

Although there are so many solutions, many researchers often ignore the impact of class imbalance unless specifically research this problem. So we should choose the stable prediction model for better performance in empirical studies. Which prediction models are more stable on imbalanced datasets? Based on this question, we conduct an empirical study to evaluate the performance stability of defect prediction models with class imbalance.

3. Our Approach

Class imbalance is a common problem in software defect prediction, which may affect the performance of defect prediction models. In this paper, we present an approach to evaluating the performance stability of defect prediction models with class imbalance. The details are described as follows.

3.1 The Framework

Suppose that a standard defect dataset is expressed as $\boldsymbol{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, and $\boldsymbol{x}_i \in R^d$. It indicates that there are n samples in dataset $\boldsymbol{D}$ and d features in each sample of this dataset. We count the number of defective samples and non-defective samples in dataset $\boldsymbol{D}$, marked as n_1 and n_2. The imbalance ratio (IR) [22] can be defined as the number of non-defective samples divided by the number of defective samples, that is n_2/n_1. In our approach, we redefine IR as the floor of n_2/n_1, which is shown in formula (1):

$$\mathrm{IR} = \lfloor n_2/n_1 \rfloor \tag{1}$$

Where $n_2 > n_1$, and IR > 1. The higher the value of IR is, the more imbalanced the dataset is.

Based on the definition of IR, we give the framework of our approach in Fig. 1. For an original imbalanced dataset $\boldsymbol{D}$, we suppose that the IR of dataset $\boldsymbol{D}$ is represented by r. Then, we construct new datasets by random sampling, which can convert the original imbalanced dataset $\boldsymbol{D}$ into a set of new datasets $\boldsymbol{D}_i^*$ $(i = 1, 2, \ldots, r)$ with IR increased one by one, as IR $= 1, 2, \ldots, r$. After that, we select prediction models to make predictions on new datasets $\boldsymbol{D}_i^*$ and evaluate the performance stability of these prediction models eventually.

3.2 New Datasets Construction

In order to evaluate the performance stability of prediction models on imbalanced datasets, we need to obtain a set of datasets with different levels of IR. In our approach, we use random sampling to convert the imbalanced datasets into a set of new datasets with increasing IR. The process of new datasets construction is shown in Fig. 2. The meanings of graphs and symbols are explained in the bottom-left corner of Fig. 2.

As shown in Fig. 2, there are n samples in the original imbalanced dataset $\boldsymbol{D}$, and the number of defective samples

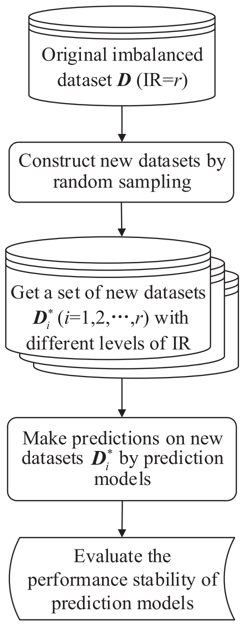
Original imbalanced dataset D (IR=r)
Construct new datasets by random sampling
Get a set of new datasets D_i^* (i=1,2,⋯,r) with different levels of IR
Make predictions on new datasets D_i^* by prediction models
Evaluate the performance stability of prediction models

Fig. 1 The framework of our approach.

