Impact of imbalanced data on the performance of software defect prediction classifiers

Lichao Wang\textsuperscript{1,2}, Wei Wang\textsuperscript{1}, Bingyou Liu\textsuperscript{1,2} and Shuqiao Geng\textsuperscript{1,2,3*}

\textsuperscript{1} School of Electrical Engineering, Anhui Polytechnic University, Wuhu, Anhui, 241000, China

\textsuperscript{2} Key Laboratory of Advanced Perception and Intelligent Control of High-end Equipment, Ministry of Education, Wuhu, Anhui, 241000, China

\textsuperscript{3} School of Engineering Science, University of Science and Technology of China, Hefei, Anhui, 230026, China

*Corresponding author’s e-mail: gengshuqiao@ahpu.edu.cn

Abstract: Software defect prediction plays an important role in analysing software quality and balancing software cost. However, it lacks suggestions for project managers and software engineers in selecting classifiers. Firstly, a method for building imbalanced distribution data is proposed. Then, Matthews correlation coefficient is used to measure the performance of different classifiers, and the coefficient of variation is utilised to evaluate the stability of classifiers on imbalanced distribution data. Finally, an experiment is conducted on 8 common classifiers and 12 publicly available and widely used data sets. Results show that NaiveBayes behaves steadily when the imbalance rate of data sets changes significantly. The experimental results provide a basis for project managers and software engineers to select classifiers.

1. Introduction
Software has become an important factor that affects the national economy, military, political and other fields\cite{1}. High-end complex systems heavily depend on the reliability of software being used. The internal defects of software cause system errors, failures and even crashes. Defects significantly affect software quality\cite{2}. Software defect detection is an important work for ensuring software quality.

Software defect prediction is a typical binary classification problem\cite{3}. It aims to divide software modules into defective and non-defective ones. Classifiers are trained with historical data sets to predict new data sets. Usually, the number of defective modules is smaller than that of non-defective modules in one data set, that is, the distribution of data is evidently imbalanced\cite{4}. However, traditional classifiers often aim to maximise the overall classification precision whilst ignoring the classification of the minority class, thereby resulting in no practical value of classification results\cite{5}.

Therefore, a method for building data sets with different imbalance rates is proposed. The new data sets established by this method can be used to evaluate the performance change of classifiers on imbalanced distribution data. This study focused on the research question (RQ) which classifiers are stable with imbalanced distribution data?
This paper is organised as follows: Section 2 explains the proposed classifier performance evaluation method for imbalanced distribution data. Section 3 presents the findings of RQ. Section 4 discusses research conclusions and future work.

2. Classifier performance evaluation method for imbalanced distribution data

2.1. Imbalance Ratio
The distribution of software defect prediction data is imbalanced, that is, the number of non-defective instances in the data set is higher than that of defective instances[6]. Suppose a data set $D$ contains $n$ instances and 1 class feature that labels instance as defective or non-defective. Imbalance rate is defined as follows:

$$ R = \frac{n_n - n_d}{n_n + n_d} \times 100\% , \quad (1) $$

where
- $R$ is the imbalance rate.
- $n_n$ is the number of non-defective modules.
- $n_d$ is the number of defective modules.

Evidently, $R$ ranges from -1 to 1. If $R$ is positive, then the number of non-defective modules is larger than that of defective modules. If $R$ is negative, then the opposite is true. The larger the absolute value of $R$ is, the more evident the imbalance of the distribution of the data set is.

2.2. New data sets built with different imbalance rates
Several data sets with different imbalance rates are constructed to investigate the sensitivity of classifiers to imbalanced distribution data. The process of building new data sets is presented as follows:

**Step 1.** According to the label, the original data are decomposed into two parts: defective data set $D_d$ and non-defective data set $D_n$. The number of instances is $n_d$ and $n_n$. $R$, which is the imbalance rate of the original data, is calculated.

**Step 2.** A new empty data set $D_j$ is created, where $j$ is the current number of repetitions. Defective instances are all placed into the new data set.

**Step 3.** The instances in $D_n$ are reshuffled. In $D_n$, $i$ instances are randomly extracted and placed into new data set,

$$ i = n_d + (j - 1) \frac{n_n - n_d}{kR + 1} , \quad (2) $$

where
- $i$ is the number of instances extracted of which the value is rounded up.
- $n_d$ is the number of defective modules.
- $n_n$ is the number of non-defective modules.
- $j$ is the current number of repetitions.
- $k$ is the maximum number of the building new data sets the value of which is set artificially and can usually be set to 3.
- $R$ is the imbalance rate of the original data.

**Step 4.** The instances in $D_j$ are reshuffled.

**Step 5.** Repeat Steps 2 to 4 until $i \geq n_n$.

Currently, a series of new data sets $D_1$, $D_2$, ..., $D_j$ with declining imbalance rates have been constructed.

