AN EFFECTIVE AND ACCURATE BUG TRIAGE SYSTEM WITH SVM AND KNN

Jagruti R. Patil[1], Swapnaja Suryawanshi[2]
[1]Savitribai Phule Pune University, Indira College of Engineering and Management, Pune, Maharashtra, India
[2]Professor, Savitribai Phule Pune University, Indira College of Engineering and Management, Pune, Maharashtra, India

ABSTRACT

Bugs are very crucial aspects in a software corporation. Bug triage is a necessary step in a software company and it cannot be avoided. In bug triage a correct developer should be assigned for fixing the bug. To manually perform the bug triage is very expensive and even lengthy process. In manual bug triage human triagers are there who perform the bug triage. So existing techniques uses text classification. In this techniques automatic bug triage is performed. But still there is a problem of large data, so there is a need the data should be reduced which will increase the quality of the data. So To perform data reduction instance selection and feature selection methods are used at the same time. The sequence for applying instance selection and feature selection should be predicted, and to know that order to extract the attributes from bug data sets. Even the task of domain specific bugs is performed and accurate developer assignment is performed in this system. And as a result it shows that the data is reduced tremendously which gives high quality of bug data sets.

KEYWORDS – *Bug repository, Bug data reduction, Bug triage, kNN, SVM*

I. INTRODUCTION

Due to the complexity of software development, bugs are unavoidable and it is necessary to resolve all the bugs. Bug resolution, which means the diagnosis, fixing, testing, and documentation of bugs, is a significant activity in software development and maintenance. Consider a programming developer working on a long period software development product, like most programming developers generally makes modification that are bug free and not containing a hidden bug. Sometimes, developer makes changes that set up new features or adjust the software to a changing operational environment. Developers construct bug fixing modifications that patch up bugs. Occasionally, developers make a bug launching change and introduce wrong statements into the source code. Since programmers usually do not understand that they are generating erroneous software, there is always the question of introducing a bug. After some time programmer get the bug report, developer must spend time with the source code and the recent changes made to it.

The bug repository mining has employ data mining to deal with software engineering problems. A software repository has a large-scale database which is used for storing the output of software development. Usually for huge-scale and complex data in software repositories, software analysis is not completely suitable. So data mining techniques, mining software repositories can discover fascinating data in software repositories and solve real world software problems.

The main goal of data reduction for bug triage to construct a small scale and high superiority set of bug data by removing bug reports and words which are redundant or not-useful. So instance selection and feature selection techniques are used at the same time to reduce the bug dimension and the word dimension. The reduced bug data have small number of bug reports and smaller number of words than original bug data. And they also provide analogous data than original bug data. The instance selection

means subset of related instances i.e. bug report in bug data and the feature selection means subset of related features i.e. words in bug data.

II. MOTIVATION

Real world data always consist of noise and redundancy. Noisy data may mislead the data analysis techniques while redundant data increase the cost of data processing. In bug repositories all the bug reports are filled by developer in their own languages. In bug repositories the low quality bugs accumulate with the growth in scale. Such low quality and large scale data may affect the performance of bug fixing. So there is need to reduce the scale and improve the quality of bug data.

III. LITERATURE SURVEY

Bug repositories are generally utilized for handling software bugs. A lengthy stride of handling software bugs is bug triage, which expects to allocate a right developer to fix a new bug [1]. Due to the significant number of consistently bugs and the lack of skill of the considerable number of bugs, manual bug triage is costly in time cost and low in terms of accuracy. Cubranic and Murphy first recommend the issue of automatic bug triage to lessen the expense of manual bug triage [3]. They apply text categorization method to predict specific developer that should handle the bug based on the bugs details. Jeong et al. discover that more than 37 present of bugs have been reassigned in manual bug triage [7]. They recommend a tossing graph process to lessen reassignment in bug triage. Park et al. change over bug triage into an advancement issue and propose a collaborative filtering approach to deal with decreasing the bug fixing time [18].

