ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

An Approach to Efficient Software Bug Prediction using Regression Analysis and Neural Networks

Surbhi Parnerkar[1], Ati Jain[2], Vijay Birchha[3]

Student, Department of Computer Science & Engineering, SVCE, Indore (M.P.), India[1]

Assistant Professor, Department of Computer Science & Engineering, SVCE, Indore (M.P.), India[2]

Head, Department of Computer Science & Engineering, SVCE, Indore (M.P.), India[3]

ABSTRACT: In software development, early prediction of defective software modules can reduce overall time and budget and increase customer satisfaction [1], by meeting customer requirements to the utmost. To deliver reliable software, it is essential to execute a number of test cases, which is tedious and costly. Huge dimensions and noisy data make the task of testing more cumbersome. In this paper, we discuss a new model towards reliability and quality improvement of software systems by predicting fault-prone modules before the testing phase. It focuses on outlier detection and removal, followed by dimensional reduction, which aims at reducing the noise and dimensions of the data sets. Further, the system is trained to predict bugs efficiently at an earlier stage. The goal is to help testers to concentrate their testing efforts to modules which have lesser bugs.

KEYWORDS: Defect prediction, Noisy data, Dimension reduction, Outlier detection, bugs.

I. INTRODUCTION

Day by day, as the need of software is increasing, so is the need of enhancing software quality. Software is used almost everywhere. It is a challenging job for software companies to deliver reliable and quality software. A major setback to development of good quality and reliable software is the occurrence of bugs. Bugs lower the quality of software, resulting in unreliable end product, and ultimately customer dissatisfaction.
The Software Development Lifecycle describes the phases of software cycle and the order in which these phases are executed. It comprises of the following stages:

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

According to IEEE standards, a ‘bug’ is an incorrect step, instruction or data in a program. A software bug can be defined as that part of the code which would result in an error, fault or malfunctioning of the program [2].

In spite of proper planning, analysis and control, the occurrence of bugs is inevitable. Different types of bugs can be introduced in each phase of SDLC. In the requirement analysis phase, some of the requirements can be forgotten, which will lead to development of an incomplete product. A bug in the design phase would probably mean that the product does not work as intended. During the development phase, there can be regression bugs. As we proceed towards further stages of SDLC, it becomes harder to predict and remove bugs. It also becomes tough not to break existing functionality, when implementing new features.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

Software testing is done in order to find the quality of the software developed and it's deviation from the desired software, which was identified during requirements gathering and analysis phase. The intention of testers is to identify the bugs.

Besides serving as a target on how many defects to capture in system testing, defect prediction can also become an early quality indicator for any software entering the testing phase.
Testing team can use the predicted defects to plan, manage and control test execution activities.
Having defect prediction as part of the testing process allows testing team to strengthen their test strategies by adding more exploratory testing and user experience testing to ensure known defects are not escaped and re-introduced to end-users. Test engineers would be able to have better root cause analysis of the defects found.

The evolution of software engineering techniques has led to development of many bug prediction algorithms. These algorithms intend to find the bug-prone areas of the software. Some algorithms make use of statistical methods, some use code metrics; some are based on feature extraction. Evaluation of the results produced by these algorithms has shown that these algorithms have been successful in reducing the tediousness of the job of testers. Machine learning metrics of accuracy, error rate, precision, time taken, memory used, recall, F-measure etc have been used to find out the performance of these algorithms.

Defect analysis can be done by two approaches: Prediction and Classification [3]. Classification and prediction that can be used to extract models describing significant defect data classes or to predict future defect trends. Classification predicts categorical or discrete, and unordered labels, whereas prediction models predict continuous valued functions.
Of the various types of testing methods used, our work focuses on regression testing. Regression testing aims to uncover new bugs. It ensures that making any changes or removal of bugs does not lead to new bugs. Common methods of regression testing include rerunning previously completed tests and checking whether program behaviour has changed and whether previously fixed faults have re-emerged. Regression testing can be performed to test a system efficiently by systematically selecting the appropriate minimum set of tests needed to adequately cover a particular change. Regression testing has traditionally been performed by a software quality assurance team after the development team has completed its work. Regression testing can be used not only for testing the correctness of a program, but often also for tracking the quality of its output.

Regression analysis can be done to predict bugs at an early stage. Datasets can be collected from previous stages, and selected data sets can be put into regression analysis, for bug prediction.

