SOFTWARE DEFECT FORECASTING BASED ON CLASSIFICATION RULE MINING

Nafeesa Hamid[1], Jyoti Arora[2]
[1]*M.Tech Student, Desh Bhagat University, Mandi Gobindgarh*
[2]*Assistant Professor, Desh Bhagat University, Mandi Gobindgarh*
(E-mail: nafisa_hamid@yahoo,com)

***Abstract*—**The ability to measure software defect can be extremely important for minimizing cost and improving the overall effectiveness of the testing process. The major amount of faults in a software system are 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 effort. The possibility of early estimating the probable faultiness of software could help on planning, controlling and executing software development activities. Different 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 use full to give clue about the actual number of faults. Many researcher used many techniques with different dataset that predict faultiness. But there are so many classification rule algorithms that can be effective to predict faultiness. All these issues motivates to our research in these field of software defect prediction. In order to improve the efficiency and quality of software development, we can make use of the advantage of data mining to analysis and predict large number of defect data collected in the software development. 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)

Keywords*—*Rule Mining; Classificaion; Software defect Detection, Data Mining.

I. INTRODUCTION

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 effectiveness of the testing process. The major faults in a software system are 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 effort.

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.

Different 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 use full to give clue about the actual number of faults. Many researcher used many techniques with different dataset that predict faultiness. But there are so many classification rule algorithms that can be effective to predict faultiness. All these issues motivates to our research in these field of software falult/defect prediction.

II. RELATED WORK

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) [4].The whole process of Regression via Classification (RvC) comprises two important stages: Firstly, the discretization of the numeric target variable in order to learn a classification model, and secondly, the reverse process of transforming the class output of the model into a numeric prediction.

Menzies, Greenwald, and Frank (MGF) [5] 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 datasets.

In 2007, Iker Gondra [6]used a machine learning methods for defect prediction. He used Artificial neural network as a machine learner.

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

In 2011 Baojun, Karel [3] used classification based association rule named CBA2 for software defect prediction. In these research they used association rule for classification. and they compare with other classification rules such as C4.5 and Ripper.

In 2011, Song, Jia, Ying, and Liu proposed a general frame work for software defect-proneness prediction. in this research they use M*N cross validation with the dataset (NASA, Soft lab Dataset) for learning process. and they used 3 classification algorithms(Naive baysed, One R, J48). and they compared with MGF [5] framework. In 2010 a research has been done by Chen, Sen, Du Ge, [8] on software defect prediction using datamining. In this research they used probabilistic Relational model and Baysean Network.

III. PROPOSED WORK

A. *Overview*

In General, before building defect prediction model and using them for prediction purposes, we first need to decide which learning scheme or learning algorithm should be used to construct the model. Thus, the predictive performance of the learning scheme should be determined, especially for future data. However, this step is often neglected and so the resultant prediction model may not be Reliable. As a consequence, we use a software defect prediction framework that provides guidance to address these potential shortcomings.

The framework consists of two components:

- scheme evaluation
- defect prediction.

Figure 1 contains the details. At the scheme evaluation stage, the performances of the different learning schemes are evaluated with historical data to determine whether a certain learning scheme performs sufficiently well for prediction purposes or to select the best from a set of competing schemes.

From figure 1, we can see that the historical data are divided into two parts: a training set for building learners with the given learning schemes, and a test set for evaluating the performances of the learners. It is very important that the test data are not used in any way to build the learners.

This is a necessary condition to assess the generalization ability of a learner that is built according to a learning scheme and to further determine whether or not to apply the learning scheme or select one best scheme from the given schemes.

