PREDICTING THE PRIORITY OF BUG REPORTS USING CLASSIFICATION ALGORITHMS

Pushpalatha M N
Assistant Professor, M S Ramaiah University of Technology, Visvesvaraya Technological University, MSR Nagar, MSRIT Post, Bangalore, Karnataka, India
pushpalathamn@msrit.edu
http://www.msrit.edu/department/faculty-detail.html?dept=ise&ID=15

M Mrunalini
Assistant Professor, M S Ramaiah University of Technology, Visvesvaraya Technological University, MSR Nagar, MSRIT Post, Bangalore, Karnataka, India
mrunalini@msrit.edu
http://www.msrit.edu/department/faculty-detail.html?dept=mca&ID=7

Sulav Raj Bista
Student, Dept of ISE, M S Ramaiah University of Technology, Visvesvaraya Technological University, MSR Nagar, MSRIT Post, Bangalore, Karnataka, India
bistasulove@gmail.com

Abstract - Severity of the bug reports are assigned by the user, developer and tester of the software, but priority is assigned by the developers. The developer assigns different priorities such as P1, P2, P3, P4, and P5 where P1 is higher priority and P5 is lower priority. If a bug report contains the priority level P1, then it will be given higher priority for fixing. In order to prioritize which bug to be fixed first based on the priority, priority information is required. Even though, the priority is assigned by developer, sometimes it may incorrect, because of busy schedule or inexperienced developer. That time, developer can use this recommendation system for more accurate priority assignment and also time may be saved.

In this work, predicting the priority of bug report is presented using different classification algorithms such as Naïve bayes, Simple Logistic and Random Tree. Among the three classifiers Simple Logistics gives better result over other two classifiers.

***Keywords*:** Priority, Bug Reports; classification algorithms.

1. Introduction

Open source software development and maintenance receives a lot of bug reports from all over the world. If bug triager starts analyzing each bug reports, it takes lot of resources and time. In the bug triaging, only few bugs will get a chance to fix because of limited resources and time. The bugs are selected based on the severity and priority. Severity information is given by the user when they report the bug, while priority is assigned by the developer for prioritizing which bug to be fixed first. Both severity and priority help for fixing most critical bug first.

Severity information is assigned by the submitter (user or developer). By seeing the severity and other information, priority information is assigned by the developer. This information is used for prioritizing the bug report for fixing. High priority bug report will be given first preference than low priority bug reports. Inexperienced and busy developer may make a mistake in identifying the priority. For avoiding this, many researchers proposed different method for automatically predicting this priority label.

Sample bug report is shown in the Table 3.1, which contains different fields such as BugID which is unique ID generated by the Bugzilla repository, Product which gives the name of the product where bug found, in the product for which particular component bug was found, Assignee gives information about for whom the bug is assigned, the status of the bug, in the life cycle the bug moves into different status like new, assigned, reopened etc, summary about the bug will written in the form of text in this field, severity field tells about how severe the bug from the user perspective. Here, 6 severity levels are present for open source bug reports of Bugzilla and it will vary for other bug reports of other repositories and closed source. 6 severity levels are Blocker, Critical, major, minor, trivial and enhancement and last field is priority which is assigned by the developer and it tells about how urgent it needs to be fixed according to developer perspective.

2. Literature Survey

In [10], authors presented a survey on how data mining is used in software engineering. In [10], authors explained how different researcher used the different data mining techniques like classification, clustering and association mining on the software engineering data set and extracted the useful information out of it to improve the software process. Researchers are trying to improve the quality of software with reduced time and resources using different data mining techniques. After a brief survey of current uses, [6] offered insight into how data mining can make a significant contribution to the success of current software engineering efforts. In [7] authors presented the survey on bug report analysis. They divided work into two parts optimization and usage of bug reports. Some of the work on the bug reports optimization are severity of bug report prediction and misclassified bug report identification, which helps for improving bug reports quality. Again categorized the bug reports usage into two sub parts. Bug report triage and bug fixing. Bug fixing helps to remove the bugs by applying different bug localization techniques. Different work on prioritization of bug reports, automatic assignment and detection of duplicate bug report work is grouped under bug report triage.

