ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 6, June 2016

Exhuming Software Defect Using Static Defect Model in Data Mining

Bhawna Dubey

M.Tech Student, Department of Computer Science, USCIT, GGSIPU, Delhi, India

ABSTRACT: Lot of research done in mining software repository. In this paper we discussed about the static model of data mining to extract defect .Different algorithms are used to find defects like naïve bayes algorithm, Neural Network, Decision tree. But Naïve Bayes algorithm has the best result .Data mining approach are used to predict defects in software .We used NASA dataset namely, Data rive. Software metrics are also used to find defects.

KEYWORDS: Naïve Bayes algorithm, Software Metric, Solution Architecture,.

I. INTRODUCTION

According to [1], multiple algorithms are combined to show better prediction capability using votes. Naïve Bayes algorithm gives the best result if used individual than others. The contribution of this paper based on two reasons. Firstly, it provides a solution architecture which is based on software repository and secondly it provides benchmarks that provide an ensemble of data mining models in the defective module prediction problem and compare the result. Author used NASA dataset online [2] which contain five large software projects with thousands of modules. Bohem found 80/20 rule and about the half modules are defect free [3]. Fixing Defects in the operational phase is considerably more expensive than doing so in the development or testing phase. Cost-escalation factors ranges from 5:1 to 100:1 [3]. It tells defects can be fixed in operational phase not in development and testing phase. The study of defect prediction can be classified into two categories: Dynamic model and Static model .The dynamic models are used during the software development lifecycle to predict the future number of defects based on the defects distribution over time. These dynamic models use the number of defects detected in the earlier phases of the development process as the independent variable [4]. Defect prediction and estimation models are used to predict the total number of defects and their distribution over time. An old review of these researches is found in [5]. Dynamic defects models follow distribution patterns including Rayleigh distribution and other similar curves [6,7,8]. Ref. [9] found that software estimation models that are based on COQUALMO are useful in facilitating the right balance of activities to meet quality goals.The static model based on product and project metrics and used Bayesian Belief network model and AgenaRisk tool are used [10,11]. Other researches regarding defect models include regression models [12,13], statistical models and machine learning based models [14,15]. This includes artificial neural networks, instance-based reasoning, decision trees, and rule inductions. Many techniques are used as enhancements to predict unknown or missing data like [16]. Other static models include regression models [17] and metric based technology [18]. Other Infrastructures include SUDS for creating dynamic bug detection tools [19,20]. The used software metrics include McCabe metrics, Halstead metrics, branch-count and five different measures representing the lines of code [21]. A software defect prediction framework refer to the system that can predict whether a given software module is defective or not. In general, a software defect prediction model is trained using software metrics and defect data that have been collected from previously developed software releases or similar projects. The model can then be applied to program modules with unknown defect data. The features or attributes of software defect prediction data sets influence the performance and effectiveness of the defect prediction model. Most of the experiments related with Defect Prediction are conducted in a Machine Learning tool or environment called WEKA and SQL Server Data Tool 2015. Since every Organization will try to maintain their data secret, only few datasets are available for public which can be used for experiments. One of the most popular publicly available Datasets includes MDP and PROMISE repository provided by NASA [22]. We work on the future work of the paper [23].

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 6, June 2016

II. RELATED WORK

This paper uses four models for static defect prediction. They are Naïve Bayes, Neural Networks, Association Rules and Decision trees. Bayes net is a graphical representation of a set of random variables and their conditional dependencies via a directed acyclic graph. Here, the random variables are denoted by nodes and their conditional dependencies are denoted by connecting edges. The edges can be directed or undirected. If there is no edge between two nodes, that means they are conditionally independent whereas if there is an edge, that means they are conditionally dependent. Bayes net is often termed as belief networks or causal networks [24]. The Naive Bayes algorithm can be used for both binary and multiclass classification problems. NB builds and scores models extremely rapidly, it scales linearly in the number of predictors and rows. NB makes predictions using Bayes' Theorem, which derives the probability of a prediction from the underlying evidence. Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository. They are readable by the humans. For example, an association rule could be ''If the lines of code is larger than 400, the module are 80% likely to be defective''. The decision trees algorithm is a classification algorithm for use in predictive modeling. The decision trees algorithm builds a data mining model by creating a series of splits in the tree. These splits are represented as nodes. These nodes can be understood by the development team by using the decision tree viewer.

III. SOFTWARE METRICS

a software metric is a standard of measure of a degree to which a software system or process possesses some property. the use of software metrics includes mccabe metrics, halstead metrics, branch count and line of code. mccabe is collection of different metrics like cyclomaticecomplexity, design complexity etc. halstead metric based on base measure, derive measure and loc measure.

Project ID	Development Location	Project	Laungage	Numbers of Modules	% of defective modules
01	Location 9	DATATRIEVE Transition	C	130	8.46

TABLE 1 PROPERTIES OF THE SOFTWARE PROJECTS DATASETS USED IN THE REPOSITORY

IV. EXPERIMENTAL RESULTS

Data is analyzed using the four different data mining algorithms. The results and discussions of executing the aforementioned data mining models are shown in the following subsections under the headings: association algorithm, Naı¨ve Bayes, Neural Networks, Association Rules and Decision Tree. Each section contains the results of the ''All Modules Data Table'' that represents the four projects. The section is finalized with a performance evaluation comparison of the algorithms.