Anvik et al study different technique on bug triage, including data preparation and distinctive classifiers [2]. Open source development projects normally support an open bug storage facility to which both developers and customers can report bugs. The reports that emerge in this storehouse must be triaged to make sense of whether the report is one which requires consideration and in the event that it is, which developer will be allocated the commitment of resolving the report. Broad open source advancements are troubled by the rate at which new bug reports show up in the bug store. In this paper shows a semi-automated approach expected to simplicity one piece of this procedure, the task of reports to a developer [2]. To analyse the interrelations in bug data, Sandusky et al. build a bug report network to assess the interdependency between bug reports [11]. Also examine connections among bug reports, Hong et al. build a developer public network to analyse the relationship among developers in view of the bug data in Mozilla project [6]. This developer public network is helpful to understand the developer society and the project advancement. By plotting bug priorities to developers, Xuan et al. know the developer prioritization in open source bug storehouses [8].

The developer prioritization can separate developers and help assignments in programming upkeep.

Bettenburg et al. led an overview on developers and clients from APACHE, ECLIPSE, and MOZILLA to figure out which data substance, in their suppositions, contain great quality bug reports [16]. Their CUEZILLA apparatus utilized the reactions from the study to quantify nature of bug reports progressively and give prompt criticism to journalists on upgrading data quality. Just et al. dissected the reactions from the same overview to recommend enhancements to make bug following frameworks less demanding to utilize and encourage accommodation of better quality bug reports [17]. In conclusion, another study by Bettenburg et al. contended for converging of copy bug reports alongside the firsts to make more extraordinary data about the bug accessible to designers [14]. Conversely, author work concentrates on the collaboration in the middle of developers and clients with the objective of enhancing apparatus support for this communication. Sandusky and Gasser concentrated on the part of arrangement in programming issue administration and how it influences the association of data [13]. In existing work on bug repository engineers are constantly treated similarly. Be that as it may, the need of a designer assumes a critical part in the errands. For instance, a dynamic designer might make a larger number of commitments on bug settling than a dormant one; an accomplished analyser might discover bugs with higher severities than a typical end client [9]. In this paper, they rank every one of the engineers of bug repository to help the undertakings around bug repository. We indicate the procedure of creating the engineer needs as developer prioritization. For

bug information, a few different tasks exist once bugs are triaged. For e.g. severity identification which aims to distinguish the significance of bug reports for further planning in bug handling [19].

IV. PROPOSED SYSTEM

To fix the bugs in a software company bug triage is the process which is used. In this process accurate developer is assigned for fixing the particular bug. But manual bug triage process is very lengthy and expensive process. So to avoid time and cost in manual bug triage, automatic bug triage is used in which text classification techniques are used. The problem which is addressed for this is the large bug dataset. And the large bug dataset affects the quality of the bug datasets. So to reduce the bug dataset we use feature selection and instance selection techniques as shown in block diagram. This techniques reduces the bug data in both bug and word dimensions. And even we want to know the order of applying the instance selection and feature selection for this the attributes of the historical bug datasets are extracted. This gives us the reduced and quality bug dataset. Even would like to build the domain specific system for which the domain relevance class labels are generated. And the necessary things which need to be improved are the accuracy of the developer assignment. Correct developer assignment would increase the quality of the bug triage which is the main aim of the system. So weight estimation is performed for assigning the accurate developer.