In [5], authors done survey of users and developers of open source software such as MOZILLA, APACHE and ECLIPSE for finding out features of a good bug reports and developed the tool called CUEZILLA for measuring the quality of bug reports and it also recommends, which data should be added in order to improve the quality. By taking high quality bug reports will help to improve the accuracy of severity, priority and developer prediction. Assigning the priority to bug report is a manual one. Recently many researchers are working on the automating this task. The new framework is proposed in [1] for predicting the bug report priority and experimentation is done on Eclipse bug datasets taken from the Bugzilla repository. Multiple features are extracted from factors such as textual, product, severity, author, related-report and temporal. This features used for the prediction purpose. For addressing the imbalanced data linear regression is combined with thresholding approach and they called this approach as GRAY. Comparison of GRAY with algorithm such as SVM, NBM and RIPPER is done. Found GRAY performed better than other three. In [2] authors used naïve bayes, Decision tree and random forest classification algorithms on the Firefox and eclipse datasets for priority prediction. They considered two feature sets for experimentation. In the first approach, only textual information present in the summary and description fields are taken as feature sets. In the second approach, component, operating system and severity fields are taken as feature sets. In [2] authors concluded that second feature sets approach perform better than first feature sets approach. In [2], experiment shows that Random forest and decision tree performs better than Naïve byes on their datasets. Moreover, both Random Forests and Decision Trees outperform Naive Bayes. Priority prediction is done using neural network, support vector machine, Naïve bayes and K-nearest neighbour algorithms [3] and cross project prediction is done by taking eclipse version 2 as training set and tested on the eclipse version 3 got accuracy of 72%. Concluded that neural network and support vector machine gave good overall accuracy compared to naïve bayes and k-nearest neighbor on different datasets, also done research on how accuracy varies with number of terms. In [4] Multilayer Perceptron and naïve bayes is used for predicting the bug priority and done experimentation on the five different versions of the eclipse. Used different features such as textual information, author, related bug reports, severity of bug reports, product name where bug found and at the same time other bug reports reported. In [8] and [9], authors used emotions word present in the bug description for predicting the priority and severity of bug reports.

In [11], authors used the Naïve Bayes classification algorithm for automatic bug assignment using Eclipse bug reports got the precision of 30%. In [12] the authors used Mozilla bug report, comparison is done with 5 different classification algorithm such as Naïve Bayes, RBF network, Decision table, REP Tree, Random Forest, Support vector machine and J48, feature selection and feature extraction is done using Latent Semantic Indexing. Support vector machine with combination latent semantic indexing for 100 terms it is giving the accuracy of 44.4%. In [13], the term selection using 5 different term selection algorithms such as Log Odds Ratio, Chi-Square, Term Frequency Relevance Frequency, Mutual Information and Distinguishing Feature Selector, then efficient Bug triaging is done using Naïve bayes algorithm got F-score of 6.2%, 38.2%, 26.5%, and 12.1% on Eclipse for SWT, Eclipse, Netbeans and Maemo product respectively. [14] Naïve Bayes, Nearest neighbor, SVM, rules, Expectation maximization and C4.5 machine learning algorithm used for Eclipse, Firefox, gcc, MyLyn and Bugzilla achieved the precision between 70% to 98% for recommending the developer for bug report. In [15], authors used Bagging and Naïve Bayes classifier for automatic bug assignment to developer. Comparison is performed among the two classifier and they found that Bagging gives good accuracy over Naïve Bayes on the given datasets.

In [16] predicting the severity is done using naïve byes classification analyzing the one line description of bug. It helps for developer to decide how soon it should be fixed. For each severity 500 bug reports are taken in the training set. Different text mining techniques such as tokenization, stop word removal and stemming are applied. Comparison is done on the different text classification algorithms such as K-nearest neighbor, support vector machine, naïve bayes and naïve bayes multinomial in [17]. Concluded that naïve byes multinomial gave good accuracy over other three. In [18], authors used Naïve Bayes classification for predicting the severity of android bug reports. Eclipse and Mozilla bug datasets are taken from Bugzilla bug repositories for training and for testing

android bug reports are taken from the Android bug tracker system which are unlabeled. Text Preprocessing is done in order to extract more relevant keywords or features from text, next features are reduced using Importance Degree Reduction (IDR) technique based on rough set. IDR with NB approach gave good accuracy for eclipse and Mozilla bug repository datasets. In [19] experimentation is done on both open source and closed source datasets using different classification algorithms such as support Vector Machine, J48, Naïve Bayes, RIPPER and Random Forest. In [20], authors used Bagging and J48 algorithms for severity prediction; comparison is carried out among two algorithms. They found out that Bagging gives good accuracy over J48 algorithm on the given data set.

