International Journal of Information Technology and Knowledge Management
January-June 2008, Volume 1, No. 1, pp. 97-102

FAULT PREDICTION USING MACHINE LEARNING TECHNIQUES

Deepali Gupta

Abstract

Faults in software systems continue to be a major problem and many systems are delivered to users with excessive faults. This is been recognized that seeking out fault-prone parts of the system and targeting those parts for increased quality control and testing is an effective approach to fault reduction. Being able to estimate software faultiness before and during testing and analysis activities would greatly help software testing and analysis. Many systems are delivered to users with excessive faults. Prediction models based on software metrics, can estimate number of faults in software modules.

In this paper different machine learning algorithms and neural network techniques on two different real-time software defect datasets are evaluated. The results show that when all the prediction techniques are evaluated then best algorithm for classification of the software components into faulty/fault-free systems is found to be Generalized Regression Neural Networks.

Keyword: Fault proneness, Software metrics, Software reliability, Machine learning techniques, Neural Network algorithms and Software quality.

1. Introduction

Quality of software is increasingly important and testing related issues are becoming crucial for software. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiency of software testing. Being able to measure the fault-proneness of software can be a key step towards steering the software testing and improving the effectiveness of the whole process. A huge amount of development effort is going into fault reduction in terms of quality control and testing. It has been long recognized that a limited amount of valuable work in that area has been carried out previously. Timely predictions of faults in software modules can be used to direct cost-effective quality enhancement efforts to modules that are likely to have a high number of faults [3].

Metrics is defined as "The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products". Software metrics can be classified as either product metrics or process metrics. Product metrics are measures of the software product at any stage

of its development, from requirements to installed System. Product metrics may measure the complexity of the software design, the size of the final Program or the number of documentation produced. Process metrics on the other hand, are measures of the software development process, such as overall development time, type of methodology used, or the average level of experience of the programming staff. Using software complexity measures, the techniques build models, which classify components as likely to contain faults or not. The modeling techniques include principal component analysis, discriminate analysis, logistic regression, logical classification models, and layered neural networks.

The rest of the paper is organized as follows. Section 2 gives overview of the problem formulation. Present work is presented in Section 3. Section 4 focuses on conclusion of this work.

2. PROBLEM FORMULATION

Fault-proneness of a software module is the probability that the module contains faults. A correlation exists between the fault-proneness of the software and the measurable attributes of the code (i.e. the static metrics) and of the testing (i.e. the dynamic metrics). Early detection of fault-prone software components enables verification experts to concentrate their time and resources on the problem areas of the software system under development [4].

Prediction of fault-prone modules:

1. Supports software quality engineering through improved scheduling and project control.
2. Can be a key step towards steering the software testing and improving the effectiveness of the whole process by planning and executing testing activities.
3. Enables effective discovery and identification of defects.
4. Enables the verification and validation activities focused on critical software components.
5. Used to improve software process control and achieve high software reliability.
6. Can be used to direct cost-effective quality enhancement efforts to modules that are likely to have a high number of faults.

From previous experiments it has been observed that:

1. Existence of a correlation between a reasonable set of static metrics and software fault proneness [1].

2. Use of various methods for predicting fault-prone modules in software development projects [5].
3. Size and complexity metrics are not sufficient for accurate prediction.
4. There is a need to find the best prediction techniques for a given prediction problem.

A wide variety of techniques have been proposed [2]. The modeling techniques cover the main classification paradigms, including principal component analysis, discriminate analysis, logistic regression, logical classification models and layered neural networks [6].

3. PRESENT WORK

The real-time faulty data sets used in this paper can be accessed from NASA's MDP (Metric Data Program) data repository, available online at *http://mdp.ivv.nasa.gov*. The CM1 data is obtained from a spacecraft instrument, written in C, containing approximately 506 modules. The PC1 data is collected from a flight software system coded in C, containing 1108 modules.

Table 1
Classes of Machine Learning Algorithms

Sr. No.	*Classes of Machine Learning Algorithms*
1.	Bayes
2.	Functions
3.	Lazy
4.	Meta
5.	Misc
6.	Trees

These complexity and size metrics include well known metrics, such as Halstead, McCabe, line count, operator/operand count, and branch count metrics. There are six main classes of machine learning algorithms as shown in Table 1. Regarding the software, the freely available WEKA machine learning tool kit and Mat lab were used to conduct these experiments. The WEKA software computes the mean absolute error, root mean squared error, relative absolute error, and root relative squared error. However, the most commonly reported error is the mean absolute error and root mean squared error. A variety of many machine learning algorithms and neural network techniques are analyzed.

CONCLUSION

Prediction of fault-prone modules supports software quality engineering through improved scheduling and project control and helps in steering testing activities by planning them.