V. SYSTEM ARCHITECTURE

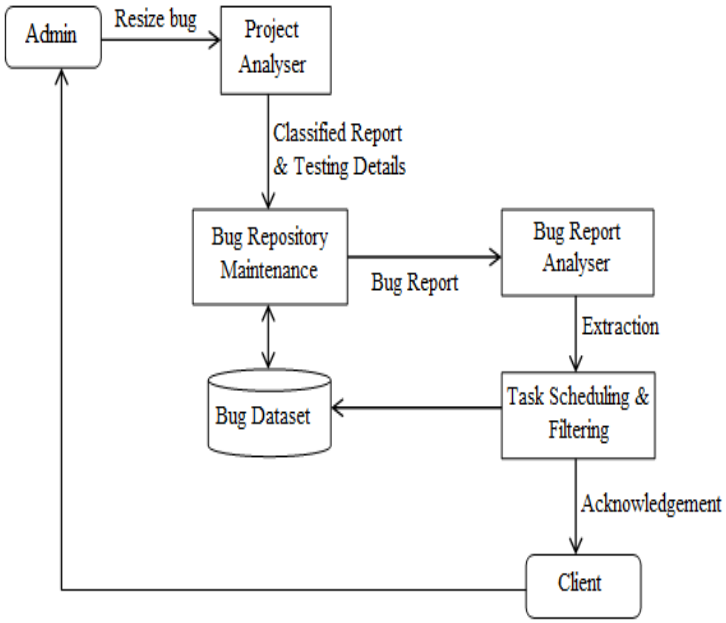

Figure 1: System architecture

A. Project Analyzer:

Project Analyzer gives the information of bug report to user. It is tool which has through support for detecting the dead code, optimizing the code, optimizing the coding styles and implementation styles. This tool provides the functionalities which reduces the cost and improves the code maintainability resulting in delivering quality product. It also provides certain approaches for documentation features such as reporting documents and diagramming the tree structures. It can also provide facility to provide documentation in case of uncommented code. This approach helps us to collect all the information of software enlargement, like software halt configuration, module measurement, improvement scheme, module needs, database set, group details, supervisor details, originator information, tester details, organization testing team, services details, and external reference. These details are used in module development, other organization tool detail, module working details etc. from the companies perspective.

B. Bug Repository Maintenance:

The Bug Repository dataset is used to gather information regarding software systems modules and historical records. Objective of such system is to match different bug expectation methods to evaluate new approach which will help to improve system result.

C. Bug Report Analysis:

In bug reports analysis artefacts describe software bugs, especially in open-source software. Lately, due to the availability of a large number of bug reports, a considerable amount of research has been carried out on bug-report analysis, such as automatically checking duplication of bug reports and localizing bugs based on bug reports.

D. Task Scheduling & Notification:

This module has ability to write own solution for problems which also gives acknowledgement to customer for bug report which is again used for further bug report analysis phase.

VI. ALGORITHM

Three familiar classifiers are used namely the Support Vector Machine, the *k* Nearest Neighbor and the Naïve Bayes. These classifications methods are considered because of their straightforward and improved performance behavior.
For classification Support Vector Machine is used and for comparison k Nearest Neighbor is used. To remove noise data and redundant dataset, data reduction techniques such as instance selection and feature selection are used in Naïve Bayes.
Let us contemplate the classification function Φ of the data points to x_i ($i = 1...l$) add into a class $y_i \in C = \{+1, -1\}$. Let us define *d* it measurement of the data space. The three classification algorithms are represented in the following subsections:

A. Support Vector Machine (SVM)

The purpose of SVM is to calculate decision surface which is used to maximize the edge between the data points of two classes. Classification document contains following results and previously published studies. The *dual* form of the *linear* SVM optimization problem is to maximize:

$$\propto^* = maximise_{\propto} \sum_{i=1}^{l} \propto_i \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \propto_i \propto_j \langle x_i, x_j \rangle,$$

$$subject \sum_{i=1}^{l} y_i \propto_i = 0$$

$$0 \leq \propto_i \geq C, i = 1 \dots l$$

In the equation, weight of the samples is assigned to α_i and *C* is a relation complexity of model and error value. For point x′, expectation $\hat{\Phi}(x')$ is given by,

$$\hat{\Phi}(x') = sign\left(\sum_{i=1} \propto_i y_i \langle x_i, x' \rangle + b\right) = sign(\langle w', x' \rangle + b)$$

Where, $w^* = \sum_{i=1}^{l} \propto_i y_i x_i$

B. k Nearest Neighborhoods (kNN)

Most similar points from the dataset k, is always used for predefined comparison. For each respective class y_i, sum of similar neighbors value of the same class. Then, the class y_i score is allocate to the data point x′ by the *k* nearest neighbor's algorithm.