2.3. Classifier performance evaluation
After the classifier test is completed, the test results are analysed, and the classifier performance is evaluated. Matthews correlation coefficient (MCC) is used as the performance criteria.
MCC is a metric used to measure the classifier performance of binary classification. It is essentially a correlation coefficient that describes the relationship between actual and predictive classification. The range of values of MCC is [-1,1]. When the value is 1, the perfect prediction of the data set is achieved. When the value is 0, the prediction results are not as good as those of random prediction. When the value is -1, the predictive and actual classifications are totally inconsistent. This metric is suitable for the performance evaluation of classifiers to predict imbalanced distribution data.

3. Experiment

3.1. Data sets and classifiers
A total of 12 data sets from the NASA MDP repository are used in this study[7]. These data sets have been widely used in software defect prediction research. Each data set includes various static code metrics and label of whether defects are found or not. The defective rates of these data sets are less than 33%, and the lowest is only 0.41%. The distribution of software defect data set has evident imbalanced characteristics. Thus, researchers need to fully consider this factor when building models.

Researchers need to select appropriate classifiers in investigating software defect prediction. Classifiers continuously adjusts internal parameters by learning labelled data. Then, they can classify new instance to a given label, thereby realising the prediction of new instance. Common classifiers can be divided into Bayes, decision trees, k-nearest neighbours, support vector machine (SVM) and linear regression.

| Source data set | New Building data set | Imbalance rate | Naïve Bayes | J48 Random Forest | Simple Cart | IBk Simple Logistic | Multilayer Perceptron | SMO |
|-----------------|-----------------------|----------------|-------------|------------------|-------------|-------------------|----------------------|-----|
| CM1 D1          | 0                     | 0.539         | 0.611       | 0.646             | 0.634       | 0.522              | 0.611               | 0.584 |
| D2              | 55.56                 | 0.376         | 0.386       | 0.421             | 0.324       | 0.367              | 0.460               | 0.518 |
| D3              | 71.34                 | 0.282         | 0.167       | 0.149             | 0           | 0.188              | 0.059               | 0.260 |
| D4              | 0                     | 0.269         | 0.382       | 0.380             | 0.379       | 0.349              | 0.367               | 0.355 |
| D5              | 35.86                 | 0.270         | 0.350       | 0.389             | 0.327       | 0.317              | 0.314               | 0.297 |
| D6              | 52.79                 | 0.271         | 0.266       | 0.315             | 0.266       | 0.269              | 0.225               | 0.260 |
| D7              | 0                     | 0.450         | 0.530       | 0.551             | 0.532       | 0.502              | 0.538               | 0.532 |
| D8              | 42.17                 | 0.371         | 0.400       | 0.466             | 0.428       | 0.436              | 0.391               | 0.391 |
| D9              | 59.32                 | 0.310         | 0.279       | 0.388             | 0.297       | 0.404              | 0.216               | 0.267 |
| D10             | 0                     | 0.376         | 0.560       | 0.421             | 0.581       | 0.420              | 0.377               | 0.396 |
| D11             | 55.90                 | 0.332         | 0.358       | 0.505             | 0.187       | 0.425              | 0.351               | 0.411 |
| D12             | 71.62                 | 0.322         | 0.348       | 0.308             | 0           | 0.389              | 0.334               | 0.371 |
| D13             | 0                     | 0.711         | 0.795       | 0.779             | 0.765       | 0.825              | 0.854               | 0.841 |
| D14             | 94.55                 | 0.400         | 0.432       | 0.464             | 0.480       | 0.453              | 0.440               | 0.499 |
| D15             | 98.58                 | 0.310         | 0.470       | 0.483             | 0.446       | 0.383              | 0                   | 0.426 |
| D16             | 0                     | 0.395         | 0.389       | 0.686             | 0.443       | 0.386              | 0.522               | 0.423 |
| D17             | 21.21                 | 0.340         | 0.188       | 0.153             | 0.206       | 0.124              | 0.273               | 0.222 |
| D18             | 0                     | 0.662         | 0.648       | 0.645             | 0.683       | 0.550              | 0.654               | 0.775 |
| D19             | 61.01                 | 0.386         | 0.340       | 0.324             | 0.355       | 0.398              | 0.492               | 0.371 |
| D20             | 75.69                 | 0.387         | 0.334       | 0.351             | 0.264       | 0.351              | 0.437               | 0.319 |
| D21             | 0                     | 0.611         | 0.776       | 0.737             | 0.671       | 0.613              | 0.593               | 0.620 |
| D22             | 63.72                 | 0.379         | 0.444       | 0.540             | 0.465       | 0.471              | 0.382               | 0.250 |
| D23             | 77.81                 | 0.263         | 0.402       | 0.557             | 0.298       | 0.502              | 0.305               | 0.347 |
| D24             | 0                     | 0.766         | 0.696       | 0.783             | 0.739       | 0.427              | 0.624               | 0.614 |
| D25             | 96.81                 | 0.352         | 0.214       | 0.341             | 0.359       | 0.309              | 0.275               | 0.341 |
| D26             | 0                     | 0.513         | 0.606       | 0.632             | 0.569       | 0.367              | 0.651               | 0.656 |
| D27             | 53.49                 | 0.434         | 0.468       | 0.591             | 0.502       | 0.340              | 0.438               | 0.468 |
| D28             | 69.67                 | 0.339         | 0.130       | 0.274             | 0.104       | 0.266              | 0.153               | 0.145 |
| D29             | 0                     | 0.525         | 0.860       | 0.869             | 0.850       | 0.754              | 0.832               | 0.826 |
| D30             | 48.70                 | 0.457         | 0.681       | 0.753             | 0.657       | 0.574              | 0.474               | 0.598 |
| D31             | 65.47                 | 0.431         | 0.623       | 0.670             | 0.599       | 0.533              | 0.448               | 0.493 |
| D32             | 0                     | 0.488         | 0.822       | 0.868             | 0.805       | 0.764              | 0.683               | 0.684 |
| D33             | 80.39                 | 0.591         | 0.753       | 0.778             | 0.740       | 0.740              | 0.609               | 0.632 |
| D34             | 89.13                 | 0.469         | 0.587       | 0.643             | 0.598       | 0.609              | 0.429               | 0.542 |
| D35             | 0.327                 | 0.317         | 0.314       | 0.297             | 0.260       | 0.260              | 0.095               | 0.306 |