II. LITERATURE SURVEY

Previous studies have shown that, of the overall development process 27% man hour is consumed by testing [4]. Early prediction of bugs in software has proven to be helpful to software testing and quality assurance team and has been an area of continuous study since long. Defect prediction is very important in the field of software quality and reliability. Over the past two decades, software fault prediction models have got a lot of attention from developers and researchers. Yet, there is a need for robust, easy to implement and widely applicable fault prediction methods.

Bugs can be categorized on the basis of frequency of their occurrence and severity of the bug. According to severity, bugs can be classified as catastrophic, major, minor, and those with no effect [2]. It is important to predict and remove the occurrence of major and catastrophic bugs at least.

Defect analysis is of two types: Classification and prediction. It includes finding out defect classes and predicting future defect trends.

The commonly followed defect prediction process [4, 5] is:

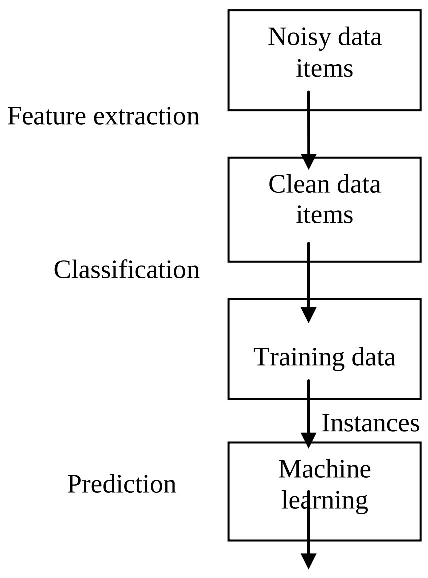

Figure 1: Steps of defect prediction process

The commonly adapted steps of defect prediction process are:

- Labeling: After gathering the defect data, the instances are labeled with buggy or clean or with the number of bugs.
- Preprocessing: Preprocessing is commonly used in machine learning. Techniques like feature extraction, data normalization and noise reduction can be used.
- Creating training sets: By combining labels and features of instances, training sets can be created, for training the model used for prediction.
- Prediction and assessment: The prediction model can be used to predict whether the instance has bug or not.

Various defect prediction techniques like metrics, models and algorithms can be used.

Many different approaches such as genetic programming [6], neural networks [7], fuzzy logic [8], decision trees [9], Naı¨ve Bayes [10], and logistic regression [11], Support Vector Machines [12], etc have been proposed. The usage of machine learning algorithms has increased since 2005. It is still one the most popular methods for fault prediction [13]. Genetic programming can be used, with software metrics as input and the prediction of number of faults as output [6].Use of subtractive fuzzy clustering can be used for finding out the number of faults and module-order modeling can be used to find whether or not the modules are fault prone [8].

Naïve Bayes algorithm has also got high performance with respect to fault prediction [14].
It is found that various machine learning approaches [7] such as supervised, semi-supervised, and unsupervised have been used for building fault prediction models. Among these, supervised learning approach is widely used and found to be more useful for predicting fault-prone modules if sufficient amount of fault data from previous releases are available. Generally, these models use software metrics of earlier software releases and fault data collected during testing phase. The supervised learning approaches cannot build powerful models when data is limited. Therefore, the some researchers presented a semi-supervised classification approach for software fault prediction with limited fault

data. Catal and Diri [15] investigated the effects of data size, metrics, and the feature selection techniques for software fault prediction.

Decision trees can also be used to classify whether the modules are fault prone or not.

Metrics collected from earlier phases can be identified and analysed to select the potential predictors, on which regression analysis [11] can be applied, to generate mathematical equations.

Apart from using Support Vector Machine, Support Vector based fuzzy classification system (SVFCS) can also be used for defective module prediction, which is quiet suitable for object oriented software [1].

In general, software fault prediction approaches use previous software metrics and fault data to predict fault prone modules. The occurrence of an error in a module's data is marked as 1; otherwise it is marked as 0.

Of the various bug prediction techniques, hybrid techniques have given considerably good results. Hybrid Support Vector Machine (SVM) classifier can be used. Principal Component Analysis (PCA) can be used for feature extraction [16].

III. PROPOSED WORK

Bug prediction is needed to identify the different semantics and syntactical attributes analysis for identifying the error and bugs. Therefore it turns into a classification and pattern analysis problem. Additionally a lot of data is available in high dimensional attributes, thus it is needed to optimize it by enhancing classification accuracy and resource consumption optimization in terms of space and time complexity.