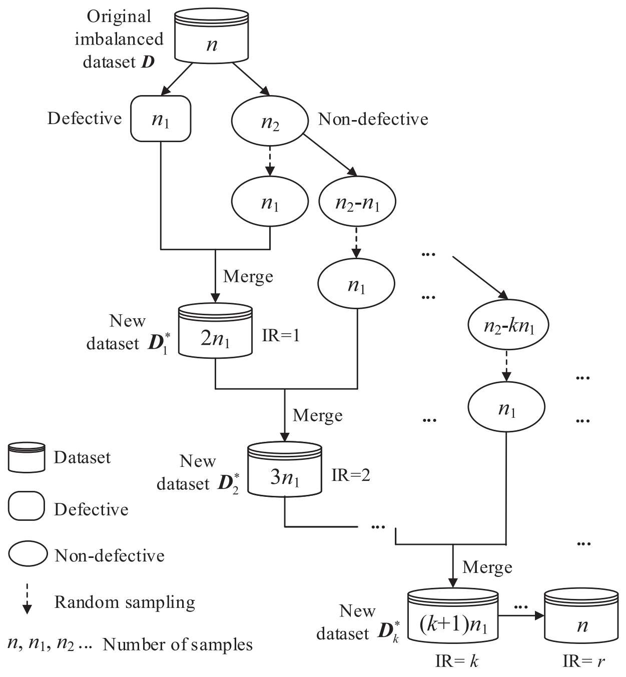

Fig. 2 New datasets construction.

and non-defective samples are marked as n_1 and n_2. First, we select n_1 non-defective samples from all non-defective samples (n_2) randomly, and merge them with all defective samples (n_1). The new dataset $\boldsymbol{D}_1^*$ is produced, and the IR of $\boldsymbol{D}_1^*$ is equal to 1. Next, we also select n_1 non-defective samples from the rest non-defective samples (n_2-n_1) randomly. We would randomize the rest non-defective samples before each sampling to ensure that these samples are evenly distributed. After that, we merge the selected samples with $\boldsymbol{D}_1^*$, so the new dataset $\boldsymbol{D}_2^*$ is produced. The IR of $\boldsymbol{D}_2^*$ is equal to 2. Repeat the above operations until the number of rest non-defective samples is less than n_1. Finally, we can convert the original imbalanced dataset $\boldsymbol{D}$ into a set of new datasets $\boldsymbol{D}_i^*$ with IR increased one by one, as IR $= 1, 2, \ldots, r$.

We find that this process is achieved by under-sampling, and we can get a set of new datasets with IR increased one by one. If we apply over-sampling to construct new datasets, the values of IR may not be increased one by one. For example, there are 10 defective samples and 500 non-defective samples in an imbalanced dataset, and its IR is equal to 50. Then, we apply over-sampling to replicate defective samples, and the IR of new datasets are $\lfloor 500/11 \rfloor = 45$, $\lfloor 500/12 \rfloor = 42$, $\lfloor 500/13 \rfloor = 38$, $\lfloor 500/14 \rfloor = 36$ etc., which are intermittent. However, this may affect the accuracy of the following evaluation.

Therefore, we apply under-sampling to construct new datasets in our approach. Particularly, we assume that the distributions of samples between original dataset and the new datasets are the same. That is to say, we take no account of the information loss in the process of under-sampling.

3.3 Prediction Models Evaluation

We count the widely used prediction models in software defect prediction. Table 1 lists the name, explanation and related references of these models. Particularly, references indicate that they have used these models in their empirical studies. Considering that the performance stability of these models may be related to the validity of empirical studies, it is important to be aware of the performance stability of different prediction models. Based on this, we aim to evaluate the performance stability of these widely used prediction models with class imbalance.

As listed in Table 1, C4.5 [10] is a decision tree algorithm, and it uses information gain ratio for feature selection, which eliminates the bias of selecting frequent features of information gain. K-Nearest Neighbors (KNN) [31] is an instance-based algorithm, and one sample can be classified by a majority vote of its k nearest neighbors. Logistic Regression (LR) [32] is designed to add the logistic regression function into linear regression model. MultiLayer Perceptron (MLP) [33] is a feed-forward artificial neural network model with multiple layers. Naive Bayes (NB) [11] is a probabilistic classifier based on Bayes theorem, and it supposes that all features are independent. Random Forest (RF) [12] is an ensemble model of random trees, and the output of RF is obtained by voting in classification problem. Based on the above prediction models, we aim to evaluate their performance stability on imbalanced datasets.

Next, we should select reasonable metric to evaluate the performance of these prediction models. In this paper, we apply AUC [34] metric, which represents the Area Under the receiver operating characteristic (ROC) Curve. As a binary classification problem, there are four different results between actual results and prediction results as the confusion matrix shown in Table 2.

True positive rate (TPR) represents the ratio of cor-

Table 1 The description of prediction models.

