

# Performance Analysis and Enhancement of Software Quality Metrics using Decision Tree based Feature Extraction.

N Gayatri<sup>1</sup>, S Nickolas<sup>2</sup>, A.V.Reddy<sup>3</sup>  
Department of Computer Applications  
National Institute of Technology  
Tiruchirappalli, India

<sup>1</sup> Research scholar, <sup>2</sup> Assistant Professor, <sup>3</sup> Professor  
<sup>1</sup>gayatrinandam@yahoo.co.in, <sup>2</sup>nickolas@nitt.edu, <sup>3</sup>reddy@nitt.edu

**Abstract**— Software managers routinely face the problem with software projects that contain error or inconsistencies which increases the budget, time limit and reduces the customer satisfaction. By applying the data mining technique for software metrics dataset as a quality prediction model, helps manager to tackle the above problems in an efficient way and improve the quality. In this paper, Decision tree classifier, a data mining technique is used as the software quality prediction model and applied on the reduced dataset for evaluating the overall performance by measuring the Accuracy, Mean Absolute Error(MAE), Root Mean Square Error(RMSE), Precision and Recall. The observations show that the overall accuracy is increased when applied with the reduced dataset than the original data set. Hence we can term that the reduced dataset play a vital role in the improving the software quality by increasing the classifier performance and hence can be said that the attributes in reduced dataset are enough as good predictors for future classification. The dataset used here is the KC2 NASA data set. To evaluate the performance of decision tree classifier, measures such as Accuracy, MAE, RMSE, Precision and Recall are used.

**Index Terms**— software quality, software metrics, classification and decision trees

## I. INTRODUCTION

Metrics are promising and potentially very valuable for providing standards for efficient software quality outcome. Software metrics are collected at various phases of software development process which contains information of software and can be used to predict the quality in the early stage of life cycle. Recent studies show that the software metrics can be used in software fault prediction and software quality prediction using relationship between software metrics and software module's fault proneness [1]. In the case of quality predictions software metrics are used to predict the software bugs in the software module and in the case of software quality classification, we use software metrics to classify software modules as fault prone or not fault prone. Software defects/faults play an important role in quality prediction and early identification of faults/bugs/risks in the program helps in reducing the risks at the end and to get a better quality product. Recently machine learning techniques are applied as a software quality prediction model [2,4]. This paper

focuses on prediction of faults in software modules by the use of Data mining techniques.

By applying data mining techniques for software metrics predictive models can be introduced which helps managers to tackle the time, quality and budgeting problems easily in efficient way and Classification is one of the important Data mining technique in which researches are interested for improving the quality of software. The existing techniques for software classification are Logistic regression, Discriminant analysis and Artificial Neural networks. Decision trees is one of the popular classification technique which is being used rapidly for software quality prediction recently and this approach can be interpreted by using the tree structure where at the root nodes which software metrics are useful can be known to predict the quality.

In this paper the decision tree algorithms J48, CART, BFTree are explored for software fault prediction. All methods measure performance based on Accuracy, MAE, RMSE, Precision, and Recall and for performance evaluation cross validation is used. From the previous work we have identified that only some metrics take an active part for forming rules and can be considered as the future predictors when experimented with KC2 NASA dataset [7] which contains 22 attributes. In this context, we experimented and evaluated the performance of the above algorithms using the reduced no of attributes which took active part in the previous work. We have observed that overall accuracy is increased with reduced data set and also there is reduction in the error rate. This shows that the performance of classifier increases as the number of attributes reduced, by removing the attributes affecting the classification process. Hence we can say that not all 22 attributes present in the data set must be used for software quality prediction but only 15 attributes can be used for further predictions and to evaluate the better performance. The paper is structured as follows: section 2 provides detailed insight of metrics used in the data set. Section 3 provides the overall view of data mining classification techniques. In section 4 experimental results are discussed. Conclusions and scope for the future work is given in the section 5.

II. SOFTWARE METRICS

The NASA dataset [7] contains twenty two measurement attributes which can be used for quality prediction. These are discussed in this section.

One of the basic objectives of software engineering is to transform the creation of software systems from poorly understood activity into a carefully controlled, methodical and presentable enterprise. Developers should be able to express characteristics of system in to quantitative terms to make a software product an engineer able [3]. Quantitative measures are collectively referred as software metrics. Basically metrics are divided as Process metrics, Product metrics and Project/resource metrics.

Here in this paper we are considering product metrics. Product metrics are defined in terms of features that are not visible until the detail coding has been completed. The product software metrics deal with the characteristics of source code for a software project and these are subdivided in: Size Metrics, Complexity Metrics, and Halstead’s Software Metrics.