The goal of the work is to perform efficient bug prediction in software, so that the cumbersome task of testing will become easier. The approach will find applicability in software development companies, to make the task of testing easier and to get a better, reliable and less faulty end product. It will prove to be helpful for software testers by making the test cases better and to the end user by delivering a better product.

After the literature survey, it has been found that the major problems incurred are:

- Huge dimensions: When the data sets have huge dimensions, processing is improper and it is difficult to test it.
- Noisy data: Addition of noise to the relevant data makes the task of testing complex.
- Cost: The cost of defect correction is significantly high after software testing.

For providing a solution to these problems, it is required to have better data sets.
The huge dimensions can be reduced by dimension reduction techniques and outlier detection and removal can be done by using data mining and data clustering techniques.
To avoid application of the method time and again, the system can be trained using a neural network, and regression analysis can be performed.
The issue of cost will get resolved up to an extent by early prediction of bugs, rather than getting it accomplished at the testing phase.

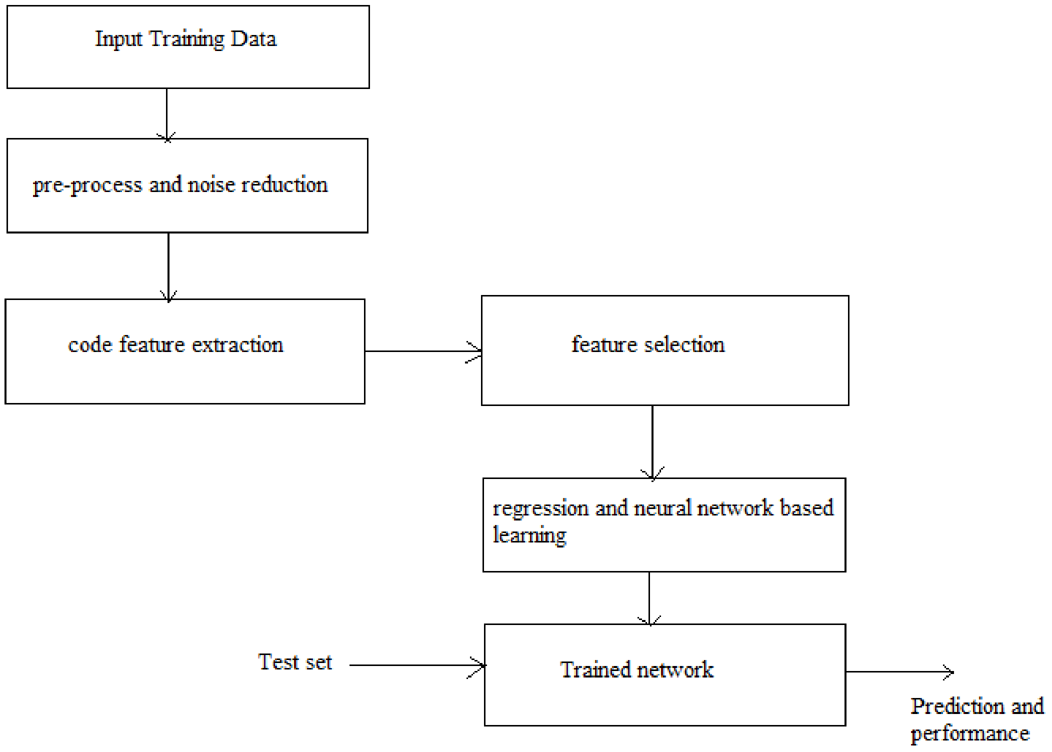
Input Training Data
pre-process and noise reduction
code feature extraction
feature selection
regression and neural network based learning
Test set
Trained network
Prediction and performance

Figure 2: Steps of the proposed approach

The proposed model helps in predicting the fault in the code changes. The main components of the presented system, mentioned in the flow diagram are:

- Input training set: That is a simple user interface which accepts the raw training set. The training set may contains noisy data, and irrelevant instances.
- Pre-process data set: In this phase the training data is transformed and normalized for finding the more relevant attributes and instances.
- Code feature extraction: In this phase using neighbour component analysis method significant features are extracted.
- Feature selection: Initially the calculated features are found in huge quantity therefore using K-PCA algorithm the dimensionality of the data is reduced for finding more appropriate features from the training samples.
- Regression and neural network: Neural network is an essential data mining tool which can be implementable with the verity of applications such as recognition, pattern detection and other similar data models. But that suffers from the slow learning rate. Therefore the weight adjustment and initialization phase is modified using the regression analysis concept. That may improve the learning capability of the system.

