Predicting the Severity of Closed Source Bug Reports Using Ensemble Methods

M. N. Pushpalatha and M. Mrunalini

Abstract Severity tells about how urgent given bug is to be fixed. There are large numbers of bug identified during software development and maintenance for each bug the bug report will be submitted. Bug report gives very important information such as Description, Severity, Priority, Date and Time of bug report, etc. Even though there are clear guidelines present about how to assign the severity, the inexperienced and busy test engineer may make the mistake in correctly identifying the severity in closed source software development. Automatic prediction of severity helps inexperienced and busy engineer in saving time and resources. In this paper, bug report dataset (PITS) is taken for NASA projects from PROMISE Repository. Predicting the severity is done using Bagging, Voting, Adaboost and Random forest ensemble methods. The result shows bagging gives better accuracy than other ensemble algorithms. Two preprocessing techniques, i.e., Information gain and Chi-square are considered for data reduction. Information gain gives slightly good accuracies over chi-square.

1 Introduction

During open source software development lots of bug reports are submitted by end-users, developers, testers from all over the world are stored in different open source bug repositories like Bugzilla, JIRA, etc. lot of research is done on these repositories to analyze the bug reports for improving the quality of software and delivering the software according to customer requirements. Some of the research is related to predicting the severity of bug reports and predicting the proper developer for fixing that bug. Closed source software developments use slightly different approach for software development than open source. In open source anyone (users, developers,

M. N. Pushpalatha (✉) · M. Mrunalini
Ramaiah Institute of Technology, Bangalore 560054, India
e-mail: pushpalathamn@msrit.edu

M. Mrunalini
e-mail: mrunalini@msrit.edu

S. C. Satapathy et al. (eds.), *Smart Intelligent Computing and Applications*,
Smart Innovation, Systems and Technologies 105,
https://doi.org/10.1007/978-981-13-1927-3_62

and testers) can test the software and if they find any defect then can submit the bug report to the developer. Even though there are clear guidelines on how to assign the severity. Many a times the users make the mistake when assigning the severity to the bug report. In closed source, the assignment of severity is done by test engineer who tests the software. If the test engineer is busy or inexperienced then they are the chances of making the mistakes. Prediction of severity of bug report for closed source will help for the inexperienced and busy test engineer. It is very important to identify it correctly for resource allocation and fixing urgent and critical bug. Assessment of severity in closed source depends on the experience of test engineer and the time he spends on the defect report [1].

In [1–6] used the general classifier such as rule based classification, Naïve Bayes, Naïve Bayes Multinomial, K-Nearest Neighbor, J48, RIPPER, probability based Naïve Bayes, Random Forests and Support vector machine, etc., for predicting the severity. Bagging ensemble method used in [7] for predicting the severity for open source data sets and compared with C4.5. Bagging gave good accuracy over C4.5 general classifier. Literature shows that ensemble methods are not addressed in available work for open source NASA datasets.

In this paper, the defect reports of the NASA's Project Issue and Tracking system (PITS) from PROMISE repository are considered for experiment. PITS is database contains all findings which are captured during NASA's Independent Verification and Validation (IV & V) and contains data for more than 10 years [1]. It contains the data about nuclear reactor, robotics and human-rated missions.

In this paper, different ensemble methods used for predicting the severity for PITS defect reports.

The paper organization is Sect. 2 explains the literature survey, Sect. 3 is about methodology used, in Sect. 4 Result and Discussion is presented and Sect. 5 is about the conclusion and future work.

2 Literature Survey

In the literature machine learning and Text mining techniques are used to address the different problems on bug tracking repositories. Some of the problems addressed by different researchers are on predicting the duplicate bug reports, predicting the severity and priority of bug reports and predicting the developer to resolve the bug, etc. In author [8] used natural language processing to detect duplicate defect reports. In the presence of ancillary data about a bug (e.g., number of affected users), the process of bug triaging could be automated. In this vein, Naive Bayes based classification algorithm has been used to automatically predict the severity of reported bugs [2] of Bugzilla repositories for Eclipse, Mozilla and GNOME component.

Since bug reports typically come with textual descriptions, text mining techniques have been applied on the descriptions of bug reports to automatically triage bugs [9, 10].