Datarive	
LOC6_0	0.998457
LOC6_02	0.91721
AddedLOC	0.869719
DeletedLOC	0.743272
DifferentBlocks	0.211792
ModificationRate	0.102613
ModuleKnowledge	0.10438
ReusedLOC	-0.08221

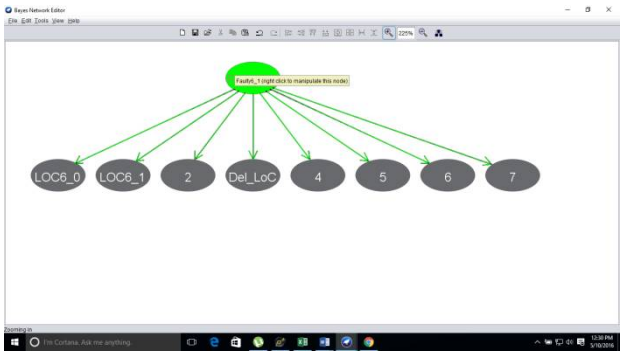
ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 6, June 2016

SIMPLE CORRELATION



Dependency network of the Naïve Bayes classifier for the all projects dataset

V. CONCLUSION

In this research we use Navie Bayes Classifier, Decision tree Classifier, Association Rules and Neural Network Classifiers with the help of Weka, Excel and SQL Server Data Tool. We use these classifiers in more details to show Naïve Bayes Algorithm is better than others in static defect model. We can suggest same work with different defect model and compare the result.

REFERENCES

1. Ahmed H. Yousef ,"Extracting software static defect models using data mining", Ain Shams Engineering Journal Vol. 6, 133–144,2015
2. Kelly JC, Sherif JS, Hops J." An analysis of defect densities found during software inspections." J. Syst. Softw.;17(2):111–7. 1992
3. Boehm B, Basili VR. "Software defect reduction top 10 list. In: Foundations of empirical software engineering": the legacy of Victor R. Basili; p. 426,2005.
4. Putnam LH. "A general empirical solution to the macro software sizing and estimating problem". IEEE Trans SoftwEng ,SE- 4(4):345–61,1978.
5. Clark B, Zubrow D. "How good is the software: a review of defect prediction techniques". Sponsored by the US department of Defense; 2001.
6. Li PL et al. "Empirical evaluation of defect projection models for widely-deployed production software systems", vol. 29. ACM; 2004.
7. Bergander T, Yan L, Ben Hamza A. "Software defects prediction using operating characteristic curves". In: IEEE internationalconference on information reuse and integration, IRI 2007; 2007.
8. Raja U, Hale DP, Hale JE." Modeling software evolution defects: a time series approach". J SoftwMaintEvol: Res Pract ;21(1):49–71.2009
9. Yousef AH. "Analysis and enhancement of software dynamic defect models". In: International conference on networking and media convergence, ICNM 2009.
10. Gopal A et al. "Measurement programs in software development: determinants of success". IEEE Trans SoftwEng ;Vol 28(9): 863–75.2002
11. Neil M, Fenton N. "Improved software defect prediction". In: 10th European SEPG, London; 2005.
12. Muhammad Dhiauddin, Mohamed Suffian SI." A prediction model for system testing defects using regression analysis". Int J Soft ComputSoftwEng (JSCSE) ;2(7),2012.
13. Ostrand T, Weyuker E. "Progress in automated software defect prediction". In: Chockler H, Hu A, editors. Hardware and software: verification and testing. Berlin Heidelberg: Springer;
14. Challagulla VUB, et al. "Empirical assessment of machine learning based software defect prediction techniques." In: 10th IEEE international workshop on object-oriented real-time dependable systems, WORDS 2005; 2005.
15. Challagulla VUB et al. "Empirical assessment of machine learning based software defect prediction techniques". Int J ArtifIntell Tools;17(02):389–400,2008.
16. Fakhrahmad S, Sami A." Effective estimation of modules' metrics in software defect prediction". In: Proceedings of the world congress on engineering; 2009.
17. Khoshgoftaar TM, Seliya N. "The necessity of assuring quality in software measurement data". In Proceedings 10th International Symposium on Software Metrics; 2004.
18. Sunghun K, Whitehead EJ, Yi Z." Classifying software changes: clean or buggy? "IEEE Trans SoftwEng ;34(2):181–96,2008.
19. Larson E. SUDS:" An Infrastructure for Creating Bug Detection Tools". In: Seventh IEEE international working conference on source code analysis and manipulation, SCAM 2007; 2007.
20. Larson E. SUDS: "an infrastructure for creating dynamic software defect detection tools". AutomSoftwEng ;17(3):301–46,2010.
21. Menzies T, Di Stefano JS. "How good is your blind spot sampling policy". In: Proceedings eighth IEEE international symposium on high assurance systems engineering. IEEE; 2004.
22. Reena P &BinuRajan "Software Defect Prediction System –Decision Tree Algorithm With Two Level Data Preprocessing", (IJERT) ,2014
23. A. Günes. Koru and Hongfang Liu "An Investigation of the Effect of Module Size on Defect Prediction Using Static Measures"
24. SakthiKumaresh , MeenakshySivaguru " Software Defect Classification using Bayesian Classification Techniques"