IMPLEMENTATION OF REDUCING FEATURES TO IMPROVE CODE CHANGE BASED BUG PREDICTION BY USING COS-TRIAGE ALGORITHM

Veena Jadhav[1], Vandana Gaikwad[2], Netra Patil[3]

[1]*Student, Computer Engineering Department, BVDUCOE Pune-43, Maharashtra, India*
[2]*Assistant Professor, Computer Engineering Department, BVDUCOE Pune-43, Maharashtra, India*
[3]*Assistant Professor, Computer Engineering Department, BVDUCOE Pune-43, Maharashtra, India*

Abstract

Today, we are getting plenty of bugs in the software because of variations in the software and hardware technologies. Bugs are nothing but Software faults, existing a severe challenge for system reliability and dependability. To identify the bugs from the software bug prediction is convenient approach. To visualize the presence of a bug in a source code file, recently, Machine learning classifiers approach is developed. Because of a huge number of machine learned features current classifier-based bug prediction have two major problems i) inadequate precision for practical usage ii) measured prediction time. In this paper we used two techniques first, cos-triage algorithm which have a go to enhance the accuracy and also lower the price of bug prediction and second, feature selection methods which eliminate less significant features. Reducing features get better the quality of knowledge extracted and also boost the speed of computation.

***Keywords**: Efficiency, Bug Prediction, Classification, Feature Selection, Accuracy.*

--***--

1. INTRODUCTION

Classifiers such as SVM and navies byes are accomplished on previous software project information, also it can be expended to estimate the presence of a bug in an specific record in software, as complete in previous work[2].

To fix bug cos-triage algorithm is used. Afterward that data is added in historical data or in log record. Afterward, classifier is accomplished on data originate in previous log record. Then, it can be expended to categorize a renewed variation is either buggy or clean.

Now days, The Eclipse IDE contains a prototype displaying server which is used to compute bug predictions [4]. It can aslo offer accurate predictions. Bug prediction system must offer a lesser number of incorrect variations that are predicted to be pram but which are very new, if software developers are to belief [3]. Suppose, if enormous numbers of improved variations are incorrectly estimated to be pram, engineers won’t have belief in the bug prediction.

Previous bug prediction methodology and identical work offered by scientist which employ the removal of “features” from the prior of variations created to a software system[2]. These features comprise completely distributed by blank space in the code and that all were included or rejected in a variation. Henceforth, to prepare the SVM classifier entirely variables, comment words, numerical operatives, name of methods, and development language keywords are used as features which is existing in this paper.

Cost of large feature set is tremendously high. Because of multiple interactions and noise classifiers cannot handle such an enormous feature set. As well as number of features increases time also increases and expanding to a number of seconds per categorization for tens of thousands of features, and minutes for enormous project data accounts. Henceforth, this will influences the availability of a bug prediction scheme.

There are several classification approaches could be working. But in this paper we used cos-triage algorithm and SVM.

1) This paper contributes three aspects:
2) Study of various feature selection methods to categorize bugs in software program variations.
3) Use of cos-triage algorithm to increase accuracy and reduce cost of bug prediction.

The rest of this paper is prepared as follows: In Section 2 primary steps involved in performing change classification is presented. Also this section discuss about feature selection techniques in more detail. Section 3 discuss prior work. Section 4 contains system overview for proposed system. Finally a conclusion is made in section 5.

2. CHANGE CLASSIFICATION PROCESS

Following steps are concerned in executing Various categorization on distinct project. Creating a Corpus:
1) Change deltas are mining from the log records of a project, as kept in its SCM repository.

2) For each file bug fix variations are known by observing keywords in SCM variation log messages.
3) At the record level without bug and buggy variations are known by finding reverse in the review log record from bug resolve.
4) Features contain both buggy and clean and that mined from all variations. In Complete source code all expressions are considered as features. Features are nothing but the lines fitted in each variation, meta-data such as author, time etc. we can calculate complexity metrics at this stage.
5) In this paper to compute a reduced set of features combination of wrapper and filter approaches are used. In this paper four filter based rankers are used. i) Relief-F. ii) Chi-Squared iii) Gain Ratio iv) Significance. The wrapper methods are depends on the SVM classifiers.
6) Classification model is accomplished by using reduced set.
7) Skilled classifier is set to use. Whether a original variation is more analogous to a buggy variation or a clean variation is proven by classifier.

2.1 Detection of Buggy and Clean Changes

For bug prediction training data is fixed and used. Mining variation log records is used to determine bug introducing variations and to identify bug fixes. First is Examining for keywords in old revision such as “Set” “Bug” or other keywords possible to develop in a bug repair.second, Examining for another situations to bug.

2.2 Feature Extraction

With the help of support vector machine algorithm, using buggy and clean variations a classification model must be accomplished which is used to consolidate software variations.

The entire word in the source code programme distributed with the help of blank space or a semicolon which is used as a article .Example of that is method name, keyword, variable name, comment word, function name and operator.

2.3 Feature Selection Techniques

Huge feature groups need extended working out and estimation periods, also need lot of memory these are the requirement to execute classification.
Feature selection is common solution to this problem In which only the subcategory of structures that are generally suitable for creating grouping outcomes are truly used.