$$\hat{\Phi}(x') = argmax_{y_i \in C} \sum_{i=1}^{l} \delta\big(y_i, \Phi(x_i)\big) \sim (x_i, x')$$

C. Naïve Bayes

Let $P(y_i)$ is the preceding possibility of the class y_i and $P(a_j' | y_i)$ be the restrictive probability observe attribute value a_j' that gives the class y_i. Then, a naive Bayes classifier shares a data point x' with attributes $(a_1' \ldots a_d')$, the class $\hat{\Phi}(x')$ maximizing:

$$\hat{\Phi}(x') = argmax_{y_i \in C} P(y_i) \prod_{j=1}^{d} P(a'_j | y_i)$$

D. Instance Selection (IS) and Feature Selection (FS)

For data processing and data reduction feature selection and instance selection are used. Instance selection is a data reduction technique removes number of instance to remove noise data and redundant instances. For choosing a compact dataset for creating a comprehensive dataset feature selection is a pre-processing data reduction technique.

E. Evaluation Methodology

To guess classifier enactment, classical cross maintenance is used. The macro is an average of $F_1 measure MF_1 = \frac{2 \times MPrecision \times MRecall}{MPrecision + MRecall}$, where the MPrecision stands for the preparation time and MRecall stands for testing time. *MPrecision* and the *MRecall are* averages of the precision and the recall calculated on the basis of the two problem environments (in one, a class is measure calculation which can be positive or negative; in the other the task tread of). As a final point, verify the total handling time calculated in seconds (the addition of the preparation and the testing time). As the dimension of the test data set is almost same for each experiment, this preprocessing time replicates frequently sequence time of classifiers.

VII. RELATED WORK

The Life cycle of bug report: Bug has to from several states. When a bug report is first submitted to the repository, its status is set to NEW. Once a developer is assigned, the status is set to ASSIGNED. When a report is closed its status is set to RESOLVED. It can be further also be marked as being verified (VERIFIED) or closed for good (CLOSED). If the resolution resulted in a change to the code base, the bug is resolved as FIXED. When a developer determines the report is a duplicate of an existing report then it is marked as DUPLICATE. If the developer was unable to resolve or reproduce the bug it is indicated by setting status to WORKSFORME. If the report describes a problem that will not be fixed, or is not an actual bug, the report is marked as WONTFIX or INVALID, respectively. A resolved report may be reopened at a later for use, and then its status set to REOPENED.

Bug Triage System: In A time-consuming step of handling software bugs is bug triage, which aims to assign a correct developer to fix a new bug. Here, we address the problem of data reduction for bug triage, i.e., how to reduce the bug data to save the labour cost of developers and improve the quality to facilitate the process of bug triage. Data reduction for bug triage aims to build a small-scale and high-quality set of bug data by removing bug reports and words, which are redundant or non-informative.

Bug Repository: A bug repository is also known as software repository which is used storing information of bugs. The bug triage is essential steps for fixing a bug which are appropriately assigning a developer to new bug. For open source huge-scale software projects, the number of day-to-day bugs is so enormous which creates the triaging process very hard and challenging. Software companies pay most of cost in fixing bugs. In a bug repository, a bug is maintained as a bug report, which records the textual description of replicating the bug and updates according to the status of bug fixing. In bug repository, bug reports are called bug data. There are two leading difficulties in software development associated to bug data that may pretend on bug repositories that are the huge scale and the low Superiority. Because of regularly reported bugs, enormous number of new bugs is stored in bug repositories. And low quality bugs are noisy and redundancy. Noisy bugs may worthless data that are related developers and redundant bugs means the same attribute may have different name in different database.