3. Framework for automatic priority prediction

Bug reports are taken from Eclipse bugzilla repository. Product, component, assignee, summary and severity field are considered for prediction of priority field. For the experimentation WEKA Machine learning tool is used. Dataset is given in the Table 3.1.

Dataset contains class imbalance, P1 class bug reports are very less compared other priority classes, for that SMOTE algorithm is used for solving the imbalance problem. A SMOTE stand for synthetic Minority oversampling technique is an oversampling method used for class imbalance.

Next summary is converted into string using the filter nominal to string, then stringtowordsvector filter is used for converting the string to word vector, we first tokenized into words, then stopwords are removed and stemming is done, next tf-idf weight is assigned for each word.

Table 3.1. Sample bug report which contains the priority

Bug ID	Product	Component	Assignee	Status	Summary	Severity	Priority
18145	Working Groups	Location Tech	locationtech. wg-inbox	NEW	Website update	major	P3
03276	Working Groups	Location Tech	nathan	NEW	Cannot change time zone setting on Location Tech	normal	P3
91185	Platform	UI	Platform-UI-Inbox	NEW	Change default in project explorer to hierarchical mode	enhance ment	P5
29199	Platform	UI	platform-ui-triaged	NEW	[Editor Mgmt] Ghost editors in workbench	major	P1
406646	Platform	Releng	platform-releng-inbox	ASSIGNED	Fix test framework documentation	normal	P2
07007	Platform	Debug	platform-debug-inbox	ASSIGNED	[breakpoints] flash when deleting breakpoints	minor	P4
2742	Platform	Debug	platform-debug-inbox	ASSIGNED	[debug view] Move preferences to debug view	enhance ment	P5
518145	Working Groups	Location Tech	locationtech. wg-inbox	NEW	Website update	major	P3

3.1 ***In the first step, the eclipse bug reports are taken from the Bugzilla bug repository -.*** Considered product, component, assignee, summary and severity field for prediction of priority field Numbering and spacing.

3.2 ***Preprocessing-*** Dataset contains class imbalance, P1 class bug reports are very less compared other priority classes, for that SMOTE algorithm is used for solving the imbalance problem. Next summary is converted into string using the filter nominal to string, then stringtowordsvector filter is used for converting the string to word vector, we first tokenized into words, then stopwords such as a, an, the, in etc are removed, because they are not useful for classification. Next using porter stemmer [17] stemming is done, which transform each word into basic form, for example, it transforms propose, proposed and proposing into basic form propose.

3.3 ***Split the preprocessed datasets and testing datasets dataset into training*** - 80% of datasets is taken for training and remaining 20% is used for testing.