Name	Explanation	References
C4.5	Decision tree	[23]–[27]
K-Nearest Neighbors (KNN)	Instance-based algorithm	[28], [29]
Logistic Regression (LR)	Regression model with logistic function	[24]–[28]
MultiLayer Perceptron (MLP)	Neural network model with multiple layers	[27]–[30]
Naive Bayes (NB)	A probabilistic classifier based on Bayes theorem	[23]–[30]
Random Forest (RF)	An ensemble model of random trees	[23], [26], [27], [30]

Table 2 The confusion matrix.

		Prediction results	
		Defective	Non-defective
Actual results	Defective	True positive (TP)	False negative (FN)
	Non-defective	False positive (FP)	True negative (TN)

rectly predicted as defective samples and actual defective samples, as TP/(TP+FN). False positive rate (FPR) represents the ratio of incorrectly predicted as defective samples and actual non-defective samples, as FP/(FP+TN). ROC curve shows the relationship between TPR and FPR, and AUC is equal to the area under the ROC curve. The range of AUC is between 0 and 1. The larger the value of AUC is, the better the performance of one prediction model is.

Although there are a lot of metrics for evaluating the performance of prediction models, such as Precision, Recall and F-Measure. Jiang et al. [35] proved that AUC was more accurate and reliable than other metrics. Moreover, AUC has been widely used to evaluate the performance of prediction models in software defect prediction [23], [28], [29], [36], [37]. Therefore, we also use AUC to evaluate the performance of prediction models.

4. Empirical Study

To investigate the validity of our approach, we conduct an empirical study on datasets from the PROMISE repository[†]. All experiments are conducted on Open JDK 1.7 and Weka 3.7[††].

4.1 Experimental Subjects

We select eight datasets from NASA and PROMISE as the subjects of our experiments. The reason why we choose these datasets is that they are imbalanced datasets and their imbalance ratios are in different levels. In addition, Shepperd et al. [38] indicated that there were many repeated and inconsistent data in the original NASA datasets, and they provided the cleaned datasets. Therefore, we use these cleaned NASA datasets in our experiments, which can be obtained from the PROMISE repository.

The details of these datasets are described in Table 3. It shows the name of group, dataset and the number of features included in each dataset (columns 1–3). Then it shows the number of all samples, defective samples and non-defective samples (columns 4–6). Finally, it shows the value of IR (column 7), which can be calculated by formula (1). The higher IR indicates that the dataset is more imbalanced. We find that the IR of three datasets MC1, PC2 and Jedit-4.3 are 42, 45 and 43 respectively. It indicates that these datasets are more imbalanced than others.

[†]http://openscience.us/repo/
[††]http://www.cs.waikato.ac.nz/ml/weka/

4.2 Experimental Design

Our experiments are conducted on all prediction models in Table 1 and all datasets in Table 3. All prediction models are implemented in Weka, where these prediction models are designed as classifiers. Considering that most of researchers are used to taking default parameters of prediction models in empirical studies, we also take the default parameters in our experiments.

All experiments are conducted over ten times of 10-fold cross-validation, which is commonly used in empirical studies [28], [39]. In order to reduce the effect of random sampling on new datasets construction, we take the average of 100 times same experiments as the final value of AUC, so as to ensure the prediction results more accurate and reliable.

4.3 Experimental Results and Analysis

First, we select a dataset from Table 3 and suppose that its IR is r. We use random sampling to convert this dataset into a set of new datasets with different levels of IR. Then, we apply the prediction models in Table 1 to make predictions on these new constructed datasets, and we can get a set of AUC, marked as $\{\text{AUC}_i\}$, and $i = 1, 2, \ldots, r$. At last, we calculate the average μ and standard deviation σ of all values of AUC, thereby analyzing the performance differences of each prediction model on imbalanced datasets. The calculations are shown in formulas (2) and (3):

$$\mu = \frac{\sum_{i=1}^{r} \text{AUC}_i}{r} \tag{2}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{r} (\text{AUC}_i - \mu)^2}{r}} \tag{3}$$

Based on the above formulas, we can get the average μ and standard deviation σ of all prediction models, and the results are listed in Table 4. It shows that the average performance of these prediction models are different.

In order to show the average performance of different prediction models clearly, we use the line charts as displayed

Table 3 Defect datasets.

