An Review on Software Defect Prediction based on Various Data Mining

Rupesh Kumar Sahu[1], Rahul Kumar Chawda[2]
[1]Student of MCA, [2]Assistant Professor
Department of Computer Science, Kalinga University, Raipur.

Abstract - There has been rapid growth of software development. Due to various causes, the software comes with many defects. In Software development process, testing of software is the main phase which reduces the defects of the software. If a developer or a tester can predict the software defects properly then, it reduces the cost, time and eff ort.

Keyword - Software defect prediction, classification Algorithm, Confusion matrix

I. INTRODUCTION

Software Defect Prediction There has been a huge growth in the demand for software quality during recent ages. As a consequence, issues are related to testing, becoming increasingly critical. The ability to measure software defect can be extremely important for minimizing cost and improving the overall eff ectiveness of the testing process. The major amount of faults in a software system is found in a few of its components. Although there is variety in the definition of software quality, it is truly accepted that a project with many defects lacks the quality of the software. Knowing the causes of possible defects as well as identifying general software process areas that may need attention from the initialization of a project could save money, time and working eff ort. The possibility of early estimating the probable faultiness of software could help on planning, controlling and executing software development activities. A low cost method for defect analysis is learning from past mistakes to prevent future ones. Today, there exist several data sets that could be mined in order to discover useful knowledge regarding defects.
Using this knowledge one should ideally be able to:–
a. Identify potential fault-prone software.
b. Estimate the distinct number of faults, and
c. Discover the possible causes of faults..

Motivation - Diff erent data mining methods have been proposed for defect analysis in the past, but few of them manage to deal successfully with all of the above issues. Regression models estimates are difficult to interpret and also provide the exact number of faults which is too risky, especially in the beginning of a project when too little information is available. On the other hand classification models that predict possible faultiness can be specific, but not so much useful to give clue about the actual number of faults. Many researchers used many techniques with diff erent dataset that predict faultiness. But there are so many classification rule algorithms that can be eff ective to predict faultiness. All these issues motivates to our research in these field of software fault/defect prediction.

II. LITERATURE REVIEW

Poor software quality may be manifested through severe software defects, or software maintenance may be costly due to many defects requiring extensive eff ort to correct. Last, we explore relevant research methods for this study. The following digital sources were consulted: ACM Digital Library, IEEE Xplore, and Science Direct.

Data Mining for software Engineering - To improve the software productivity and quality, software engineers are applying data mining algorithms to various SE tasks. Many algorithms can help engineers figure out how to invoke API methods provided by a complex library or framework with insufficient documentation. In terms of maintenance, such type of data mining algorithms can assist in determining what code locations must be changed when another code location is changed. Software engineers can also use data mining algorithms to hunt for potential bugs that can cause future in-field failures as well as identify buggy lines of code (LOC) responsible for already-known failures. The second and third columns of Table 2.1 list several example data mining algorithms and the SE tasks to which engineers apply them

Table 1: Example software engineering data, Mining algorithm, SE task

SE Data	Mining algo.	SE Tasks
Sequences: execution/ static traces, co-changes	Frequent itemset/ sequence/ partial-order mining, sequence matching/ clustering/ classification	Programming, maintenance, bug detection, debugging
Graphs: dynamic/ static call graphs, program dependence graphs	Frequent subgraph mining, graph matching/ clustering/ classification	Bug detection, debugging
Text: bug reports, e-mails, code comments, documentation	Text matching/ clustering/ classification	Maintenance, bug detection, debugging

Binary classification - In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. Binary or binomial classification is the task of classifying the members of a given set of objects into two groups on the basis of whether they have some property or not.

Data Classification is two-step process. In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm is builds the classifier by analyzing or "learning form" a training set made up of database tuples and their associated class labels. In the second step the model is used for classification. Therefore, a test set is used, make up of test tupples and their associated class labels.

A classification rule takes the form X=> C, where X is a set of data items, and C is the class (label) and a predetermined target. With such a rule, a transaction or data record t in a given database could be classified into class C if t contains X.

Bayesian Classification - The Naive Bayesian classifier is based on Bayes theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Rule-Based Classification - Rules are a good way of representing information or bits of knowledge. A rule-based classifier uses a set of IF-THEN rules for classification. An IF-THEN rule is an expression of the form

Logistic Regression - In statistics, logistic regression or logit regression is a type of regression analysis used for predicting the outcome of a categorical dependent variable (a dependent variable that can take on a limited number of values, whose magnitudes are not meaningful but whose ordering of magnitudes may or may not be meaningful) based on one or more predictor variables.

III. RELATED WORKS

Regression via classification - In 2006, Bibi, Tsoumakas, Stamelos, Vlahavas, apply a machine learning approach to the problem of estimating the number of defects called Regression via Classification (RvC) The whole process of Regression via Classification (RvC) comprises two important stages:

a. The discretization of the numeric target variable in order to learn a classification model,
b. the reverse process of transforming the class output of the model into a numeric prediction.