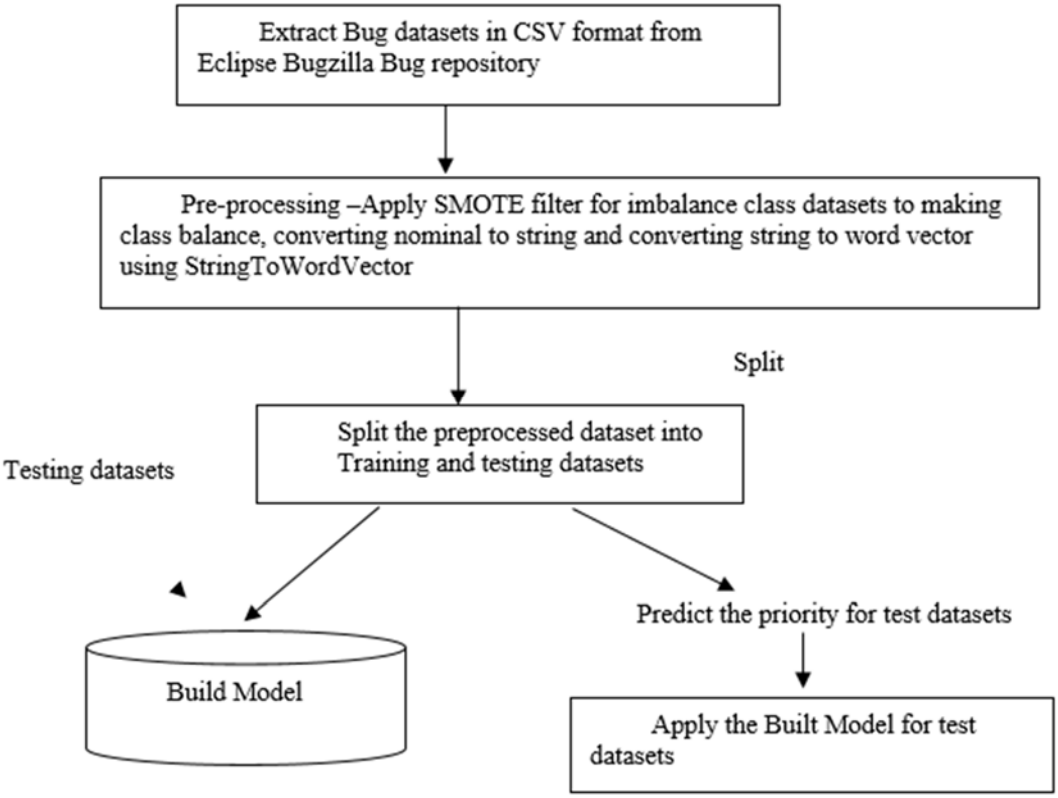

Figure 2.1 Framework for automatic priority prediction

3.4 ***Training dataset is used for building the model,*** the model is built using three such as classification algorithms Naïve bayes, Random tree and Simple Logistic

3.4.1 **Random tree:** Random tree is one of the supervised technique, which constructs the tree by considering the n randomly selected features at each node and no pruning is performed.

3.4.2 **Simple Logistic:** Decision Tree Induction and linear logistic regression models are combined to create Logistic Model Tree (LMT) for building model [14] [15]. Linear Logistic regression models are used for building classifiers.

3.4.3 **Naïve Bayes:** Each feature or word are independent (no dependence between words)

Let S is the datasets which contains m number of bug reports with associated class labels n-d words set is represented as $M=(m_1,m_2,m_3\ldots\ldots m_n)$

Suppose there are x classes $L_1,L_2\ldots\ldots L_x$. The class which gives highest posterior probability will be assigned to new tuple M. i.e. the naïve bayes classifier predicts for the tuple M to class L_i when below condition satisfies

$P(Li|M)>P(Lj|M)$ for $1\leq j\leq x$, j not equal to i.

Using bayes' theorem, we have to find the the maximum posteriori $P(L_i||M)$ using below equation 3.1

$$P(Li|M) = \frac{P(M|Li)P(Li)}{P(M)} - - - - - - - - - - - - - - - - 3.1$$

3.5 **Apply the built model** on the testing datasets for predicting the priority of bug reports

4. Results and Discussion

The accuracy is found out using equation 8.4. Naïve Bayes gave 65%, Random tree 60% and for simple logistic gave 78% of accuracy.

Table 4.1 Accuracy Comparison of classification algorithms

Simple Logistic	Random Tree	Naïve Bayes
78%	60%	65%

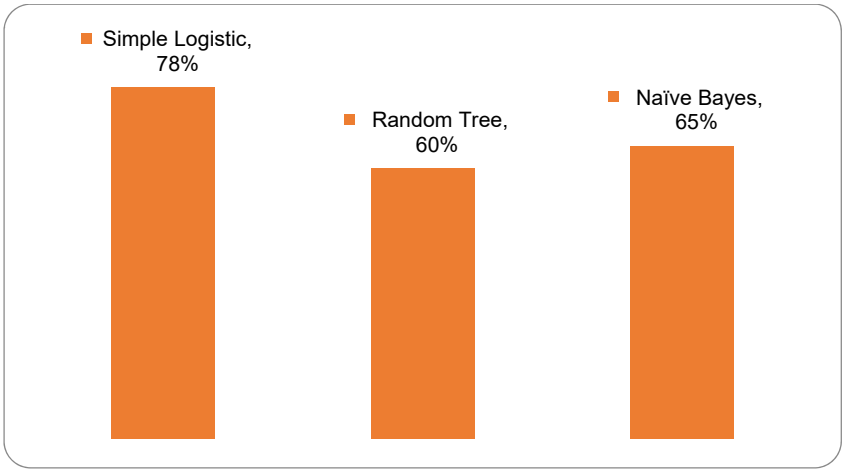

Figure 4.1 Accuracy comparison

Priority Class of Bug report is P

$$\text{Precision}=\frac{\text{Number of Priority of bug reports correctly predicted as P}}{\text{Number of Priority of bug reports predicted as P}} \quad ---------- 4.1$$

$$\text{Recall}=\frac{\text{Number of Priority of bug reports correctly predicted as P}}{\text{Number of Priority of bug reports P}} \quad ---------- 4.2$$

$$\text{Accuracy}=\frac{\text{Total number of bug reports correctly predicted}}{\text{Total Number of Prediction}} \quad ---------- 4.3$$

$$\text{F-Measure}=\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}} \quad ---------- 4.4$$

Accuracy is calculated using equation 4.3 which tells about the percent of prediction which were correct. Precision is also known as Positive prediction value. Percent of true positive among all the positive prediction is calculated using equation 4.1.