Group	Dataset	Number of features	Number of all samples	Number of defective samples	Number of non-defective samples	IR
NASA	MC1	38	1988	46	1942	42
	PC1	37	705	61	644	10
	PC2	36	745	16	729	45
	PC3	37	1077	134	943	7
PROMISE	Camel-1.0	20	339	13	326	25
	Ivy-1.4	20	241	16	225	14
	Jedit-4.3	20	492	11	481	43
	Tomcat	20	858	77	781	10

Table 4 The average μ and standard deviation σ of prediction models.

Model	MC1		PC1		PC2		PC3	
	μ	σ	μ	σ	μ	σ	μ	σ
C4.5	0.626	0.039	0.722	0.023	0.588	0.066	0.658	0.024
KNN	0.673	0.011	0.703	0.016	0.566	0.043	0.650	0.023
LR	0.733	0.009	0.820	0.021	0.713	0.022	0.816	0.006
MLP	0.698	0.013	0.807	0.014	0.715	0.016	0.784	0.004
NB	0.699	0.006	0.765	0.010	0.740	0.011	0.753	0.005
RF	0.896	0.012	0.869	0.008	0.798	0.008	0.826	0.004

Model	Camel-1.0		Ivy-1.4		Jedit-4.3		Tomcat	
	μ	σ	μ	σ	μ	σ	μ	σ
C4.5	0.498	0.057	0.501	0.035	0.512	0.071	0.652	0.018
KNN	0.608	0.025	0.541	0.019	0.695	0.017	0.656	0.014
LR	0.577	0.018	0.501	0.031	0.611	0.027	0.788	0.012
MLP	0.629	0.013	0.554	0.039	0.542	0.073	0.761	0.005
NB	0.727	0.007	0.672	0.013	0.592	0.011	0.799	0.003
RF	0.709	0.016	0.658	0.011	0.786	0.008	0.822	0.004

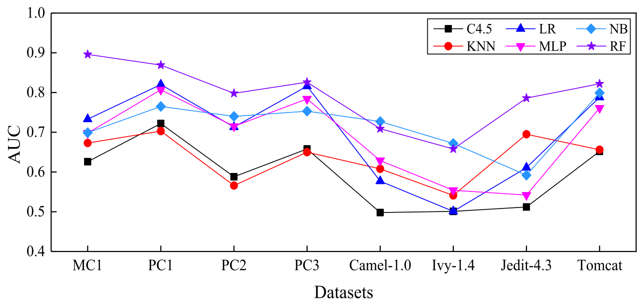

Fig. 3 The average performance of prediction models.

in Fig. 3. We find that the average performance of RF is higher than other models, and C4.5 may be lower than other models on most datasets. We also find that KNN is comparable to C4.5 on NASA datasets, but performs better than C4.5 on PROMISE datasets. In addition, there is no significant difference between LR and MLP.

When it comes to the performance stability, we would first think of the standard deviation. As shown in Fig. 3, when different prediction models are performed on the same dataset, their average performance μ may be different. We could not only apply the standard deviation σ to evaluate their performance stability. In this paper, we apply $C{\cdot}V$ to evaluate the performance stability of different models, which represents the percentage of standard deviation σ and average μ, thereby eliminating the effect of average difference on the comparison of performance stability. The calculation of $C{\cdot}V$ is shown in formula (4).

$$C{\cdot}V = \frac{\sigma}{\mu} \times 100\% \tag{4}$$

The higher the value of $C{\cdot}V$ is, the more unstable the performance of one model is.

Based on the results in Table 4, we calculate the $C{\cdot}V$ of prediction models as listed in Table 5. The last column shows the total $C{\cdot}V$ on all datasets. We can conclude that the total $C{\cdot}V$ of C4.5 (59.3%) is much higher than that of other models, which indicates that its performance is unstable on most datasets. In contrast, the performance of NB (9.6%) and RF (9.2%) are more stable than other models. In addition, the distribution differences of datasets may affect the performance stability of prediction models to a certain extent. For instance, the performance of MLP is extremely unstable on dataset Jedit-4.3 (13.5%) and Ivy-1.4 (7.0%), but it is stable on other datasets.