VIII. RESULT

Bug report which is generated from different system is in the XML format and contains generation time, date, bug status, platform and bug description etc. Assigning this bug report as follows:

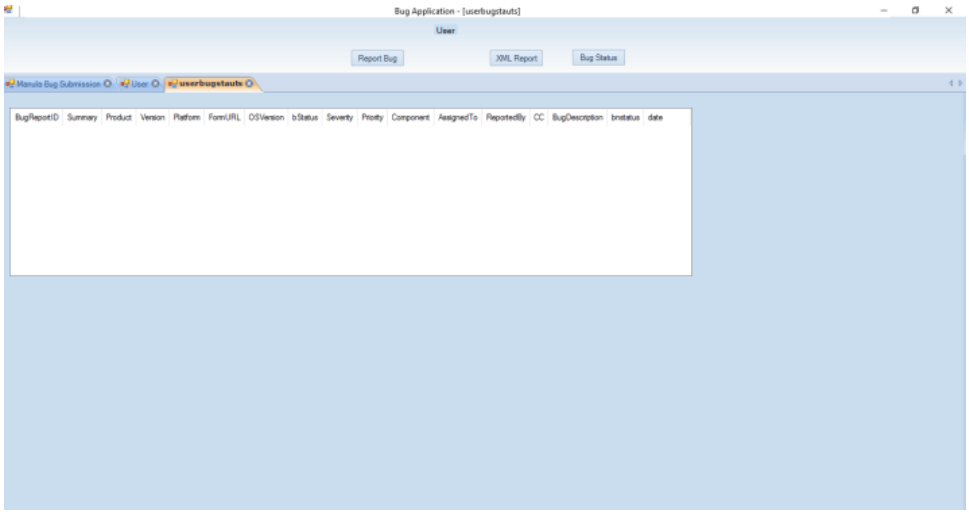

Figure 2: Bug Status

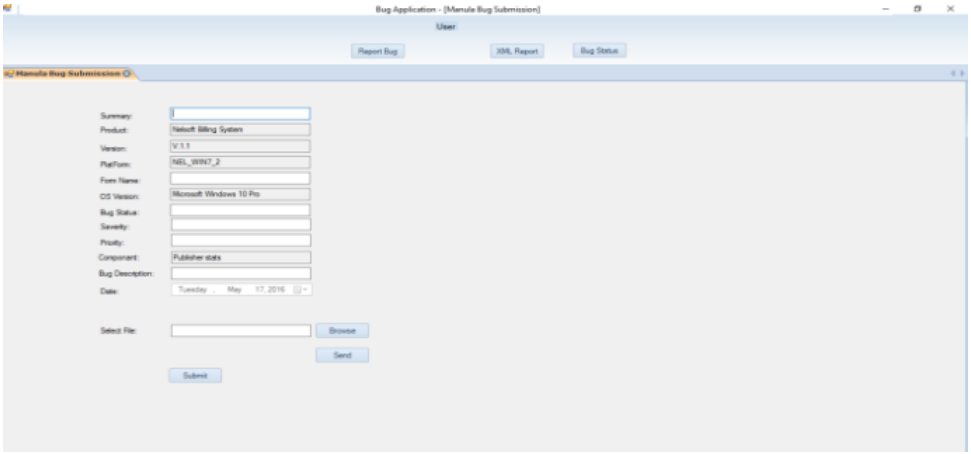

Figure 3: User Manual Bug

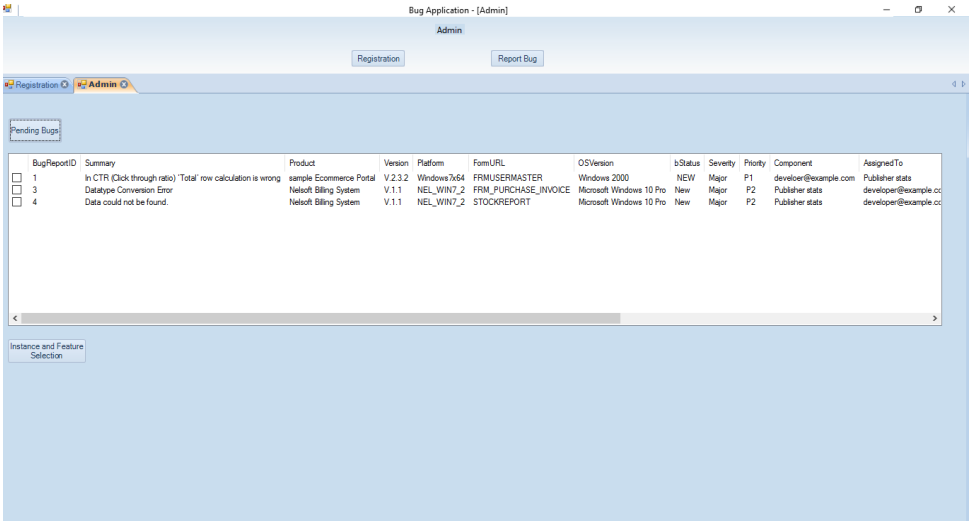

Figure 4: Bug Report

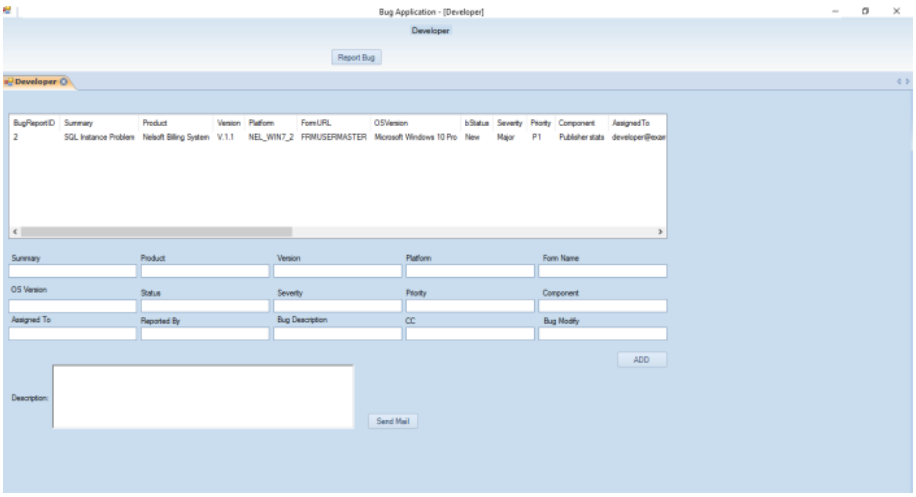

Figure 5: Developer Bug Report

In the above screenshots Figure 2 shows bug status at admin side to allocate the bug report to team leader. After that team leader assign bug report to the developer to resolve the bug. The figure 3 shows the bug report details. In the figure 4 it shows how user can send bug report manually to admin. The figure 5 shows developer side how developer view report and send solution to user also update status at admin side.

