RESEARCH ARTICLE OPEN ACCESS

Bug Prediction using Local and Global Models

Sarika Chopde*

*(Department of Computer Engineering, D Y PATIL College of Engineering, Ambi,Pune
Email: sarika.pachlore@gmail.com)

--************************----------------------------------

Abstract:

Bug prediction is an important topic in software Quality chapter which objects to identify bug-inducing variations made in the software. Performance of the defect predictors get affected by some defect data sets shown recently in some studies. Local models expand the performance of prediction models. Previously the stress was always given on module-level defect prediction.. The trialstudies three assessment scenarios. It was found that Local models are much better than global models across different cross validation scenarios.

Keywords ***—Software bug prediction; faults prediction; prediction model; machine learning; Just in Time – SDP; Regression***

--************************----------------------------------

I. INTRODUCTION

Software has become an fundamental part of humans in todaysworld .So it becomes necessary that the software should be defect free .Otherwise it can incur losses for users and business corporations NIST of United states of America have estimated losses of $60 billion per year to the economy [1]. It becomes extremely important to detect the defects early in software through different authentication and justificationprocess . Software bug prediction is an essential activity in software development. Early detection of bugs helps in proper resource utilization .

So we create different defect prediction models which help us to assign limited test resources more reasonably and improve software quality. In this paper, three supervised ML learning classifiersNaïve Bayes (NB) classifier, Decision Tree (DT) classifier and Artificial Neural Networks (ANNs) classifier are used to evaluate the ML capabilities. The discussed ML classifiers are applied to three different datasets obtained from [1] and [2] works..

II. MOTIVATION

Traditional defect predictions are performed at a package or file level. However, when big files are predicted as bug prone by predictors, it can be time-consuming to perform code checkers on them [5]. In addition, since large files have been modified by multiple developers, it is not easy to find the right resource to check the files [6]. To this endsoftware defect prediction (SDP) method is proposed [7–10]. If the bug is induced in the software through changes the defect predictors can identify it immediately . Therefore, Software defect prediction provides fine granularity , and traceability.[7]..

III. REVIEW OF LITERATURE

When the Software defect prediction technology was started in the 1970s [15], the software systems were less intricate. It was found that the number of software defects in a system was related to the number of lines of code. In 1971, Akiyama et al. [16] suggested a formulation for the relationship between the number of software bug (D) and Loc() in a project: . In 1994, Chidamber and Kemerer [18] projected CK metrics for the object-oriented program code. Nagappan and Ball [19] suggested relative code mix metrics to forecast defect density at file level by calculating code churn during development.

In recent years, research related to SDP has been progressive .Mockus and Weiss [20] planned the change metrics to assess the likelihood of defect-inducing variations for early modification requests (IMR) in a 5ESS network switch project. Yang et al. [21] first applied deep learning technology to SDP. The empirical results show that the Deeper method can differentiate more defective changes and have improved F1 indicator than the EALR method. Kamei et al. [10] directed experiment on 11 open-source projects for SDP using cross-project models. Yang et al. [22] proposed a two-layer ensemble learning method , Experimental results show that TLEL can significantly improve the PofB20 and F1 indicators compared to the three baselines. Chen et al. [9] proposed a novel supervised learning method MULTI, which applied multiobjective optimization algorithm to software defect prediction. To this end, Pascarella et al. [23] proposed a fine-grained JIT-SDP model to predict defective files in a commit based on ten open-source projects.

IV. . PROPOSED WORK

A. Proposed System Architecture

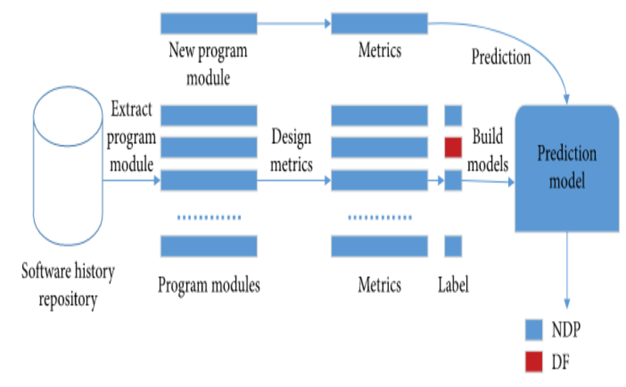

Fig.1The process of software defect prediction

B. System Overview

There are three modelling methods that are used to investigate. First, in the process of using local models, the training set is divided into multiple clusters. Therefore, this process is completed by the k-medoids model and ordering performance and effort-aware prediction performance of local models and global models Details of the three models are shown below