These data sets have been widely used in software defect prediction research. Each data set includes various static code metrics and label of whether defects are found or not. The defective rates of these data sets are less than 33%, and the lowest is only 0.41%. The distribution of software defect data set has evident imbalanced characteristics. Thus, researchers need to fully consider this factor when building models.

Researchers need to select appropriate classifiers in investigating software defect prediction. Classifiers continuously adjusts internal parameters by learning labelled data. Then, they can classify new instance to a given label, thereby realising the prediction of new instance. Common classifiers can be divided into Bayes, decision trees, k-nearest neighbours, support vector machine (SVM) and linear regression.
3.2. Experimental results
A total of 34 new data sets with different imbalance rates were built using the method proposed in Section 2.2. They were divided into 12 groups according to the source data. The distribution and imbalance rate of each data set are detailed in Table 1. On the same data set, the average performance of different classifiers vary. Therefore, the coefficient of variation (CV) was used to evaluate the discreteness of MCC values. CV can eliminate the influence of mean difference on the comparison of data discreteness. The mean value and standard deviation of a group of MCC values are expressed by μ and σ respectively; and CV is the ratio of standard deviation to mean value.

Table 2 shows the mean value, standard deviation, CV and average value of 8 classifiers on the 12 groups of new data sets. SMO, SimpleCart and SimpleLogistic were unstable when the imbalance rate of data sets changed, and their average CV values were more than 50%. NaiveBayes was the most stable classifier with an average CV value of only 25.9%.

4. Conclusion and future work
In this study, a method of building imbalanced distribution data is proposed to investigate the impact of imbalance rate on the performance of common classifiers. The experiment involves 8 commonly used classifiers and 12 publicly available and widely used NASA data sets. The following conclusions are drawn according to our RQ:

When the imbalance rate of data sets significantly changes, NaiveBayes behaves steadily. As the imbalance rates of data sets increase, the performance of SMO, SimpleCart and SimpleLogistic shows a significant downward trend. In other words, when selecting the classifier for software defect prediction, software engineers should opt for NaiveBayes.

Future works may address the following:

- In the experiment, the parameters of all classifiers are default values. In the future, we will adjust the parameters to explore the effects of different parameters on the performance of classifiers.
- In this study, only binary classification, defective or non-defective, are considered. In future research, we will consider the multiclassification problem, that is, how many defects exist.
Acknowledgement
This work was partially funded by Anhui Provincial Natural Science Foundation (grant no.1808085QE169), Anhui Provincial Natural Science Foundation (grant no.1808085MF182) and Key research and development project of Anhui Province (grant no.1804b0602036).

References
[1] Sun, Y., Zhang, H. (2011) A software safety analysis method based on S-invariant of Petri Net. In: IEEE International Conference on Reliability. Guiyang. pp. 487-492.
[2] Rempel, P., Mader, P. (2016) Preventing defects: the impact of requirements traceability completeness on software quality. IEEE Transactions on Software Engineering, 43: 777-797.
[3] He, P., Li, B., Liu, X., Chen, J., Ma, Y. (2015) An empirical study on software defect prediction with a simplified metric set. Information and Software Technology, 59: 170-190.
[4] Wang, H., Khoshgoftaar, T.M., Napolitano, A. (2015) Stability of filter-and wrapper-based software metric selection techniques. In: IEEE International Conference on Information Reuse & Integration. Redwood City, CA, USA. pp. 309-314.
[5] Wang, H., Gao, Y., Shi, Y., Wang, H. (2016) A Fast Distributed Classification Algorithm for Large-Scale Imbalanced Data. In: IEEE International Conference on Data Mining. Barcelona, Spain. pp. 1251-1256.
[6] Yu, Q., Jiang, S., Zhang, Y., Wang, X., Gao, P., Qian, J. (2018) The Impact Study of Class Imbalance on the Performance of Software Defect Prediction Models. Chinese Journal of computers, 41: 809-824.
[7] Gray, D., Bowes, D., Davey, N., Sun, Y., Christianson, B. (2012) Reflections on the nasa mdp data sets. IET Software, 6: 549-558.