Table 1: Comparison With similar System

Data size in KB	Bug Report Classification Time in ms [Existing]	Bug Report Classification Time in ms [Proposed]
100	15	8
200	18	11
300	23	13
400	26	17
500	31	20
600	35	24

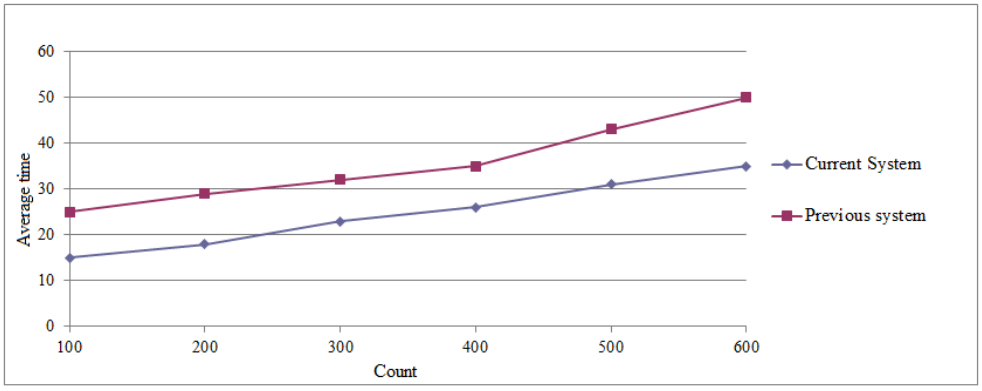

Figure 6: Graphical representation of comparison

At the point when tuples have succession numbers, taking care of copy accentuations is really simpler. The system allots the same grouping number to all copies of a bug report from previous record or SVM dataset. At the point when an accentuation is a contender for submitting downstream, the system analyses the bug flow. In the event that the arrangement number is not exactly or equivalent to the last-submitted grouping number, then the accentuation is a copy and the merger drops it. This conduct is the same with respect to data checkpoints.