2.4 Feature Selection Process

A repetitive process of choosing increasingly fewer important group of features is finished by using Filter and wrapper approaches.

This process starts by splitting the primary feature set in partial which reduces processing requirements and memory for the remaining process.

A classification model is accomplished by using the reduced feature set. Whether a new variation is more associated to a buggy variation or a clean variation is decided by classifier.

3. RELATED WORK

A model proposed by scientist Khoshgoftaar and Allen to group modules corresponding to software quality factors such as upcoming defect intensity using different phases of multi-regression[5][6][7]. Ostrand et al. discovered two model i) The top 20% of problematic records in a project [11] using upcoming defect predictors ii) A linear regression model

Totally Ordered Program Units could be transformed into a half ordered suite list, e.g. by giving the topmost N% of units as offered by Ostrand et al. Hassan and A caching algorithm to compute the set of fault-prone modules, called the topmost-10 list offered by halt [9]. The bug cache algorithm to envisage upcoming faults built on preceding fault areas offered by Kim et al.

Fault sessions of the Mozilla project through numerous releases offered by Gyimothy et al. [11]. With the help of decision trees and neural networks that use object-oriented metrics as features. Six different feature selection techniques when using the Naive Bayes and the C4.5 classifier considered by Hall and Holmes [12] .There are about 100 features in each dataset. Hall and Holmes launched many of the feature selection methods.

A general defect prediction framework which encompass a data pre-processor, feature selection methods, and machine learning algorithms elaborated by Song et al. [14]. They also consider that slight variations to data illustration can have a vast influence on the outcomes of article selection and bug prediction. Numerous feature selection algorithms to envisage buggy software modules for a hugeheritage communications software offered by Gao et al.

Numerous feature selection algorithms to envisage buggy software units for a huge legacy communications software offered by Gao et al. There are two methods used. First, filter based techniques and second, subset selection search algorithm.