The predicting the severity levels for closed source of NASA defect reports is done using RIPPER algorithm [3]. Different measures like recall, precision, and F-measure is used for evaluating the result.

Prediction of severity of open source bug reports from Bugzilla is done by using Naïve Bayes Multinomial, K-Nearest Neighbor, Naïve Bayes, and Support vector machine [3]. Among four algorithms [3] found Naïve Bayes Multinomial gives good accuracy and works with less training sets and fastest. Nearest Neighbor algorithm is used in [4] for predicting the severity of open source software bug reports of Eclipse, OpenOffice and Mozilla from Bugzilla repository. In [5] author used Naïve Bayes Multinomial, Support Vector Machine, Naïve Bayes, k-Nearest Neighbor, J48 and RIPPER algorithms are used for predicting the severity of NASA defect reports, accuracy and F-measure is used for evaluating the result. In author [6] taken NASA's defect reports from PROMISE repository as Closed source data set and bug reports of Eclipse, Mozilla & GNOME from Bugzilla bug repository as open source data sets and used different classification algorithm such as Random Forests, RIPPER, Naïve Bayes, Support Vector Machine and J48 for predicting the severity of both open source and closed source datasets.

Cross projects severity prediction of bug report is done using K-NN, Naïve Bayes and Support vector machine. K-NN gave good performance over other two [11]. For dealing with imbalance bug data problem used the vote and bagging ensemble methods from RapidMiner. F-measure Performance was increased by 5% and 10% using vote and bagging respectively [11]. In this paper used the voting, bagging, Adaboost and random forest ensemble methods from RapidMiner for predicting the severity of closed source data sets.

In [12] Bayesian Networks, Naïve Bayes, REPTree, SVM, Decision tree, rules and Random Forest machine learning algorithm along with Stacking ensemble method for predicting the developer for industrial data and comparison is done on different classification algorithm and concluded that stacking ensemble method increased the accuracy. In [13] authors used bagging ensemble method and Naïve Bayes general classifier for predicting the developer of open source projects and concluded that accuracy can be increased with bagging ensemble method. In this paper, ensemble methods are used for closed source software bug reports.

3 Methodology

NASA's PITs Datasets are taken from the Promise repository [14] and used RapidMiner tool for prediction of severity. NASA's Independent Verification and Validation facility given the five anonymous PITs projects named it as pitsA to pitsF and all five projects are related to robotic.

Table 1 shows the number of bug reports available for each severity and Table 2 tells about the total number of bug reports available for each datasets, their size and total word count is given

Table 1 Number of bug reports for each severity

Projects	Severity 1	Severity 2	Severity 3	Severity 4	Severity 5
PitsA	0	325	375	239	26
PitsB	0	23	523	382	59
PitsC	0	0	132	180	7
PitsD	0	1	167	13	1
PitsE	0	24	517	243	41
PitsF	0	9	477	209	48

Table 2 Total number of bug reports, size and word count

	Total number of reports	Size of the dataset	Total word count
PitsA	965	1.2 MB	173,963
PitsB	987	704.1 KB	104,052
PitsC	319	143.6 KB	23,799
PitsD	182	106.5 KB	15,868
PitsE	825	650.0 KB	93,750
PitsF	743	548.9 KB	82,775

PITS dataset is preprocessed before applying the classification algorithm. The dataset is first tokenized to splits the text of a document into a sequence of tokens. After stop words removal is used to remove the stop words like a, the, etc., next porter stemming is used to stem the words for example the present, presented and presenting is stemmed to present. The dimensionality is reduced to 150 by using Chi-Squared Statistic and information gain, next different ensemble methods are applied on the reduced data set for classification. From Table 2 shows that number of words for each dataset which varies from 15,868 to 1,73,964. Dimensionality is reduced in order to reduce both time and memory taken by data mining algorithms.

Bagging classifier is created using K-NN classifier and in Adaboost is created using the Naïve Bayes as base classifier. In vote used Naïve Bayes, decision tree and K-NN as base classifier and majority vote from three classifiers is considered as class.

Chi-squared

This is a preprocessing technique used for term reduction. Chi-squared is used for calculating the relevance of the terms with respect to class attribute. The term is more relevant if has higher weight. It is used only for nominal label.