1) Size Metrics

Size Metrics are represented by a number of metrics attempt to quantify software “size”. One of such metrics is line of code (LOC). The size of the product is given only after completion of detailed coding. Software size can be described with three attributes length, functionality and complexity [3].

- Length is the physical size of product.
- Functionality measures the functions supplied by the product.
- Complexity can be interpreted in different ways: Problem complexity, Algorithm complexity, Structural complexity.

To compute LOC, NCLOC, CLOC are required

$$LOC = NCLOC + CLOC \quad (1)$$

Where NCLOC is non commented LOC and CLOC is commented LOC.

2) Complexity Metrics

Complexity measures are Halstead measures and McCabe’s complexity measures.

a) Halstead measures

Halstead proposed a set of metrics for the overall software production effort. Some of metrics are Program vocabulary, Estimated length, Purity ratio, Length, Program volume (v) etc.

According to Halstead, computer program can be visualized as sequence of tokens, each token being classified as either an operand or operator.

$$\text{Length} \quad N = N1 + N2 \quad (2)$$

Where N1 =total number of operators, N2=total number of operands.

$$\text{Vocabulary} \quad n = n1 + n2 \quad (3)$$

Where n1 is the unique operators in the program, n2 is the unique operands in the program

$$\text{Volume} \quad V = N \log_2 n \quad (4)$$

Where V’ is the volume of most compact design implementation. This is good measure of program understandability.

b) McCabe’s complexity metrics

MCCabe’s complexity measures are based on control flow representation of the program. A program graph is used to depict control flow. In graph, nodes represent processing tasks .Edges represents control flow between nodes [6].The cyclomatic complexity is given as

$$V(g) = e - n + 2 \quad (5)$$

where e is the no of edges is the number of nodes.V(g) is the number of region /areas of the planar graph. It is said that generally complexity should not be more than 10, if it is then program code is not efficient.

III. CLASSIFICATION AND DECISION TREES

Classification is an important data mining technique that has wide range of applications including medical diagnosis, scientific experiments etc., The input to the classification problem is a data set of training records and each record has several attributes and the goal is to build a concise model of distribution of class label in terms of predictor attributes. Many classification models have been proposed such as neural networks, support vector machines, Decision trees (DT) and others. Decision trees are more attractive than others [5] because of several reasons like, they can be constructed relatively fast when compared to others, they are non-parametric and thus suited for exploratory knowledge discovery, and accuracy of classification can be comparable to other models.

Here we employed decision trees to predict the risk of a function with the help of all metrics as predictors. Decision tree is one of the most popular and inductive learning algorithms. It is like a tree structure of flow chart. The nodes inside of the tree correspond to the test of an attribute, and the links (linking attribute values and the leaves) present the output of a test, while the leaves represent the classification of the classes. To induce a Decision Trees, the most important attribute (according to an attribute selection criteria, such as information gain, Gain Ratio, etc.) is selected and placed at the root; one branch is made for each possible attribute value. This divides the samples into subsets, one for each possible attribute value. The process is repeated recursively for each subset until all samples at a node have the same classification, in which case a leaf node is created. To classify a sample we start at the root node of the tree and follow the path corresponding to the sample’s values until a leaf node is reached and the classification is obtained. To prevent overtraining, DT is typically pruned [6]. Here we have used three decision tree algorithms J48, CART, BFTree to evaluate the performance using the original dataset and the reduced data set.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

The performance of the decision tree algorithms is evaluated based on Accuracy and Error rate using the NASA software metrics datasets named KC2 and JMI [7]. These are real datasets available for fault predictions. The KC2 data set contains 522 instances .5000 instances

of JMI data set are taken for experimental purpose, both have 22 attributes which are used for quality prediction up to date. In our paper we have experimented the above datasets using WEKA tool to evaluate the performance of the above algorithms. From [10] we have identified that all 22 attributes are not needed for quality prediction or classification, only 15 attributes are enough. Those attributes be are shown in Table 1. The experiment is carried out on these 15 attributes using the WEKA tool[8] and observed that they have better results than when using all the attributes. The overall accuracy is increased and error rate is also decreased comparatively and this conveys the effectiveness of the model.