Recall is also known as True positive rate or sensitivity. Percent of correct prediction is calculated using equation 4.2

The harmonic mean of precision and recall is F-Measure is calculated using equation 4.4

Table 4.2 Result Evaluation for Simple Logistic

Precision	Recall	F-Measure	Class
1.00	0.974	0.987	P1
0.86	0.85	0.85	P2
0.75	0.87	0.80	P3
0.48	0.28	0.35	P4
0.67	0.67	0.67	P5

Precision varies between 0.48 to 1.00, Recall varies varies between 0.28 to 0.97 and F-Measure varies between 0.35 to 0,98 using Simple logistic.

Table 4.3 Result Evaluation for Random Tree

Precision	Recall	F-Measure	Class
0.821	0.979	0.893	P1
0.607	0.706	0.653	P2
0.583	0.526	0.553	P3
0.702	0.333	0.258	P4
0.710	0.549	0.498	P5

Precision, Recall and F-measure varies between 0.58 to 0.821, 0.333 to 0.979 and 0.258 to 0.893 respectively for Random Tree.

Table 4.4 Result Evaluation for Naïve Bayes

Precision	Recall	F-Measure	Class
1.000	0.974	0.987	P1
0.644	0.885	0.746	P2
0.769	0.384	0.512	P3
0.333	0.434	0.377	P4
0.593	0.682	0.635	P5

Precision, Recall and F-measure varies between 0.33 to 1.00, 0.434 to 0.974 and 0.377 to 0.987 respectively for Naïve Bayes.

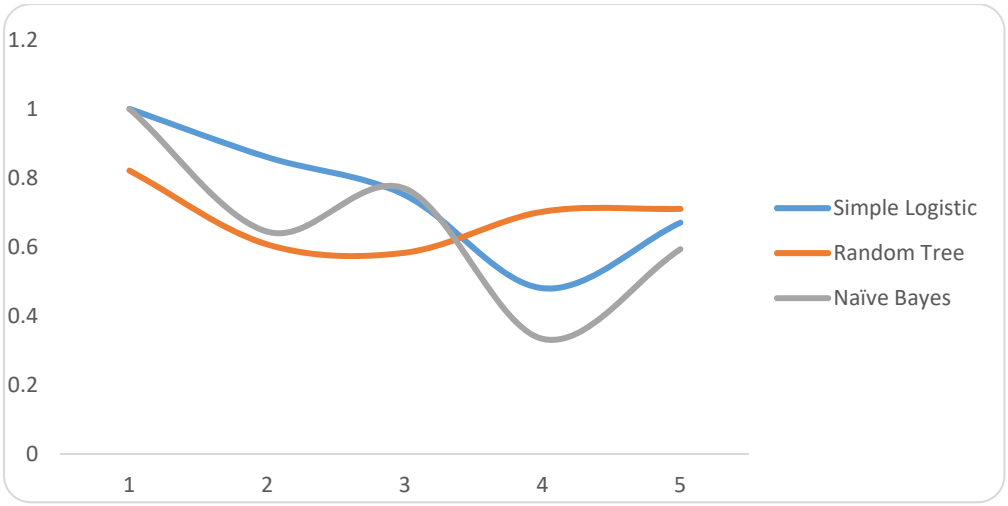

Figure 3.2 Precision comparison

Figure 3.2 shows the precision comparison of all the three classifiers.