To further explore the performance variation of different prediction models with the increasing of IR, we use the line charts as displayed in Fig. 4, where the x-axis represents the value of IR, and y-axis represents the value of AUC. When the value of AUC is 0.5, as shown with dash line in Fig. 4, it represents the performance of random guess model. Additionally, Table 3 shows that the values of IR are 42, 10, 45, 7, 25, 14, 43 and 10 respectively, so the lines vary in length as shown in Fig. 4. What's more, only half of the values are marked with symbols.

From all subgraphs in Fig. 4, we can conclude that the performance of C4.5 significantly decreases on most datasets with the increasing of IR, and even worse than that

Table 5 The $C{\cdot}V$ of prediction models.

Model	MC1	PC1	PC2	PC3	Camel-1.0	Ivy-1.4	Jedit-4.3	Tomcat	Total
C4.5	6.2%	3.2%	11.2%	3.6%	11.4%	7.0%	13.9%	2.8%	59.3%
KNN	1.6%	2.3%	7.6%	3.5%	4.1%	3.5%	2.4%	2.1%	27.1%
LR	1.2%	2.6%	3.1%	0.7%	3.1%	6.2%	4.4%	1.5%	22.8%
MLP	1.9%	1.7%	2.2%	0.5%	2.1%	7.0%	13.5%	0.7%	29.6%
NB	0.9%	1.3%	1.5%	0.7%	1.0%	1.9%	1.9%	0.4%	9.6%
RF	1.3%	0.9%	1.0%	0.5%	2.3%	1.7%	1.0%	0.5%	9.2%