TABLE I. NECESSARY ATTRIBUTES FOR DECISION TREE

No	Attribute	Attribute Information
1	loc	Line of code
2	v(g)	McCabe's cyclomatic complexity
3	ev(g)	McCabe's essential complexity
4	iv(g)	McCabe's design complexity
5	l	Program length
6	d	Difficulty
7	i	Intelligent
8	b	Halsted parameter
9	LOCcode	Halstead line count
10	LOCcomment	Halstead count of lines and comments
11	LOBlank	Halstead count of blank lines
12	LOCcode And Comment	Count of lines and comments
13	uniq_Opnd	Unique operands
14	total_Opnd	Total operands
15	defects	True or false

The comparison of accuracy, error rate for the original data set and reduced data set are depicted graphically in Figure 1 and also the values are tabulated in Table II. The experiment is carried using WEKA tool [8] on both datasets.

The results suggest that there is 2-3% increase in accuracy and decrease in error rate using reduced data set i.e. little increase in accuracy and decrease in error rate, which can be considered as a measure of enhanced effectiveness. So we can conclude that these 15 attributes are enough for further prediction. Comparison is done based on Accuracy, MAE, RMSE measures. Precision and Recall measures are also applied for evaluating the performance of the classifier.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have compared the performance of decision tree classifier using the software metrics data set with 22 attributes and also with data set containing 15 attributes for quality prediction. From the results we observed that the classifier performance with 15 attributes i.e. reduced software metrics data set is better than the original data set which contains 22 attributes. So only 15 attributes play an important role in software defect/quality prediction as the accuracy is increased and error rate is decreased with the above attributes and these are enough for building future classification model. The future scope will be the application of the dataset with 15 attributes to any new and other classifiers to check the overall performance for quality enhancement and to find out the individual good predictors which take part in

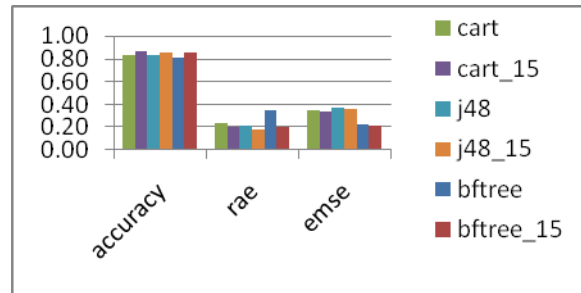


Figure 1. Comparison of DM Algorithms with 22 and 15 attributes

TABLE II. COMPARISON OF PERFORMANCE MEASURES WITH 22 AND 15 ATTRIBUTES

Alg name	Data set with 22 attributes			Data set with 15 attributes		
	MAE	RMSE	Acc	RAE	RMSE	Acc
Data set KC2						
J48	0.2119	0.3716	83.33%	0.1831	0.3559	85.33%
CART	0.232	0.3492	83.72 %	0.2052	0.3382	86.72 %
BFTree	0.2266	0.4024	81.03%	0.2047	0.3522	85.03%
Data set JMI						
J48	0.4018	0.511	64.72%	0.4063	0.5052	64.92%
CART	0.4366	0.4766	64.84%	0.4344	0.475	65.1%

the identification of risks in the program.

REFERENCES

- [1] Fenton, N.E., Neil, M., "Software metrics: successes, failures and new directions." The Journal of Systems and Software 47, 149-157, 1999.
- [2] Khoshgoftaar, T., Seliya, N., "Analogy-based practical classification rules for software quality estimation". Empirical Software Engineering 8 (4), 325-350, 2003.
- [3] N.E. Fenton and S.L Pfleeger, "Software Metrics, A Rigorous & Practical Approach", International Thomson Computer Press, London, 1997.
- [4] Monica Chiş, "Evolutionary Decision Trees and Software Metrics for Module Defects", International Journal of Computer and Information Science, Volume 2, pp. 273-277, 2008.
- [5] Han, J., & Kamber, M., "Data Mining: Concepts and Techniques", San Francisco: Morgan Kaufmann Publishers, 2001.
- [6] Zhihong Xu, Xin Zheng, and Ping Guo, "Empirically Validating Software Metrics for Risk Prediction Based on Intelligent Methods", Proceedings of the Sixth International Conference on Intelligent Systems Design and applications (ISDA'06), 0-7695-2528-8/06, 2006 IEEE
- [7] Promise Software Engineering <http://promise.site.uttowa.ca/SERpository>
- [8] WEKA: <http://www.cs.waikato.ac.nz/ml/weka>
- [9] Oliver Vandercruys, David Martens, Bart Basens, C, "Mining software repositories for Comprehensible software fault Prediction models", Journal of system software 81, pp 823-839, 2007.
- [10] N.Gayatri, S.Nickolas, A.V.Reddy, R.Chitra, "Performance Analysis of data mining Algorithms for Software Quality Prediction", accepted for International Conference, ARTCOM 2009, Kottayam, KERALA, October 2009.