The Chi-square is calculated using below equation [1].

$$X^2 = \text{Sigma}[(O - E)^2/E] \tag{1}$$

In Eq. (1) X^2 is the chi-square statistic, the observed frequency is O and the expected frequency is E. The chi-squared statistic summarizes the divergence between the

expected number of times each result occurs and the observed number of times each result occurs, by summing the squares of the variation, normalized by the expected numbers, over all the categories [15].

Information gain

Information gain is another preprocessing technique used for dimensionality reduction. The information gain is used for calculating the relevance of attributes based on the weights. The term is more relevant if it has highest weight. It calculates the weight of the terms with respect to class attribute. It can only be applied to nominal label [15].

3.1 Adaboost

The most popular boosting algorithm is Adaboost. There are data sets of D of d class-labeled records, $(A_1, c_1), (A_2, c_2), \ldots (A_d, c_d)$, Where c_i is the class label of record A_i. An equal weight of $1/d$ is assigned to each training record.

k rounds are required to generate k classifiers. In round i, the records from the D are sampled to form a training set, D_i, of size d.

The same sample may be selected more than once because the sampling with replacement is used. Based on the weight of sample is selected. The p classifiers are generated in p rounds. Training set D_i of size d is formed the samples of the D in round i. The classifier model M_i is created by using training samples of D_i. Test set D_i is used for calculating the error. If a sample is classified incorrectly then weight is increased otherwise weight is decreased. For generating the training records for the next round weights will be used. More focus is given on the misclassified samples of the previous round [16].

3.2 Bagging

Bagging is also known as Bootstrap aggregating is an ensemble classification technique, which combines the voting from multiple models. Multiple models are of same type. Over fitting can be avoided and also variance can be reduced using bagging [15].

3.3 Random Forest

Random forest is constructed using multiple decision trees or random trees. Each random tree is created using a random subset of features at each split, except this remaining everything is similar to decision tree [15]. It works well if data sets contain

Table 3 Accuracy of ensemble classifier using weight by Chi-squared statistic

	Bagging	Random forest	Voting	Adaboost
PitsA	75.33	56.23	72.93	58.10
PitsB	80.84	48.97	80.48	66.54
PitsC	89.79	79.88	90.10	78.96
PitsD	96.20	92.89	95.64	92.87
PitsE	66.42	63.15	69.45	40.26
PitsF	76.10	64.64	69.45	64.10

more redundant attributes [17]. New test data us classified based on the vote it receives from the multiple random trees. Suppose, if random forest is created using 10 random trees. If 8 random trees classifiers assign class as 4 and remaining two random trees as class 5, then it will be classified as class 4 because of majority votes.

3.4 Voting

Voting ensemble method is present in RapidMiner tool [15]. This method uses a majority vote for classification from the base classifiers provided. Base classifiers can be of different types. Suppose if there are three base classifiers it, if two base classifiers assign class as 3 and another one as 2. It will classify it as severity class 3. Majority vote is 2.

4 Result and Discussion

It will take more time and memory for data mining algorithms to work on huge dimensions (words). That is reason, reduced the dimension to 150 by using two dimensionality reduction methods, i.e., Chi-Square and Information gain. Table 3 shows the accuracies of different ensemble methods after reducing the dimension using Chi-Squared statistic. For PitsA Accuracy varies between 56.23 and 75.33, PitsB accuracy varies between 48.97 and 80.84, PitsC between 78.96 and 90.10, PitsD varies from 92.87 to 96.20, for PitsE varies between 40.26 and 69.45 and PitsF accuracy varies between 64.10 and 76.10. Table 4 shows the accuracies of different ensemble methods after reducing dimensionality using Information gain. For PitsA Accuracy varies between 58.09 and 74.07, PitsB accuracy varies between 54.38 and 80.72, PitsC varies between 79.85 and 89.80, PitsD varies between 93.42 and 96.20, PitsE varies between 69.21 and 72.36 and PitsF accuracy varies between 64.10 and 75.70 using different ensemble methods.