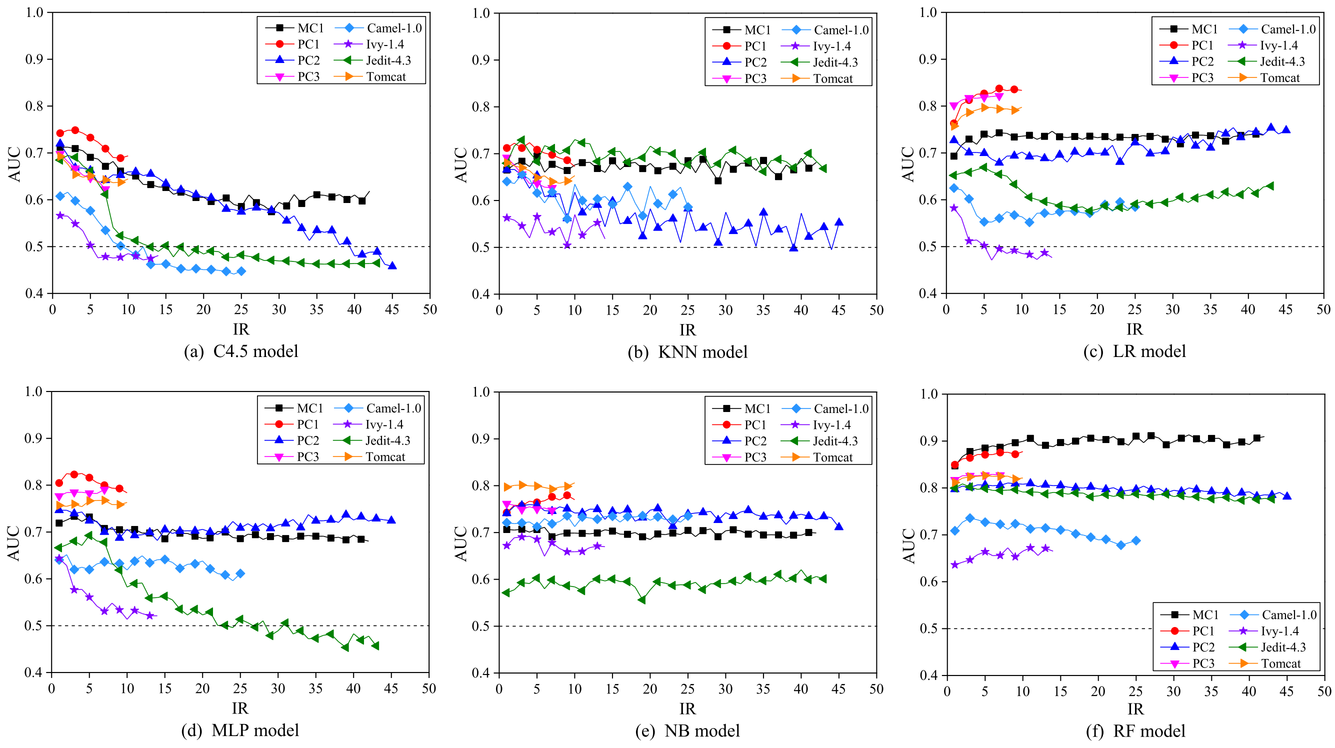

Fig. 4 The performance variation of prediction models.

of random guess model. That is to say, C4.5 is more vulnerable to the class imbalance problem. We can also conclude that the performance of RF is the most stable among all prediction models. The reason is that RF is an ensemble learning model of random trees, so its performance is more stable than the single model. Similarly, the performance of NB is also stable with the increasing of IR, which indicates that NB could be insensitive to class imbalance.

In addition, we find that most prediction models show the worst performance on dataset Ivy-1.4. The reason is that dataset Ivy-1.4 contains only 241 samples, which may reduce the performance of prediction models due to lack of historical data.

4.4 Comparison and Discussion

Based on the above experiments, we can conclude that there are indeed some prediction models whose performance are unstable on imbalanced datasets.

For example, we find that the performance of C4.5 is unstable when the dataset is imbalanced. Similarly, Japkowicz and Stephen [13] proved that class imbalance could also affect the performance of C5.0. It should be noted that C4.5 and C5.0 are different versions of decision tree model. Khoshgoftaar et al. [40] showed that Random Forest performed better on imbalanced datasets. Our experiments also show that the performance of Random Forest is more stable on imbalanced datasets. It can be demonstrated that our approach can effectively evaluate the performance stability of different prediction models with class imbalance. Compared with other methods, the advantage of our approach is that it can show the performance variation of prediction models with the increasing of IR.

Above all, based on our work, we can be aware of the performance stability of different prediction models with class imbalance. Furthermore, we can select reasonable prediction models in empirical studies. For example, Catal and Diri [23] investigated the effect of dataset size, metrics and feature selection. The experiments were developed on several models, such as C4.5, Naive Bayes and Random Forest. These models were also implemented in Weka with default parameters. However, according to our evaluation, C4.5 is unstable when the dataset is imbalanced. In order to reduce the effect of class imbalance on their experiments, C4.5 may not be suitable, but Naive Bayes and Random Forest are good choices.

In addition, He et al. [24] investigated the feasibility of cross-project defect prediction. Similarly, C4.5 is also used in their experiments, which may reduce the efficiency of cross-project results to a certain extent. In this case, we should choose such prediction models with better and more stable performance as Naive Bayes and Random Forest in empirical studies.

4.5 Threats to Validity

Based on our experiments, we conclude several threats to the validity of our empirical study.

- Construct validity

The new datasets construction is the main threat to the construct validity. Our approach is designed to convert the original imbalanced dataset into a set of new datasets by random sampling. In order to reduce the effect of random errors of sampling as much as possible, we take the average of 100 times same experiments as the final results. Moreover, we take no account of the distribution differences between the original dataset and new constructed datasets.

- Internal validity

The parameters of prediction models may be the threat to the internal validity. Considering that many researchers are used to taking default parameters in empirical studies, we also take the default parameters of prediction models in Weka. Tantithamthavorn et al. [37] investigated that different parameters indeed had an impact on the performance of defect prediction models. Whether different parameters could affect the performance stability of prediction models should be discussed in the following work.

- External validity

Dataset quality may be the most important threat to the external validity. In our experiments, we select eight imbalanced datasets from NASA and PROMISE. These datasets are in different size and different levels of IR. Moreover, they are commonly used in software defect prediction.