1) ***K-Medoids***: The foremost stage in local models is to use a clustering algorithm where division of training data is done into several consistent regions. In this process, the experiment uses K medoids used in different clustering tasks. K-medoids is less sensitive to the outliers

2) ***Logistic Regression***Logistic regression is used to build classification models [7]. In our experiment, logistic regression is used to solve sorting problems. However, the output of the model is a continuous value with a range of 0 to 1. Therefore, for a change c, if the value of is greater than 0.5, then c is classified as defective otherwise it is classified as clear

3) ***Aware Linear Regression*** "Arisholm et al. [24] noticed out that the use of prediction models are important in testing the logistics parameters along with performance . Cost effectiveness of code inspection of the model should be considered along with classification performance Effort-aware linear regression (EALR) was firstly suggested by Kamei et al. [7] for solving effort-aware SDP, which is grounded on linear regression models. Unlike from the above logistic regression, the dependent factor of EALR signifies the defect proneness of a change and denotes the effort required to code checking for the change c. According to the suggestion of Kamei et al. [7], the effort for a change can gained by calculating the number of modified lines of codes.

V. .RESULT

In this experiment, we firstly apply local models to SDP.The purpose of the study is to compare the prediction performance of local and global models under
1.Cross-validation
2. Cross-project-validation and
3.Time wise-cross-validation for defect prediction.

In order to compare theperformance of local and global models, local models are based on k medoids. In the final outcome it is seen that local models perform bad in the classification performance than global models in three evaluation scenarios. Moreover, local models have worst EALR in the time wise-cross-validation scenario. However, local models performance is outstanding than global models. Particularly, the best effort-aware prediction performance can be obtained when the parameter k of local models is set to 2. Therefore, the most promising model are local models in the context of effort-aware SDP.

VI. .CONCLUSIONS

In the future, first, we plan to consider more marketable and investigational projects to additionalauthorize the validity of the experimental conclusions. We hope to add more baselines to local and global models to test the simplification of trialinferences considering that the optimal of modeling methods will affect the prediction presentation.

ACKNOWLEDGMENT

I wish to thank all the people who gave me an unending support right from the idea was conceived.I express my sincere and profound thanks to our Head of the department and my Guide for their guidance and motivation for completing my work, and I am also thankful to all those who directly or indirectly guided and helped me in preparation of this paper.