Table 4 Accuracy of ensemble classifier using weight by information gain

	Bagging	Random forest	Voting	Adaboost
PitsA	74.07	58.09	71.27	61.01
PitsB	80.72	54.38	80.35	72.20
PitsC	89.80	79.85	89.79	84.22
PitsD	96.20	93.42	95.64	92.87
PitsE	69.21	63.76	72.36	47.39
PitsF	72.86	64.64	75.70	64.10

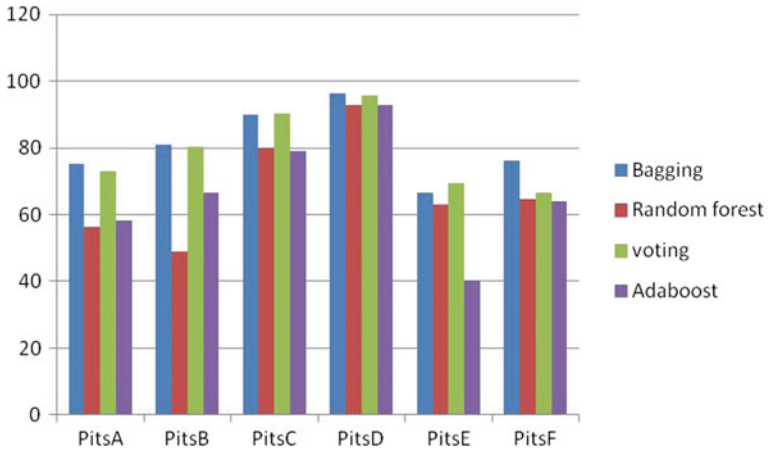

Fig. 1 Accuracies comparison using weight by Chi-squared statistic

Graphical representation of accuracies comparison is shown in Figs. 1 and 2 using Chi-Squared Statistic and Information gain respectively. Figures 1 and 2 show that bagging is given good accuracies over other ensemble methods.

Figure 3 shows accuracies comparison each classifier with different dimensionality reduction, information gain gives slightly good accuracies comparing to Chi-Squared Statistic. Accuracies of bagging and voting algorithm is same, only slight differences in the accuracies of adaboost and Random forest after reducing the dimension using Information gain and Chi-Square.

5 Conclusion

In this paper, predicting the severity of bug report for closed source dataset done using the different ensemble methods such as Bagging, Voting, Adaboost, and random forest. In that bagging is given the good accuracy over oth**er** methods. Also compared the two techniques of dimensionality reduction, i.e., chi-square and information