5. Conclusions

This paper presents an approach to evaluating the performance stability of defect prediction models with class imbalance. We evaluate the performance stability of six prediction models in our experiments. The results show that the performance of C4.5 is unstable on imbalanced datasets. In contrast, the performance of Naive Bayes and Random Forest are more stable than other models. Based on the results, we can choose reasonable prediction models in empirical studies.

We only evaluate the performance stability of six prediction models, and more prediction models should be evaluated. In addition, we take default parameters of prediction models in our experiments. The performance stability with different parameters should be discussed in the following work.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China (No. 61673384 and No. 61502497), the Guangxi Key Laboratory of Trusted Software (No. kx201530), the China Postdoctoral Science Foundation Project (No. 2015M581887) and the Scientific Research Innovation Project for Graduate Students of Jiangsu Province (No. KYLX15_1443).

References

[1] H. Wang, T.M. Khoshgoftaar, and A. Napolitano, "An empirical study on the stability of feature selection for imbalanced software engineering data," Proc. 11th Int. Conf. Mach. Learn. & Appl., vol.1, pp.317–323, Boca Raton, USA, Dec. 2012.
[2] A. Sun, E.P. Lim, and Y. Liu, "On strategies for imbalanced text classification using SVM: A comparative study," Decis. Support Syst., vol.48, no.1, pp.191–201, 2009.
[3] M.A. Mazurowski, P.A. Habas, J.M. Zurada, J.Y. Lo, J.A. Baker, and G.D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," Neural Networks, vol.21, no.2-3, pp.427–436, 2008.
[4] Y. Ma, G. Luo, and H. Chen, "Kernel based asymmetric learning for software defect prediction," IEICE Trans. Inf. & Syst., vol.E95-D, no.1, pp.267–270, Jan. 2012.
[5] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," IEEE Trans. Reliab., vol.62, no.2, pp.434–443, 2013.
[6] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," Pattern Recogn., vol.48, no.5, pp.1623–1637, 2015.
[7] L. Peng, H. Zhang, B. Yang, and Y. Chen, "A new approach for imbalanced data classification based on data gravitation," Inform. Sciences, vol.288, pp.347–373, 2014.
[8] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," Inform. Sciences, vol.250, pp.113–141, 2013.
[9] J. Forkman, "Estimator and tests for common coefficients of variation in normal distributions," Commun. Statist. – Theory & Methods, vol.38, no.2, pp.233–251, 2009.
[10] J.R. Quinlan, C4.5: Programs for machine learning, Morgan Kaufmann Publishers, San Francisco, USA, 1993.
[11] G.H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," Proc. 11th Conf. Uncertainty Artif. Intell., pp.338–345, Montreal, Canada, Aug. 1995.
[12] L. Breiman, "Random forests," Mach. Learn., vol.45, no.1, pp.5–32, 2001.
[13] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," Intell. Data Anal., vol.6, no.5, pp.429–449, 2002.
[14] T. Galinac Grbac, G. Mauša and B. Dalbelo-Bašić, "Stability of software defect prediction in relation to levels of data imbalance," Proc. 2nd Workshop Softw. Qual. Anal. Monit. Improv. & Appl., pp.1–10, Novi Sad, Serbia, Sept. 2013.
[15] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, and A. Folleco, "An empirical study of the classification performance of learners on imbalanced and noisy software quality data," Inform. Sciences, vol.259, pp.571–595, 2014.
[16] D. Ryu, O. Choi, and J. Baik, "Value-cognitive boosting with a