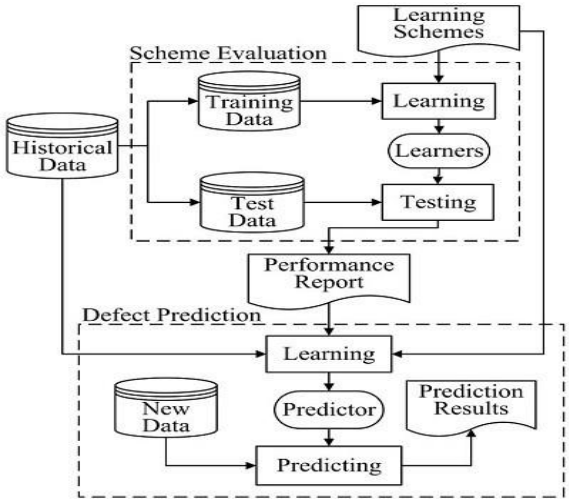


Figure 1: Proposed framework

At the defect prediction stage, according to the performance report of the first stage, a learning scheme is selected and used to build a prediction model and predict software defect. From Fig. 1, we observe that all of the historical data are used to build the predictor here. This is very different from the first stage; it is very useful for improving the generalization ability of the predictor. After the predictor is built, it can be used to predict the defect-proneness of new software components.

B. *Scheme Evaluation*

The scheme evaluation is a fundamental part of the software defect prediction framework. At this stage, different learning schemes are evaluated by building and evaluating learners with them. The first problem of scheme evaluation is how to divide historical data into training and test data. As mentioned above, the test data should be independent of the learner construction.

This is a necessary precondition to evaluate the performance of a learner for new data. Cross-validation is usually used to estimate how accurately a predictive model will perform in practice. One round of cross-validation involves partitioning a dataset into complementary subsets, performing the analysis on one subset, and validating the analysis on the other subset. To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

In our framework, an percentage split used for estimating the performance of each predictive model, that

is, each data set is first divided into 2 parts, and after that a predictor is learned on 60% instances, and then tested on the remaining 40%.

To overcome any ordering effect and to achieve reliable statistics, each holdout experiment is also repeated M times and in each repetition the data sets are randomized. So overall, M*N(N=Data sets) models are built in all during the period of evaluation; thus M*N results are obtained on each data set about the performance of the each learning scheme.

After the training-test splitting is done each round, both the training data and learning scheme(s) are used to build a learner. A learning scheme consists of a data preprocessingmethod,anattributeselectionmethod,andal earningalgorithm. Evaluation of the proposed framework is comprised of :

C. Scheme Evaluation Algoritm

Data: Historical Data Set

Result: The mean performance values

1 M=12 :No of Data Set

2 i=1;

3 **while** *i*<=*M* **do**

4 Read Historical Data Set D[i];

5 Split Data set Intances using %split;

6 Train[i]=60% of D; % Training Data;

7 Learning(Train[i],scheme);

8 Test Data=D[i]-Train[i];% Test Data;

9 Result=Test Classifier(Test[i],Learner);

10 **end**

Algorithm 1: Scheme Evaluation

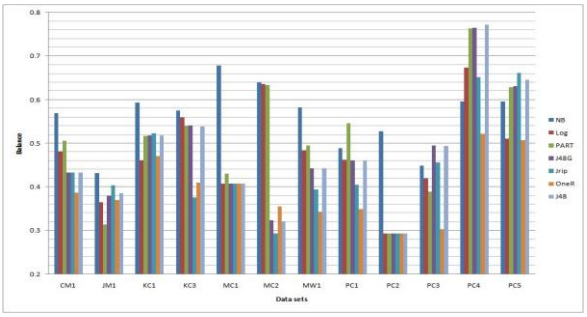
D. Defect Prediction

The defect prediction part of our framework is straightforward; it consists of predictor construction and defect prediction. During the period of the predictor construction:

1. A learning scheme is chosen according to the Performance Report.
2. A predictor is built with the selected learning scheme and the whole historical data. While evaluating a learning scheme, a learner is built with the training data and tested on the test data. Its final performance is the mean over all rounds. This reveals that the evaluation indeed covers all the data. Therefore, as we use all of the historical data to build the predictor, it is expected that the constructed predictor has stronger generalization ability. After the predictor is built, new data are preprocessed in same way as historical data, then the constructed predictor can be used to predict software defect with preprocessed new data.