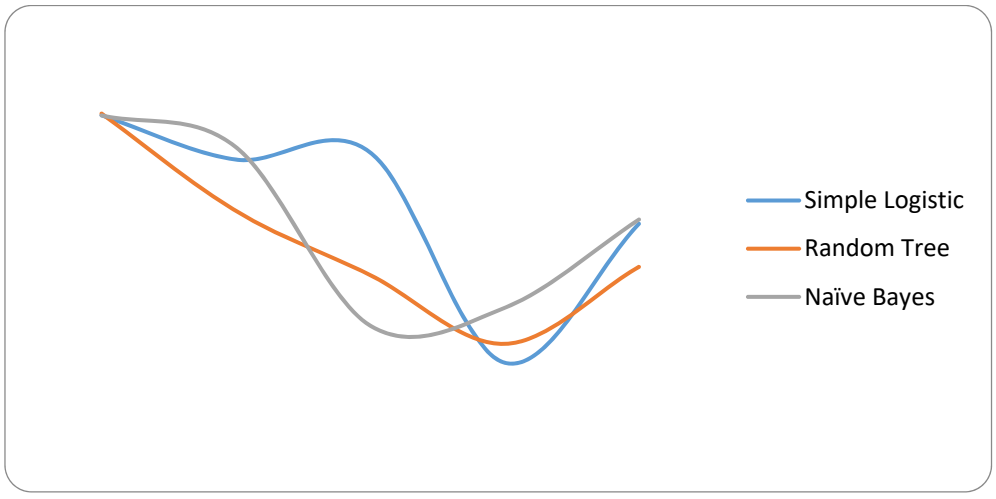

Figure 3.3 Recall comparison

Figure 3.3 shows the recall comparison of all the three classifiers.

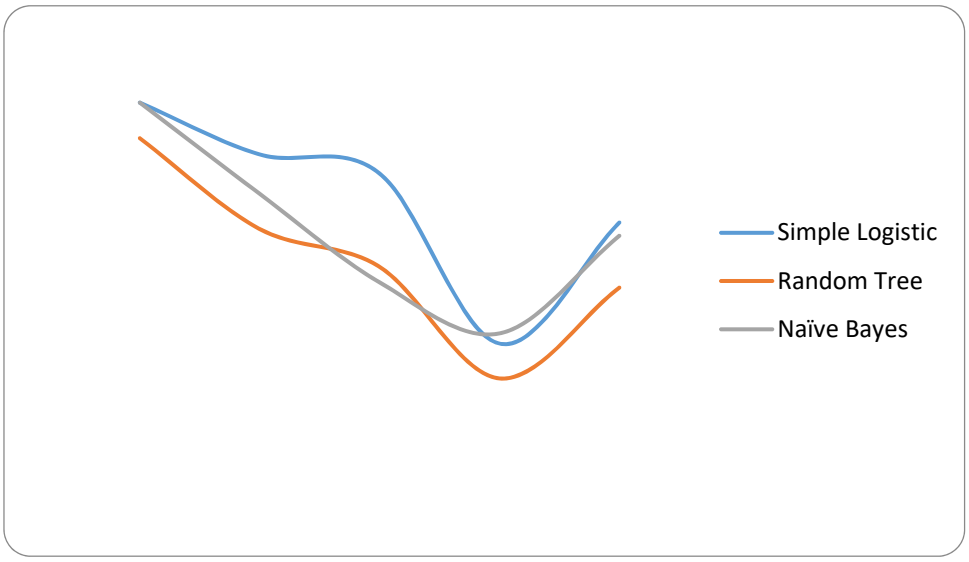

Figure 3.4 F-measure comparison

Figure 3.4 shows the F-measure comparison of all the three classifiers.

Predicting the priority of bug reports is carried out using different classification algorithms such as Naïve bayes, Random tree and simple logistic. For the given datasets simple logistic gives better accuracy over other two algorithms. Results also evaluated using other measures such as Precision, Recall, F-measure and ROC Area. From the table 17-19, we can see that Simple Logistic gives better result over other two algorithms.

5. Conclusion and Future Work

Open source repositories lot of bug reports are submitted by all around the world by user and developer. In order to prioritize which bug to be fixed first based on the priority, we need priority information. Even though, the priority is assigned by developer, sometimes it may incorrect, because of busy schedule or inexperienced developer. That time, developer can use this recommendation system for more accurate priority assignment and also time may be saved.

Predicting the priority of bug reports is done using different classification algorithms such as Naïve bayes, Random Tree and Simple logistics. Simple logistics gives good accuracy comparing to other two.

References

[1] Yuan Tian, David Lo, and Chengnian Sun," DRONE: Predicting Priority of Reported Bugs by Multi-Factor Analysis", IEEE International Conference on Software Maintenance, Pages 200-209, 2013.