4. SYSTEM OVERVIEW

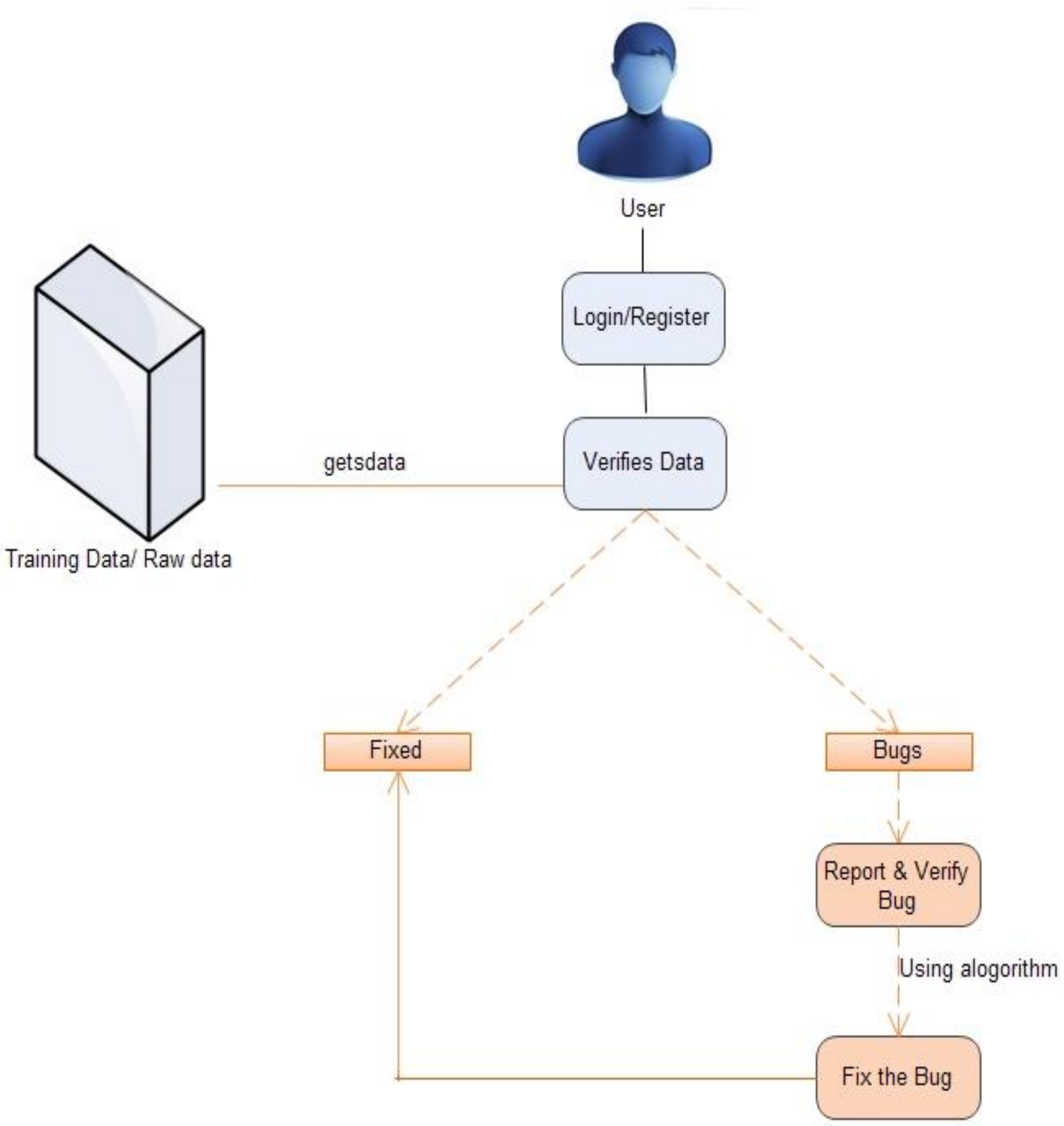
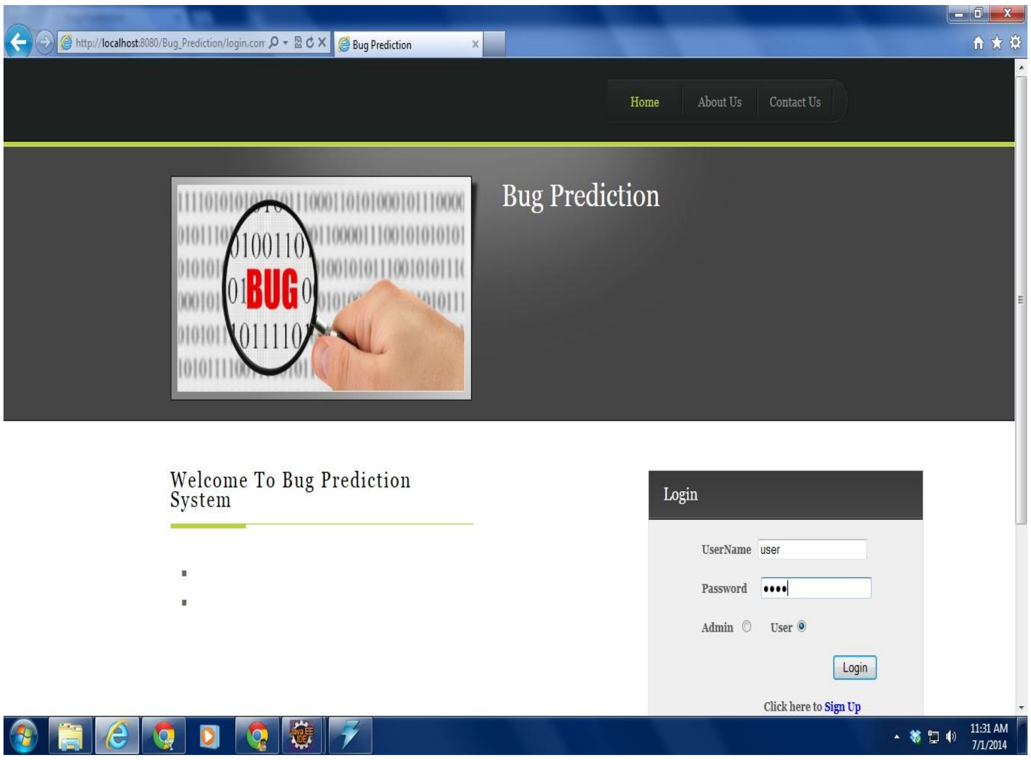


Fig -1: System Overview of bug prediction system

Above diagram fig 1. Shows general overview of the system. Originally, user login/register into the system. It also encompasses training data/raw data which retains the logs or history. Then it checks for bug. If bug present in the system verify and report bug. Then, cos-triage algorithm is applied on that bug. Cos-triage algorithm supports us for fixing bug or defects. Then at the back end that output with bug is used as input to support vector machine (SVM) classifier .SVM classifier uses different feature selection methods which is given above. After that by reducing the features we gets final output without bug. SVM classifier operate on skilled data/raw data which is kept in log record.