IV. RESULTS AND DISCUSSION

Depending on Accuracy, Sensitivity, Specificity, Balance performance we choosen 6 Algoritms: Naïve Bayes Simple, Logistic, J Rip, PART, J48 and J48Graft



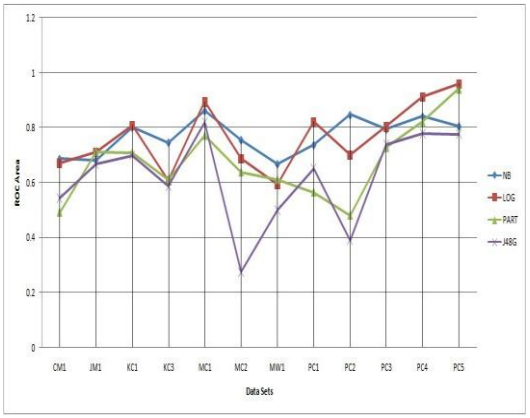


Figure 2. Performance Comparison of algorithms

Figure 3: ROC Area

V. CONCLUSION

In our research work we have attempted to solve the Software defect prediction problem through different Data mining (Classification) algorithms. In our research NB and Logistic algorithm gives the overall better performance for defect prediction. PART and J48 gives better performance than OneR and JRip.

From these results, we see that a data preprocessor/attribute selector can play different roles with different learning algorithms for different datasets and that no learning scheme dominates, i.e., always

outperforms the others for all data sets. This means we should choose different learning schemes for different datasets, and consequently, the evaluation and decision process is important.

In order to improve the efficiency and quality of software development, we can make use of the advantage of data mining to analysis and predict large number of defect data collected in the software development. 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)

REFERENCES

[1] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

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

[3] 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.

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

[5] SBibi,GTsoumakas,IStamelos,andIVlahavas.Software defect prediction using regression via classification. In IEEE International Conference on, pages 330–336,2006.

[6] 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.

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

[8] Ata¸cDenizOralandAy¸seBa¸sarBener.Defect prediction for embedded software.In Computer and information sciences, 2007. iscis 2007. 22nd international symposium on, pages 1–6. IEEE,2007.

[9] Yuan Chen, Xiangng 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 Conferenceon,volume1,pages563–567.IEEE,2010.

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

[11] [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 Transactionson,34(4):485–496,2008.

[12] 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.

[13] 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 onPredictor models in software engineering, pages11–18.ACM,2008.

[14] 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 Symposiumon,pages93–96.IEEE,2007.

[15] 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.

[16] 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.

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

[18] Norman E.Fentonand Martin Neil. Acritique of software defect prediction models.Software Engineering, IEEE Transactions on, 25(5):675–689,1999.

[19] Naeem Seliya and Taghi MKhoshgoftaar. Software qualityes timation with limited fault data: a semi-supervised learning perspective. Software Quality Journal, 15(3):327–344,2007.

[20] Frank Padberg,Thomas Ragg,and Ralf Schoknecht.Using machine learning for estimating the defect content after an inspection. Software Engineering, IEEE Transactions on, 30(1):17–28, 2004.

[21] Venkata UB Challagulla, Farokh B Bastani, I-Ling Yen, and Raymond A Paul. Empirical assessment of machine learning based software defect prediction techniques. In Object-Oriented Real-Time Dependable Systems,2005. WORDS2005.10th IEEE International Workshopon, pages 263–270. IEEE,2005.

[22] Norman Fenton, Paul Krause, and Martin Neil. A probabilistic model for software defect prediction. IEEE Trans Software Eng,2001.

[23] RaimundMoser,WitoldPedrycz,andGiancarloSucci.Acomparative analysisoftheefficiency of change metrics and static code attributes for defect prediction. In Software Engineering, 2008.

[24]

[25]

[26]

[27]

[28]

28]