[2] Alenezi, Mamdouh & Banitaan, Shadi. "Bug Reports Prioritization: Which Features and Classifier to Use?", Proceedings - 2013 12th International Conference on Machine Learning and Applications, ICMLA 2013. 2. 112-116. 10.1109/ICMLA.2013.114.

[3] Meera Sharma, Punam Bedi, K.K.Chaturvedi, V.B.Singh, " Predicting the Priority of a Reported Bug using Machine Learning Techniques and Cross Project Validation", 12th International Conference on Intelligent Systems Design and Applications (ISDA), 2012, 10.1109/ISDA.2012.6416595.

[4] Pooja Awana Choudhary, Dr. Satwinder Singh, "Neural network based Bug priority prediction model using text classification techniques", 2017.

[5] Nicolas Bettenburg, Sascha Just, Adrian Schröter, "What Makes a Good Bug Report?", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, pp.618-643, VOL. 36, NO. 5, SEPTEMBER/OCTOBER 2010.

[6] Quinn Taylor, Christophe Giraud-Carrier," Applications of data mining in software engineering", Int. J. Data Analysis Techniques and Strategies, Vol. 2, No. 3, 2010.

[7] ZHANG Jie , WANG XiaoYin, HAO Dan, XIE Bing, ZHANG Lu & MEI Hong, "A survey on bug-report analysis", Sci China Inf Sci, 2015, 58:021101(24).

[8] Qasim Umar, Hui Liu, And Yasir Sultan "Emotion Based Automated Priority Prediction for Bug Reports", pp. 35743 - 35752, 02 July 2018, IEEE Access, Volume 6.

[9] Geunseok YANG, Tao ZHANG, Byungjeong LEE, "An Emotion Similarity Based Severity Prediction of Software Bugs: A Case Study of Open Source Projects", Volume E101.D, Issue 8 , Pages 2015-2026, 2018.

[10] M. Halkidia, D. Spinellis, G. Tsatsaronis and M. Vazirgiannis, "Data mining in software engineering", Intelligent Data Analysis, 2011, pp. 413–441.

[11] D. Cubrani c and G. C. Murphy. "Automatic bug triage using text classification". In Proceedings of Software Engineering and Knowledge Engineering, pages 92–97, 2004.

[12] Ahsan S, Ferzund J, Wotawa F, "Automatic Software Bug Triage System (BTS) Based on Latent Semantic Indexing and Support Vector Machine." In: Proc. of the 4h International Conference on Software Engineering Advances, pp 216_221, 2009.

[13] Alenezi M, Magel K, Banitaan S "Efficient Bug Triaging Using Text Mining". Journal of Software 2013, pp 2185-2190, 2013.

[14] Anvik J, Murphy GC, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions". ACM Transactions on Software Engineering and Methodology, Volume 20 Issue 3, August 2011.

[15] Pushpalatha M N, Dr. M. Mrunalini," Automatic Bug Assignment using Bagging Ensemble Method", International Journal of Advanced Information Science and Technology, ISSN: 2319:2682 , Vol.40, No.40, IF Value: 5.032(SJIF): DOI-10.153, PP. 98-103, August 2015.

[16] Lamkanfi A; Demeyer S; Giger E; Goethals, B," Predicting the severity of a reported bug", In 7th Working Conference on Mining Software Repositories, Cape Town, South Africa, pp.1-10, May 2010

[17] Ahmed Lamkanfi, Serge Demeyer, Quinten David Soetens, Tim Verdonck, "Comparing Mining Algorithms for Predicting the Severity of a Reported Bug", 2011 15th European Conference on Software Maintenance and Reengineering, pp. 249–258, IEEE, 2011.

[18] Shikai Guo , Rong Chen, and Hui Li," Using Knowledge Transfer and Rough Set to Predict the Severity of Android Test Reports via Text Mining", Symmetry, 9(8), 161, https://doi.org/10.3390/sym9080161, 2017

[19] Chaturvedi K.K., Singh, V.B,"An empirical Comparison of Machine Learning Techniques in Predicting the Bug Severity of Open and Close Source Projects", International Journal of Open Source Software and Processes 4(2), 32–59, April-June 2012.

[20] Pushpalatha M N & Mrunalini M, "Predicting the severity of bug reports using classification algorithms",10.1109/CIMCA.2016.8053276, International Conference on Circuits, Controls, Communications and Computing (I4C), pp. 131-134, October 2016, Benguluru, India