5. SCREENSHOTS



Welcome To Bug Prediction System

Login

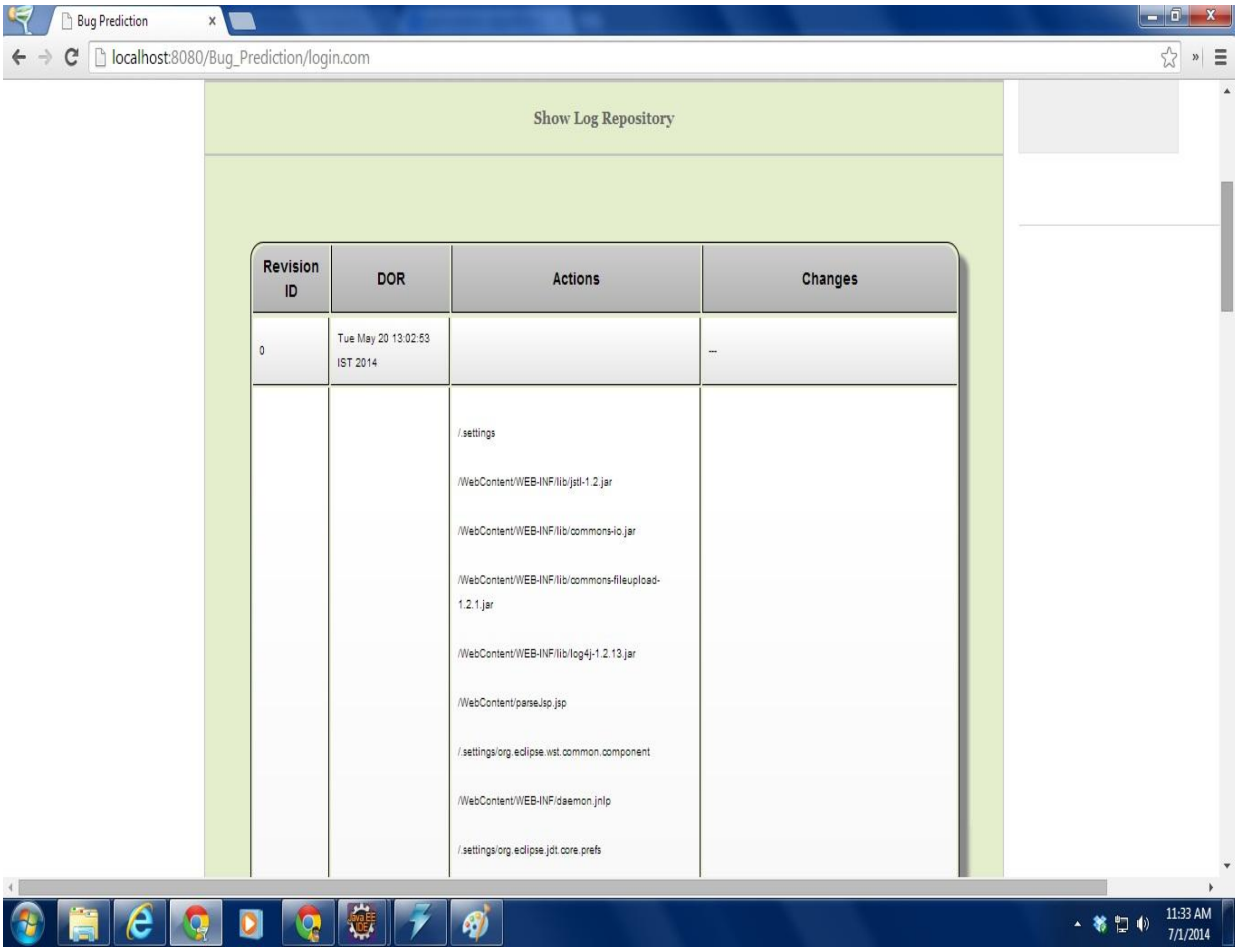
UserName user

Password ••••

Admin ○ User [✓ ●]