IV. CONCLUSION

After the study, it has been observed that early prediction of bugs leads to better resource planning, test planning and ultimately to a reliable and less faulty end product. Thus, the use of such kind of techniques will help the testers and will lead to development of a less faulty product. With the help of regression analysis approach, the occurrence of new faults and re-occurrence of existing faults will not take place. By training the neural network, the learning capacity of the system will be improved, which is a futuristic approach for bug prediction.

REFERENCES

1. Bharavi Mishra, K.K. Shukla, Defect Prediction for Object Oriented Software using Support Vector based Fuzzy Classification Model, International Journal of Computer Applications (0975 – 8887) Volume 60– No.15, December 2012.
2. Vipindeep V, Pankaj Jalote, List of Common Bugs and Programming Practices to avoid them, March 30, 2005
3. Mrinal Singh Rawat, Sanjay Kumar Dubey, Software Defect Prediction Models for Quality Improvement: A Literature Study, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, September 2012
4. Ajeet Kumar Pandey and N. K. Goyal , " Predicting Fault-prone Software Module Using Data Mining Technique and Fuzzy Logic", International Journal of Computer and Communication Technology (Special Issue) Vol. 2, Issue 2-4, pp. 56-63 (2010).
5. Jaechang Nam, Survey on Software Defect Prediction
6. Matthew Evett, Taghi Khoshgoftar, Pei-der Chien, Edward Allen, GP-based software quality prediction
7. M.M. Thwin and T. Quah, Application of neural networks for software quality prediction using object-oriented metrics, In the proceedings of the 19th International Conference on Software Maintenance, Amsterdam, The Netherlands, 2003, pp. 113-122.
8. Yuan, X., Khoshgoftaar, T.M., Allen, E.B., Ganesan, K.," An Application of Fuzzy Clustering to Software Quality Prediction.2000 In: Proceedings of The 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology
9. T.M. Khoshgoftaar and N. Seliya, Software quality classification modeling using the SPRINT decision tree algorithm, In the proceedings of the 4th IEEE International Conference on Tools with Artificial Intelligence, Washington, DC, 2002, pp. 365-374.
10. T. Menzies, J. Greenwald and A. Frank, Data Mining Static Code Attributes to Learn Defect Predictors, IEEE Transactions on Software Engineering, 33 (2007) 2-13.
11. Muhammad Dhiauddin Mohamed Suffian, Suhaimi Ibrahim, A Prediction Model for System Testing Defects using Regression Analysis, International Journal of Soft Computing And Software Engineering (JSCSE) e-ISSN: 2251-7545, Vol.2,_o.7, 2012
12. K.O. Elish and M.O. Elish, Predicting defect-prone software modules using support vector machines, Journal of Systems and Software, 81 (2008) 649-660.
13. Cagatay Catal, Ugur Sevim , Banu Diri , Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm, Expert Systems with Applications 38 (2011) 2347–2353
14. Cagatay Catal, Banu Diri, A systematic review of software fault prediction studies, Expert Systems with Applications 36 (2009) 7346–7354
15. C. Catal and B. Diri, Investigating The Effect Of Dataset Size, Metrics Set, and Feature Selection Techniques on Software Fault Prediction Problem, Information Sciences, 179 (2009) 1040-1058.
16. A. Shanthini, G.Vinodhini and RM. Chandrasekaran, Effect of Principle Component Analysis and Support Vector Machine in Software Fault Prediction, International Journal of Computer Trends and Technology (IJCTT) – volume 7 number 3–Jan 2014

BIOGRAPHY

Surbhi Parnerkar completed Bachelor of Engineering degree in 2012 from SVCE, Indore, MP, India. Her research interests are Software Engineering, Machine Learning and Information Security.

Ati Jain completed Master of Technology in Information Technology in 2011 from MIT, Ujjain, MP, India. Her research areas include Software Engineering and Object Oriented Analysis and Design.

Vijay Birchha completed Master of Engineering in Software Engineering from IET-DAVV, Indore, MP, India. His research interests are Software Engineering, Soft Computing and Cloud Computing.