Static Code Attribute - Menzies, Greenwald, and Frank (MGF) published a study in this journal in 2007 in which they compared the performance of two machine learning techniques (Rule Induction and Naive Bayes) to predict software components containing defects. To do this, they used the NASA MDP repository, which, at the time of their research, contained 10 separate data sets.

ANN - In 2007, Iker Gondra used a machine learning methods for defect prediction. He used Artificial neural network as a machine learner.

Embedded software defect prediction - In 2007, Oral and Bener used Multilayer Perception (MLP), NB, VFI (Voting Feature Intervals) for Embedded software defect prediction. there they used only 7data sets for evaluation.

Association rule classification - In 2011 Baojun, Karel used classification based association rule named CBA2 for software defect prediction.In these research they used assocition rule for clssafication. and they compare with other classification rules such as C4.5 and Ripper.

Defect-proneness Prediction framework

In 2011, Song, Jia, Ying, and Liu propased a general framework for software defect-pronness prediction. in this research they use M*N cross validation with the dataset(NASA, Softlab Dataset) for learning process. and they used 3 classification algorithms(Naive baysed, OneR, J48). and they compared with MGF framework. In 2010 a research has been done by Chen, Sen, Du Ge, on software defect prediction using data mining. In this research they used probabilistic Relational model and Baysean Network.

IV. CONCLUSION

This paper reviewed the current state of software defect management, software defect prediction models and data mining technology briefly. Then proposed an ideal software defect management and prediction system, researched and analyzed several software defect prediction methods based on data mining techniques and specific models (NB, Logistic, PART, J48G)

V. REFERENCES

[1]. Tao Xie, Suresh Thummalapenta, David Lo, and Chao Liu. Data mining for software engineering. Computer, 42(8):55–62, 2009.

[2]. Qinbao Song, Zihan Jia, Martin Shepperd, Shi Ying, and Jin Liu. A general software defect-proneness prediction framework. Software Engineering, IEEE Transactions on, 37(3):356–370, 2011.

[3]. Ma Baojun, Karel Dejaeger, Jan Vanthienen, and Bart Baesens. Software defect prediction based on association rule classification. Available at SSRN 1785381, 2011.

[4]. S Bibi, G Tsoumakas, I Stamelos, and I Vlahavas. Software defect prediction using regression via classification. In IEEE International Conference on, pages 330–336, 2006.

[5]. Tim Menzies, Jeremy Greenwald, and Art Frank. Data mining static code attributes to learn defect predictors. Software Engineering, IEEE Transactions on, 33(1):2–13, 2007.

[6]. Iker Gondra. Applying machine learning to software fault-proneness prediction. Journal of Systems and Software, 81(2):186–195, 2008.

[7]. Atac¸ Deniz Oral and Ay¸se Ba¸sar Bener. Defect prediction for embedded software. In Computer and information sciences, 2007. iscis 2007. 22nd international symposium on, pages 1–6. IEEE, 2007.

[8]. Yuan Chen, Xiang-heng Shen, Peng Du, and Bing Ge. Research on software defect prediction based on data mining. In Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on, volume 1, pages 563–567. IEEE, 2010.

[9]. Martin Shepperd, Qinbao Song, Zhongbin Sun, and Carolyn Mair. Data quality: Some comments on the nasa software defect data sets. 2013.

[10]. Stefan Lessmann, Bart Baesens, Christophe Mues, and Swantje Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. Software Engineering, IEEE Transactions on, 34(4):485–496, 2008.

[11]. Yue Jiang, Bojan Cukic, and Tim Menzies. Fault prediction using early lifecycle data. In Software Reliability, 2007. ISSRE'07. The 18th IEEE International Symposium on, pages 237–246. IEEE, 2007.

[12]. Yue Jiang, Bojan Cuki, Tim Menzies, and Nick Bartlow. Comparing design and code metrics for software quality prediction. In Proceedings of the 4th international workshop on Predictor models in software engineering, pages 11–18. ACM, 2008.

[13]. Hongyu Zhang, Xiuzhen Zhang, and Ming Gu. Predicting defective software components from code complexity measures. In Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on, pages 93–96. IEEE, 2007.

[14]. Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1):20–29, 2004.

[15]. Charles E Metz, Benjamin A Herman, and Jong-Her Shen. Maximum likelihood estimation of receiver operating characteristic (roc) curves from continuously-distributed data. Statistics in medicine, 17(9):1033–1053, 1998.

[16]. Qinbao Song, Martin Shepperd, Michelle Cartwright, and Carolyn Mair. Software defect association mining and defect correction eff ort prediction. Software Engineering, IEEE Transactions on, 32(2):69–82, 2006.