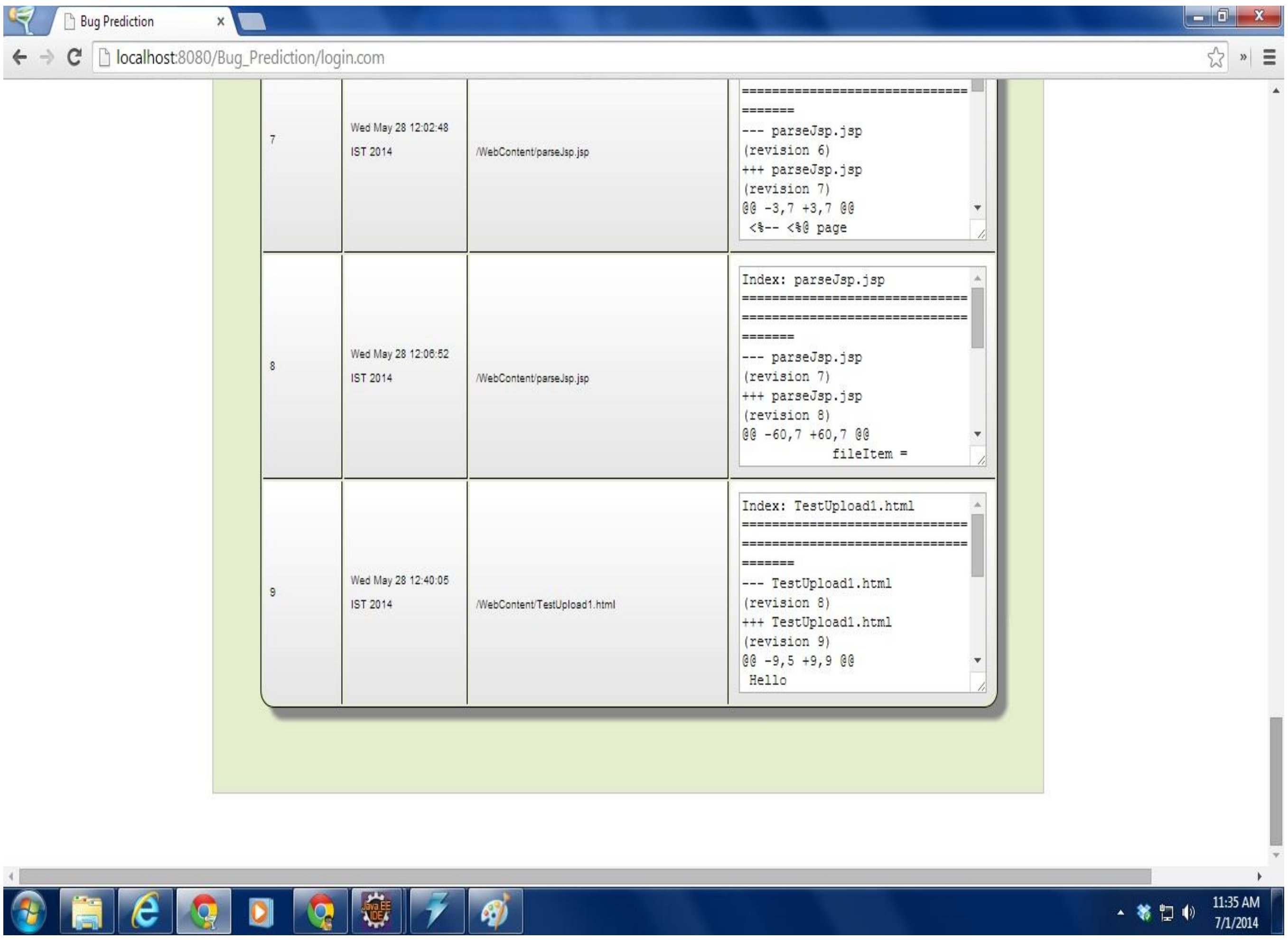
Login

Click here to Sign Up

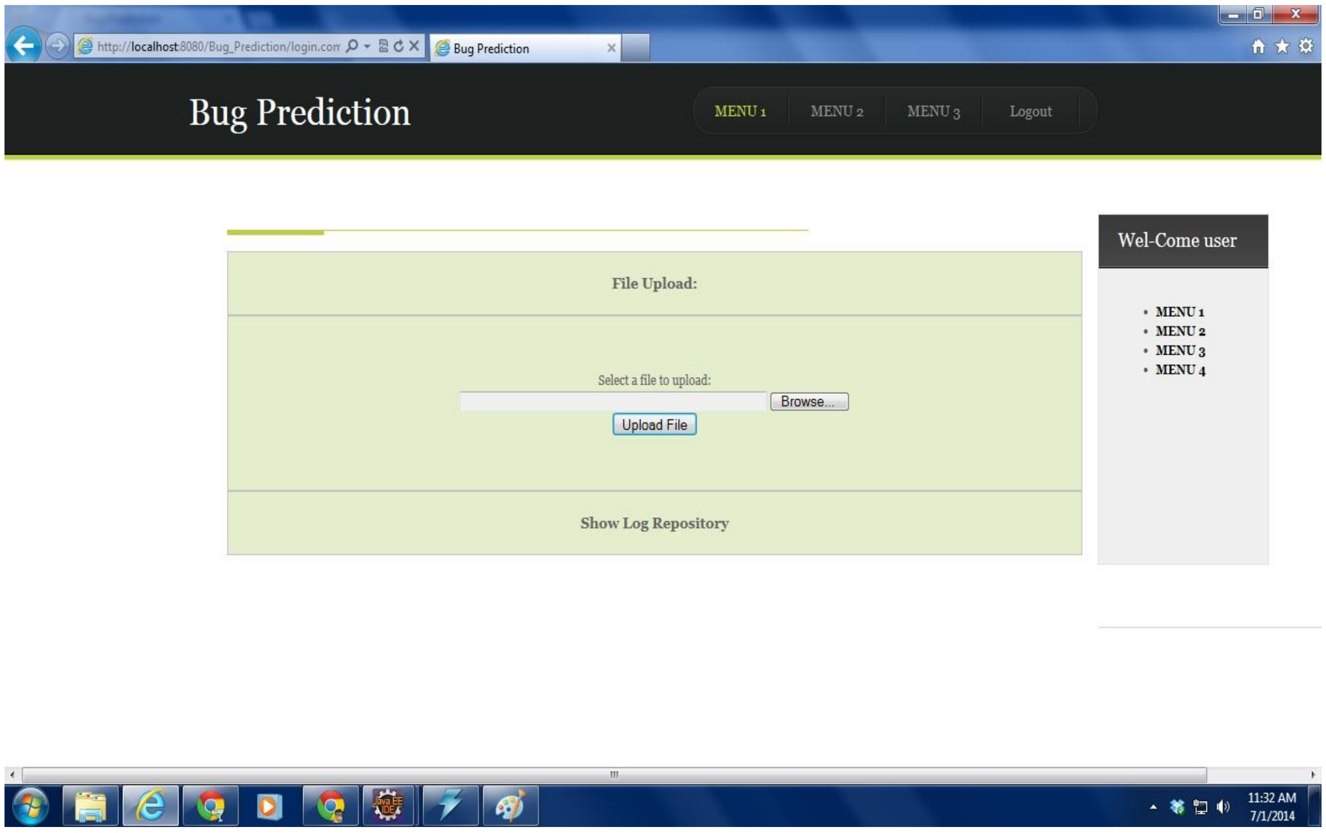


Show Log Repository

Revision ID	DOR	Actions	Changes
0	Tue May 20 13:02:53 IST 2014		---
		/.settings /WebContent/WEB-INF/lib/jstl-1.2.jar /WebContent/WEB-INF/lib/commons-io.jar /WebContent/WEB-INF/lib/commons-fileupload-1.2.1.jar /WebContent/WEB-INF/lib/log4j-1.2.13.jar /WebContent/parseJsp.jsp /.settings/org.eclipse.wst.common.component /WebContent/WEB-INF/daemon.jnlp /.settings/org.eclipse.jdt.core.prefs	