Table 2
CM1 Details of the Output of Software

Label	*Count*
0	457
1	36
2	6
3	3
4	2
5	1

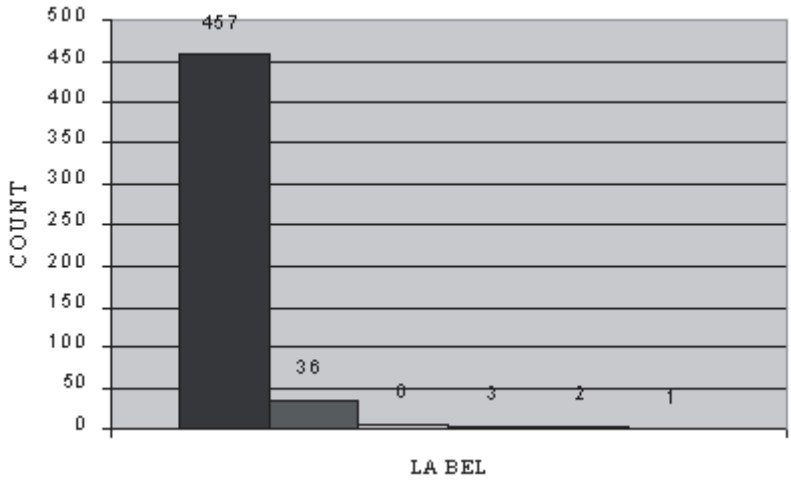

Figure 1: CM1 Graphical Details of the Output of Software

Table 2 shows the CM1 details of the output of software where label i.e. error count is meant for output and the count tells the number of occurrences of that label in the CM1 data set. Fig.1 shows the CM1 graphical representation of details of the output of software. Table 3 shows the PC1 details of the output of software where label i.e. error count is meant for output and the count tell the number of occurrences of that label in the PC1 data set. And the PC1 Graphical representation of details of the output of software is shown in Fig.2.

Table 3
PC1 Details of the Output of Software

Label	*Count*
0	1031
1	46
2	14
3	10
4	2
5	4

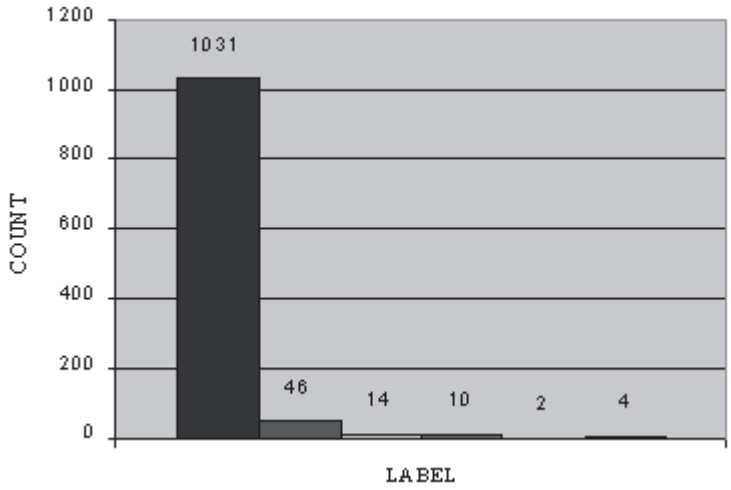

Figure 2: PC1 Graphical Details of the Output of Software

On comparing all the six classes of machine learning techniques, the *MAE* and *RMSE* value for both CM1 and PC1 is lower in case of Generalized Regression

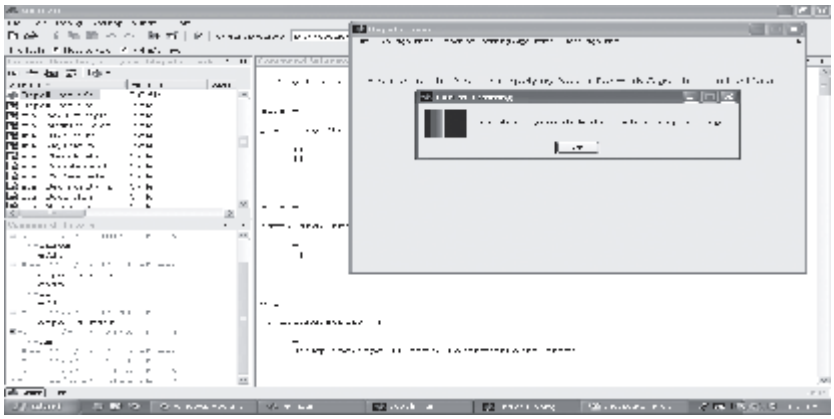

Figure 3: GUI for Generalized Regression Neural Networks

Networks. The graphical user interface shows detail of Generalized Regression Neural Networks as shown in Fig. 3.

So, Generalized Regression Networks gives the best performance among all machine learning techniques and neural network algorithms.

REFERENCES

[1] Fenton N. E. and Neil M., (1999), "A Critique of Software Defect Prediction Models", *IEEE Trans. Software Engineering,* **25**, 675-689, Bellini, I. Bruno, P. Nesi, D. Rogai, University of Florence.

[2] Lanubile F., Lonigro A., and Visaggio G., (1995), "Comparing Models for Identifying Fault-Prone Software Components", Proceedings of *Seventh International Conference on Software Engineering and Knowledge Engineering*, 12-19.

[3] Giovanni Denaro, (2000), "Estimating Software Fault-Proneness for Tuning Testing Activities", Proceedings of the *22nd International Conference on Software Engineering (ICSE2000)*, Limerick, Ireland.

[4] Mahaweerawat, A., "Fault-Prediction in Object Oriented Software's Using Neural Network Techniques", Advanced Virtual and Intelligent Computing Center (AVIC), Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand, 1-8.

[5] Runeson, Claes Wohlin and Magnus C. Ohlsson, (2001), "A Proposal for Comparison of Models for Identification of Fault-Proneness", Dept. of Communication Systems, Lund University, LNLS 2188, 341-355.

[6] Quah, T. and Thwin, M. (2002), "*Application of Neural Networks for Predicting Software Development Faults Using Object-Oriented Metrics*" Proceedings of Ninth International Conference on Neural Information Processing (ICONIP'02), **5**, 2312-2316.

Deepali Gupta
Department of Compter Science, GIMT
Kurukeshetra University, Haryana, INDIA
E-mail: deepali_gupta2000@yahoo.com