support vector machine for cross-project defect prediction," Empir. Softw. Eng., vol.21, no.1, pp.43–71, 2016.
[17] M.A. Tahir, J. Kittler, and F. Yan, "Inverse random under sampling for class imbalance problem and its application to multi-label classification," Pattern Recogn., vol.45, no.10, pp.3738–3750, 2012.
[18] Y. Qian, Y. Liang, M. Li, G. Feng, and X. Shi, "A resampling ensemble algorithm for classification of imbalance problems," Neurocomputing, vol.143, pp.57–67, 2014.
[19] Ö.F. Arar and K. Ayan, "Software defect prediction using cost-sensitive neural network," Appl. Soft Comput., vol.33, pp.263–277, 2015.
[20] I.H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," Inform. Softw. Technol., vol.58, pp.388–402, 2015.
[21] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J.C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," Proc. 18th Int. Conf. Eval. Assess. Softw. Eng., pp.43:1–43:10, London, United Kingdom, May 2014.
[22] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," IEEE Trans. Syst. Man Cybern. Part C, vol.42, no.4, pp.463–484, 2012.
[23] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," Inform. Sciences, vol.179, no.8, pp.1040–1058, 2009.
[24] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," Automat. Softw. Eng., vol.19, no.2, pp.167–199, 2012.
[25] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," Inform. Softw. Technol., vol.59, pp.170–190, 2015.
[26] J. Nam and S. Kim, "CLAMI: Defect prediction on unlabeled datasets," Proc. 30th Int. Conf. Automat. Softw. Eng., pp.452–463, Lincoln, USA, Nov. 2015.
[27] L. Li and H. Leung, "Mining static code metrics for a robust prediction of software defect-proneness," Proc. 5th Int. Symp. Empir. Softw. Eng. & Meas., pp.207–214, Banff, Canada, Sept. 2011.
[28] K. Gao, T.M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: An investigation on feature selection techniques," Softw. Pract. Exper., vol.41, no.5, pp.579–606, 2011.
[29] T.M. Khoshgoftaar, K. Gao, A. Napolitano, and R. Wald, "A comparative study of iterative and non-iterative feature selection techniques for software defect prediction," Inform. Syst. Front., vol.16, no.5, pp.801–822, 2014.
[30] Y. Zhou, B. Xu, H. Leung, and L. Chen, "An in-depth study of the potentially confounding effect of class size in fault prediction," ACM Trans. Softw. Eng. Method., vol.23, no.1, pp.10:1–10:51, 2014.
[31] D.W. Aha, D. Kibler, and M.K. Albert, "Instance-based learning algorithms," Mach. Learn., vol.6, no.1, pp.37–66, 1991.
[32] S. Le Cessie and J.C. Van Houwelingen, "Ridge estimators in logistic regression," Appl. Statist., vol.41, no.1, pp.191–201, 1992.
[33] J.-G. Attali and G. Pagès, "Approximations of functions by a multilayer perceptron: A new approach," Neural Networks, vol.10, no.6, pp.1069–1081, 1997.
[34] J. Huang and C.X. Ling, "Using AUC and accuracy in evaluating learning algorithms," IEEE Trans. Knowl. Data Eng., vol.17, no.3, pp.299–310, 2005.
[35] Y. Jiang, J. Lin, B. Cukic, and T. Menzies, "Variance analysis in software fault prediction models," Proc. 20th Int. Symp. Softw. Reliab. Eng., pp.99–108, Mysuru, India, Nov. 2009.
[36] E. Erturk and E.A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert Syst. Appl., vol.42, no.4, pp.1872–1879, 2015.
[37] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," Proc. 38th Int. Conf. Softw. Eng., pp.321–332, Austin, Texas, May 2016.
[38] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," IEEE Trans. Softw. Eng., vol.39, no.9, pp.1208–1215, 2013.
[39] Y. Zhou, Y. Yang, B. Xu, H. Leung, and X. Zhou, "Source code size estimation approaches for object-oriented systems from UML class diagrams: A comparative study," Inform. Softw. Technol., vol.56, no.2, pp.220–237, 2014.
[40] T.M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," Proc. 19th Int. Conf. Tools with Artif. Intell., vol.2, pp.310–317, Patras, Greece, Oct. 2007.

Qiao Yu is a Ph.D. candidate at School of Computer Science and Technology, China University of Mining and Technology. Her research interests include software analysis and testing, machine learning. She is a student member of IEICE.

Shujuan Jiang received the Ph.D. degree from Southeast University in 2006. She was a visiting scholar at Georgia Institute of Technology from September 2008 to April 2009. She is a professor and Ph.D. supervisor at School of Computer Science and Technology, China University of Mining and Technology. Her research interests include compilation techniques and software engineering, etc.

Yanmei Zhang received the Ph.D. degree from China University of Mining and Technology in 2012. She is a lecturer at School of Computer Science and Technology, China University of Mining and Technology. Her research interests include software analysis and testing.