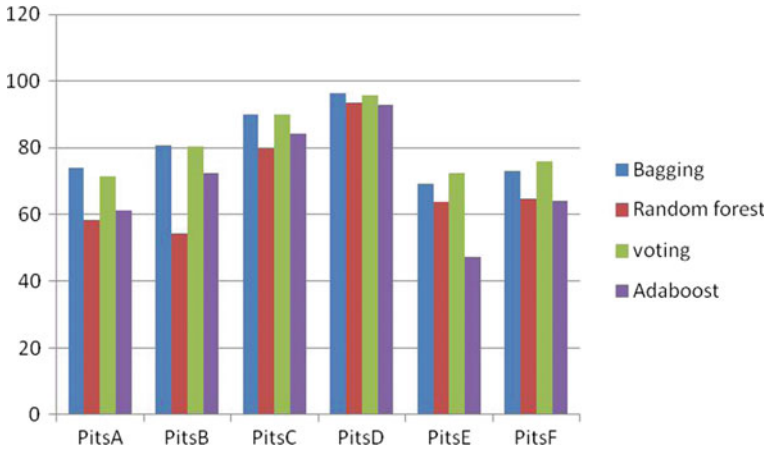

Fig. 2 Accuracies comparison using weight by information gain

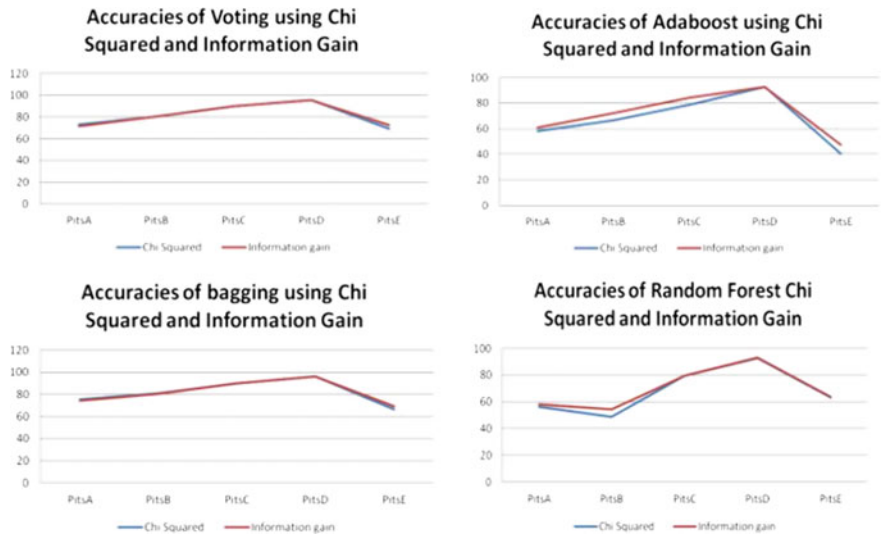

Fig. 3 Accuracies of different ensemble methods using Chi-squared and information gain

gain for reducing the number of dimension. Information gain is given slightly good accuracy over the chi-square. Better prediction of severity for NASA defect reports can be done using ensemble methods which help for improving the quality of software and on time delivery. Future work is done on the data sets of open source software for cross project context.

References

1. Menzies, T., Marcus, A.: Automated severity assessment of software defect reports. In: IEEE International Conference on Software Maintenance, vol. 28, 4 October 2008, pp. 346–355
2. Lamkanfi, A., Demeyer, S., Giger, E., Goethals, B.: Predicting the severity of a reported bug. In: 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), pp. 1–10 (2010)
3. Lamkanfi, A., Demeyer, S., Soetens, Q.D., Verdonck, T.: Comparing mining algorithms for predicting the severity of a reported bug. In: 2011 15th European Conference on Software Maintenance and Reengineering
4. Tian, Y., Lo, D., Sun, C.: Information retrieval based nearest neighbor classification for fine-grained bug severity prediction. In: 2012 19th Working Conference on Reverse Engineering
5. Chaturvedi, K.K., Singh, V.B.: Determining bug severity using machine learning techniques. In: 2012 CSI Sixth International Conference on Software Engineering (CONSEG), pp. 1–6 (2012)
6. Chaturvedi, K.K., Singh, V.B.: An empirical comparison of machine learning techniques in predicting the bug severity of open and closed source projects. In: International Journal on Open Source Software and Processes, vol. 4, issue 2, pp. 32–59 (April 2012)
7. Pushpalatha, M.N., Mrunalini, M.: Predicting the severity of bug reports using classification algorithms. In: International Conference on Circuits, Controls, Communications and Computing (I4C), pp. 1–4 (October 2016)
8. Runeson, P., Alexandersson, M., Nyholm, O.: Detection of duplicate defect reports using natural language processing. In: Proceedings of the 29th International Conference on Software Engineering (ICSE '07), pp. 499–510, May 20–26 (2007)
9. Cubranic, D., Murphy, G.C.: Automatic bug triage using text categorization. In: Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering, pp. 92–97 (June 2004)
10. Anvik, J., Hiew, L., Murphy, G.C.: Who should fix this bug? In: Proceedings of the 28th International Conference on Software Engineering (ICSE '06), pp. 361–370. ACM, New York (2006)
11. Singh, V.B., Misra, S., Sharma, M.: Bug severity assessment in cross project context and identifying training candidates. J. Inf. Knowl. Manage. **16**(01) (March 2017)
12. Jonsson, L., Borg, M., Broman, D., Sandahl, K., Eldh, S., Runeson, P.: Automated bug assignment: ensemble-based machine learning in large scale industrial contexts. Empirical Softw. Eng. **21**(4), 1533–1578 (2016)
13. Pushpalatha, M.N., Mrunalini, M.: Automatic bug assignment using bagging ensemble method. Int. J. Adv. Inf. Sci. Technol. **40**(40) (August 2015)
14. Promise. http://openscience.us/repo/issues/
15. RapidMiner. https://rapidminer.com/
16. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaugmann (2006)
17. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Education (2006)