REFERENCES

[1] M. Newman, Software Errors Cost US Economy $59.5 Billion Annually, NIST Assesses Technical Needs of Industry to Improve Software-Testing, 2002,
[2] Z. Li, X.-Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," IET Software, vol. 12, no. 3, pp. 161–175, 2018.
[3] X. Chen, Q. Gu, W. Liu, S. Liu, and C. Ni, "Survey of static software defect prediction," RuanJianXueBao/Journal of Software, vol. 27, no. 1, 2016, in Chinese.
[4] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," IEEE Transactions on Software Engineering, vol. 43, no. 1, pp. 1–18, 2017.
[5] A. G. Koru, D. Dongsong Zhang, K. El Emam, and H. Hongfang Liu, "An investigation into the functional form of the size-defect relationship for software modules," IEEE Transactions on Software Engineering, vol. 35, no. 2, pp. 293–304, 2009.
[6] S. Kim, E. J. Whitehead, and Y. Zhang, "Classifying software changes: clean or buggy?" IEEE Transactions on Software Engineering, vol. 34, no. 2, pp. 181–196, 2008.
[7] Y. Kamei, E. Shihab, B. Adams et al., "A large-scale empirical study of just-in-time quality assurance," IEEE Transactions on Software Engineering, vol. 39, no. 6, pp. 757–773, 2013.
[8] Y. Yang, Y. Zhou, J. Liu et al., "Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models," in Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 157–168, Hong Kong, China, November 2016.
[9] X. Chen, Y. Zhao, Q. Wang, and Z. Yuan, "MULTI: multi-objective effort-aware just-in-time software defect prediction," Information and Software Technology, vol. 93, pp. 1–13, 2018.
[10] Y. Kamei, T. Fukushima, S. McIntosh, K. Yamashita, N. Ubayashi, and A. E. Hassan, "Studying just-in-time defect prediction using cross-project models," Empirical Software Engineering, vol. 21, no. 5, pp. 2072–2106, 2016.
[11] T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, and D. R. Cok, "Local vs. global models for effort estimation and defect prediction," in Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 343–351, Lawrence, KS, USA, November 2011.
[12] T. Menzies, A. Butcher, D. Cok et al., "Local versus global lessons for defect prediction and effort estimation," IEEE Transactions on Software Engineering, vol. 39, no. 6, pp. 822–834, 2013.
[13] G. Scanniello, C. Gravino, A. Marcus, and T. Menzies, "Class level fault prediction using software clustering," in Proceedings of the 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 640–645, Silicon Valley, CA, USA, November 2013.
[14] N. Bettenburg, M. Nagappan, and A. E. Hassan, "Towards improving statistical modeling of software engineering data: think locally, act globally!," Empirical Software Engineering, vol. 20, no. 2, pp. 294–335, 2015.
[15] Q. Wang, S. Wu, and M. Li, "Software defect prediction," Journal of Software, vol. 19, no. 7, pp. 1565–1580, 2008, in Chinese.
[16] F. Akiyama, "An example of software system debugging," Information Processing, vol. 71, pp. 353–359, 1971.

[17] T. J. McCabe, “A complexity measure,” IEEE Transactions on Software Engineering, vol. SE-2, no. 4, pp. 308–320, 1976.
[18] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476–493, 1994.
[19] N. Nagappan and T. Ball, “Use of relative code churn measures to predict system defect density,” in Proceedings of the 27th International Conference on Software Engineering (ICSE), pp. 284–292, Saint Louis, MO, USA, May 2005.
[20] A. Mockus and D. M. Weiss, “Predicting risk of software changes,” Bell Labs Technical Journal, vol. 5, no. 2, pp. 169–180, 2000.
[21] X. Yang, D. Lo, X. Xia, Y. Zhang, and J. Sun, “Deep learning for just-in-time defect prediction,” in Proceedings of the IEEE International Conference on Software Quality, Reliability and Security (QRS), pp. 17–26, Vancouver, BC, Canada, August 2015.
[22] X. Yang, D. Lo, X. Xia, and J. Sun, “TLEL: a two-layer ensemble learning approach for just-in-time defect prediction,” Information and Software Technology, vol. 87, pp. 206–220, 2017.
[23] L. Pascarella, F. Palomba, and A. Bacchelli, “Fine-grained just-in-time defect prediction,” Journal of Systems and Software, vol. 150, pp. 22–36, 2019.
[24] E. Arisholm, L. C. Briand, and E. B. Johannessen, “A systematic and comprehensive investigation of methods to build and evaluate fault prediction models,” Journal of Systems and Software, vol. 83, no. 1, pp. 2–17, 2010. ·
[25] Y. Kamei, S. Matsumoto, A. Monden, K. Matsumoto, B. Adams, and A. E. Hassan, “Revisiting common bug prediction findings using effort-aware models,” in Proceedings of the 26th IEEE International Conference on Software Maintenance (ICSM), pp. 1–10, Shanghai, China, September 2010.
[26] Y. Zhou, B. Xu, H. Leung, and L. Chen, “An in-depth study of the potentially confounding effect of class size in fault prediction,” ACM Transactions on Software Engineering and Methodology, vol. 23, no. 1, pp. 1–51, 2014