There are extensions to the prior work on dataflow, where time complexity is expecting to reduced, as a part of the adaptive query processing line of work, perform load-balancing for parallel flows. Both of these leave safety.

IX. CONCLUSION

Bug triage assigns an appropriate developer to fix allocated bug for improving accuracy using data reduction techniques which is instance selection and feature selection. This work provides a pre-processing technique, which increases data quality and diminish data scale. Proposed system improves result of data reduction to organize a high quality bug report data set.
The bug triage system can merge with bug tracking system in future. It can also use the log repository in the bug triage system.

ACKNOWLEDGEMENT

With all respect and gratefulness, I would like to thanks all people who have helped us directly or indirectly. I am grateful to my guide, Prof. Swapnaja Suryawanshi, for his guidance and support. I wish to express my sincere thanks to the Head of department, Prof. Ram Joshi and our ME coordinator Prof. Manjusha Tatiya also grateful thanks to the departmental staff members for their support.

REFERENCES

[1] Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindong Wu, Towards Effective Bug Triage with Software Data Reduction Techniques, IEEE transactions on knowledge and data engineering, vol. 27, no. 1, january 2015.
[2] J. Anvik, L. Hiew, and G. C. Murphy, Who should fix this bug?, in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361370.
[3] D. Cubranic and G. C. Murphy, Automatic bug triage using text categorization, in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 9297.
[7] G. Jeong, S. Kim, and T. Zimmermann, Improving bug triage with tossing graphs, in Proc. Joint Meeting 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug. 2009, pp. 111120.
[8] J. Xuan, H. Jiang, Z. Ren, and W. Zou, Developer prioritization in bug repositories, in Proc. 34th Int. Conf. Softw. Eng., 2012, pp. 2535.
[9] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst, Finding bugs in web applications using dynamic test generation and explicit-state model checking, IEEE Softw., vol. 36, no. 4, pp. 474494, Jul./Aug. 2010.
[11] R. J. Sandusky, L. Gasser, and G. Ripoche, Bug report networks: Varieties, strategies, and impacts in an F/OSS development community, in Proc. 1st Intl. Workshop Mining Softw. Repositories, May 2004, pp. 8084.
[13] R. J. Sandusky and L. Gasser, Negotiation and the coordination of information and activity in distributed software problem management, in GROUP 05, pages 187196, 2005.
[14] J. Anvik, Automating bug report assignment, Proc. Intl. Conf. Software Engineering (ICSE 06), ACM, May 2006, pp. 937- 940.
[16] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, What makes a good bug report?, IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618643, Oct. 2010.
[17] S. Just, R. Premraj, and T. Zimmermann, Towards the next generation of bug tracking systems, in VL/HCC, pages 8285, 2008.
[18] J. W. Park, M. W. Lee, J. Kim, S. W. Hwang, and S. Kim, Costriage: A cost-aware triage algorithm for bug reporting systems, in Proc. 25th Conf. Artif. Intell., Aug. 2011, pp. 139144.
[19] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, Predicting the severity of a reported bug, in Proc. 7th IEEE Working Conf. Mining Softw. Repositories, May 2010, pp. 110.

AUTHORS BIOGRAPHY

Jagruti R. Patil received the Bachelor degree in BE (Computer) from the University of North Maharashtra University, Jalgaon, in 2014 and appearing Master degree ME (Computer) from the Savitribai Phule Pune University, Pune. Her research interests include data mining and information theory.

Swapnaja Suryawanshi, Assistant Professor at Indira Collage of Engineering & Management, Savitribai Phule Pune University, Pune.