7	Wed May 28 12:02:48 IST 2014	/WebContent/parseJsp.jsp	============================== ======= --- parseJsp.jsp (revision 6) +++ parseJsp.jsp (revision 7) @@ -3,7 +3,7 @@ <%-- <%@ page
8	Wed May 28 12:06:52 IST 2014	/WebContent/parseJsp.jsp	Index: parseJsp.jsp ============================== ============================== ======= --- parseJsp.jsp (revision 7) +++ parseJsp.jsp (revision 8) @@ -60,7 +60,7 @@ fileItem =
9	Wed May 28 12:40:05 IST 2014	/WebContent/TestUpload1.html	Index: TestUpload1.html ============================== ============================== ======= --- TestUpload1.html (revision 8) +++ TestUpload1.html (revision 9) @@ -9,5 +9,9 @@ Hello



Bug Prediction

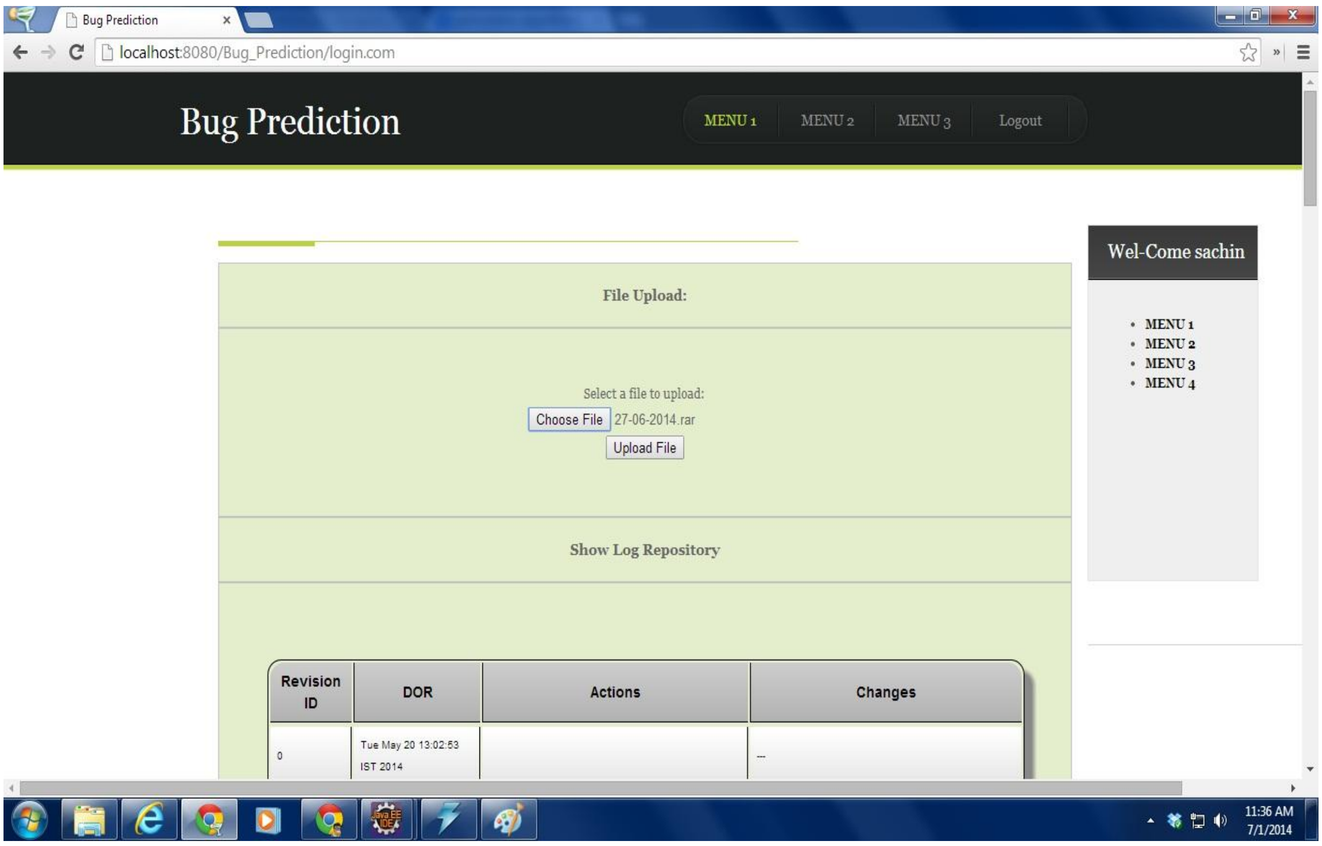
File Upload:

Select a file to upload:

Show Log Repository

Wel-Come user

- MENU 1
- MENU 2
- MENU 3
- MENU 4



Revision ID	DOR	Actions	Changes
0	Tue May 20 13:02:53 IST 2014		--

6. GRAPHICAL REPRESENTATION

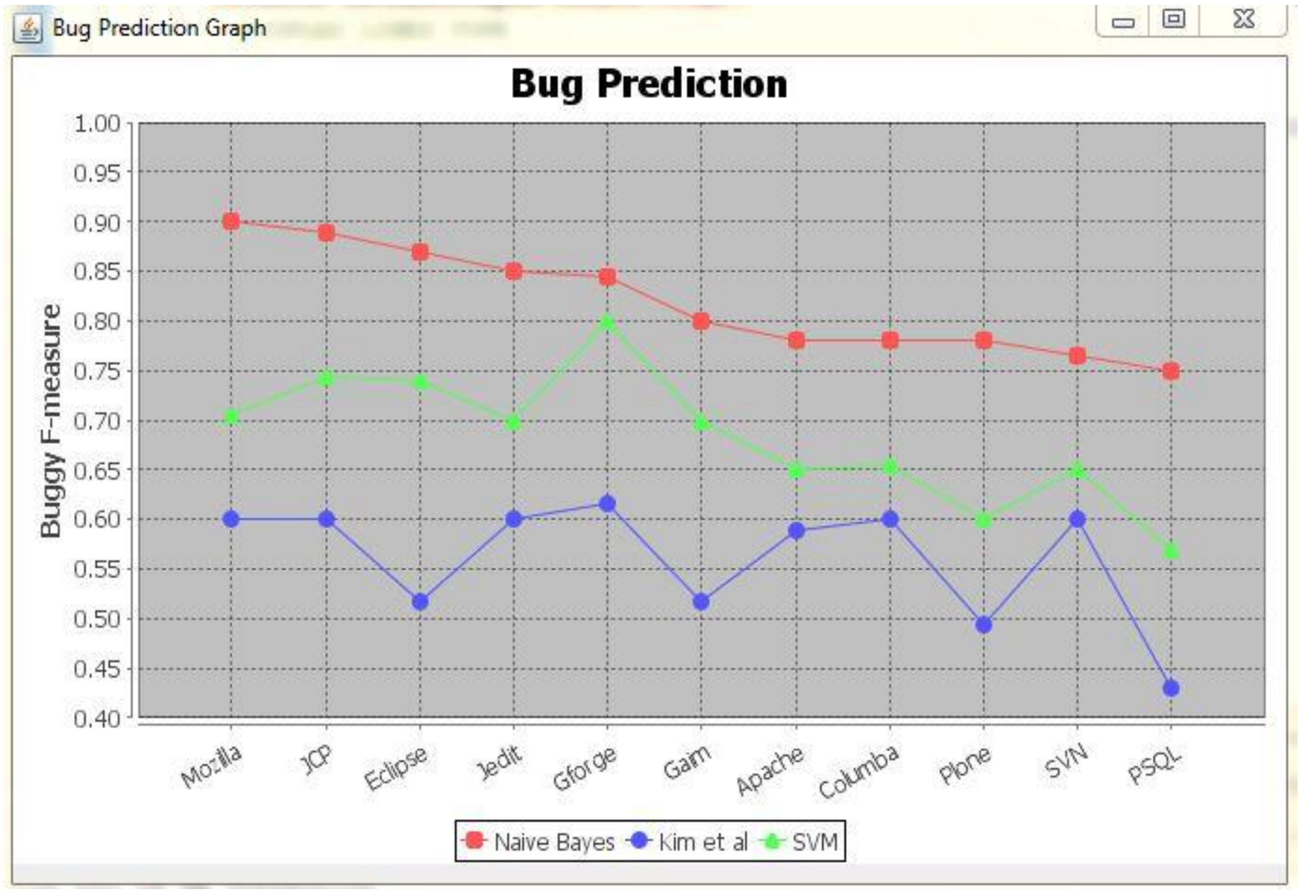


Fig 2.Graphical representation of bug prediction on projects

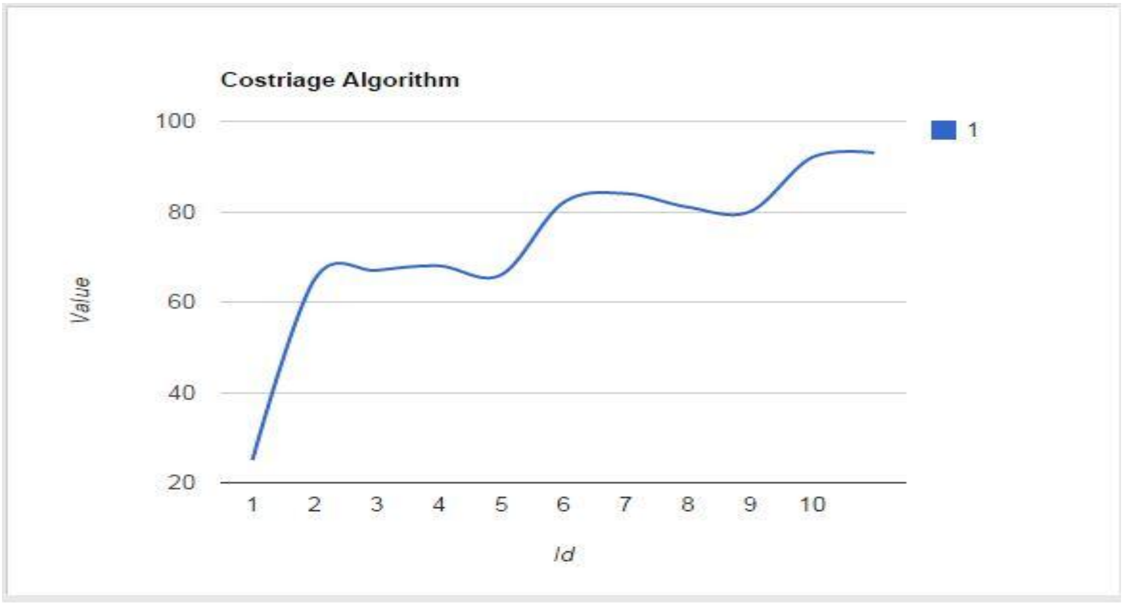


Fig.3: Cos-triage algorithm graph

7. CONCLUSION

This paper has explored algorithm called Cos-triage algorithm which helps to reduce cost and increase accuracy of bug fixing or bug prediction. In this paper we used feature selection method which reduces the quantity of structures used by a support vector device classifier and naive byes for bug prediction.

Henceforward, the usage of classifiers with feature selection will approve fast, correct, more accurate bug predictions, if software designers have highly developed bug prediction methods rooted into their software development environment. Similarly, many algorithm will build in future which will increase accuracy of bug prediction.

REFERENCES

[1] T.L. Graves, A.F. Karr, J.S. Marron, and H. Siy, “Predicting Fault Incidence Using Software Change History,” *IEEE Trans. Software Eng.*, vol. 26, no. 7, pp. 653-661, July 2000.

[2] S. Kim, E. W. Jr., and Y. Zhang, “Classifying Software Changes: Clean or Buggy?” *IEEE Trans. Software Eng.*, vol. 34, no. 2, pp. 181–196,2008.

[3] A.Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C.Henri-Gros, A. Kamsky, S. McPeak, and D.R. Engler, “A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World,” Comm. ACM, vol. 53, no. 2, pp. 66-75, 2010.

[4] J. Madhavan and E. Whitehead Jr., “Predicting Buggy Changes Inside an Integrated Development Environment,” Proc. OOPSLA Workshop Eclipse Technology eXchange, 2007.

[5] T. Khoshgoftaar and E. Allen, “Predicting the Order of Fault-Prone Modules in Legacy Software,” *Proc. 1998 Int’l Symp. on SoftwareReliability Eng.*, pp. 344–353, 1998.

[6] R. Kumar, S. Rai, and J. Trahan, “Neural-Network Techniques for Software-Quality Evaluation,” *Reliability and Maintainability Symposium*,1998.

[7] T. Ostrand, E. Weyuker, and R. Bell, “Predicting the Location and Number of Faults in Large Software Systems,” *IEEE Trans. SoftwareEng.*, vol. 31, no. 4, pp. 340–355, 2005.

[8] A.Hassan and R. Holt, “The Top Ten List: Dynamic Fault Prediction,”*Proc. ICSM’05,* Jan 2005.

[9] R. Moser, W. Pedrycz, and G. Succi, “A Comparative Analysis ofthe Efficiency of Change Metrics and Static Code Attributes for Defect Prediction,” Proc. 30th Int’l Conf. Software Eng., pp. 181-190,2008..

[10] L.C. Briand, W.L. Melo, and J. Wu¨ st, “Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects,” IEEE Trans. Software Eng., vol. 28, no. 7, pp. 706-720,July 2002.

[11] M. Hall and G. Holmes, “Benchmarking Attribute Selection Techniques for Discrete Class Data Mining,” IEEE Trans. Knowledge and Data Eng., vol. 15, no. 6, pp. 1437-1447, Nov./Dec. 2003.

[12] J. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann,1993.

[13] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, “A General Software Defect-Proneness Prediction Framework,” IEEE Trans. Software Eng., vol. 37, no. 3, pp. 356-370, May/June 2011.

[14] K. Gao, T. Khoshgoftaar, H. Wang, and N. Seliya, “Choosing Software Metrics for Defect Prediction: An Investigation on Feature Selection Techniques,” Software: Practice and Experience,vol. 41, no. 5, pp. 579-606, 2011.

[15] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C.Henri-Gros, A. Kamsky, S. McPeak, and D.R. Engler, “A FewBillion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World,” Comm. ACM, vol. 53, no. 2, pp. 66-75, 2010

BIOGRAPHIES

Miss Veena jadhav M.tech (Computer) BVDUCOE,
Pune

Prof.Vandana Gaikwad, Assistant Professor, Computer department, BVDUCOE, Pune

Prof.Netra Patil, Assistant Professor, Computer department BVDUCOE, Pune