Software Fault Forecasting Based on Statistical Attributes

Rita Rani[1], Vikas Gupta[2]
[1]Research Scholar, [2]Assistant Professor
[1,2]Department of Computer Science, Rayat Institute of Engineering and Information Technology, Railmajra, SBS Nagar, (Punjab) INDIA

***Abstract* – The growing demand for higher operational effectiveness and reliability in industrial processes has resulted in a huge attention in fault detection techniques. Researchers and practitioners are remains concerned with correct prediction when developing systems. On the other hand the most popular research area is software fault or fault prediction. Software fault prediction has security, reliability and financial benefits in technical systems by preventing future failures and further improves process upholding schedules. Software fault prediction facilitates to software engineers to attention development activities on defect less code which enhance the software quality, reliability and minimize the cost and time to develop software system in today's era of dynamic scenario of globalization. There are many prediction models which are used to filter the software defects. The present study empirically explorers the viability of reducing the software defect prediction based on statistical factors. Further, the study attempts to offer the future prospective in other dimensions like programming languages and for mapping the relation of attributes and fault tolerance.**

Keywords* – *Software defect, Fault Forecasting, Fault Proneness, Statistical attributes

I. INTRODUCTION

Faults contain in software systems and it continue to work, is a major problem in future. A software bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from producing an incorrect result. A software fault is a deficiency that causes software failure in an executable product. In software engineering, the non-conformance of software to its requirements is commonly called a bug. Most bugs arise from mistakes and errors made by people in either a program's source code or in its design, and a few are caused by compilers producing incorrect object code [1]. The ideal prediction of faults in software are expected to occur in code can help direct testing effort, minimize expenses and recover the quality of software. Our aim is to explore how the context models and independent variables used and the modeling techniques functional, manipulate the performance of fault prediction models. A software quality model is a useful tool for meeting the objectives of software reliability and software testing initiatives of different projects. Different Modeling techniques can be used to identify fault free modules [2,3]. Absence of sufficient tools to guess and evaluate the price for a software system failure is one of the main challenges in software engineering. They used old dataset of software to make and authenticate estimation or prediction system of software development efforts, which allows them to compose management decisions, such as resource allocation. The use of single feature of software to predict faults is not helpful. Fenton uses an example where the same program functionality is achieved using dissimilar programming language constructs resulting in dissimilar static measurements for that module [4]. Fenton uses this example to argue the uselessness of static code attributes. However, where single feature fail and combination succeed [5], hence combination of static features extracted from Software modules support helps to improve resource planning and scheduling as facilitating cost avoidance by efficient verification. Such models can be used to predict the response variable which can either be the class of a module (e.g. fault-prone or not fault-prone) or a quality factor (e.g. number of faults) for a module. The basic hypothesis of software quality prediction is that software currently under development is fault prone if a module with the similar product or process metrics in previous project developed in the same environment was fault free. Therefore, the figures available early within the existing project or from the earlier project can be used in making predictions. This technique is very useful for the large-scale projects or projects with multiple revisions [6,8]. Researchers discussed and analyze methodological problems of educational inquiry and especially the distinction between quantitative and qualitative approaches of research. Problems in that area have been of concern for a long time and have been the reason for many debates among educational researchers since mid 19thcentury. During the 1970s and 80s the critique against quantitative research which had dominated the field for several decades got so extensive that some authors have called this period an era of 'paradigm wars' (Gage 1989, Hamersley 1992). Thus, during last decades there has been fundamental disagreement in many aspects concerning research methodology and the principles, which should underlie educational research and it is obvious from the methodological literature that the debates on these matters are still going on. On the other hand there has been recently a serious critique of the quality of present educational research practice. Many authors have been worried that the lack of consensus in methodological issues and continuing 'paradigm wars' may have 'serious implications for the nature and function of educational research [7].Then machine learning algorithms has been discovered and different techniques has to be applied by using WEKA software. Waikato

Environment for Knowledge Analysis a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. WEKA is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. The WEKA workbench contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality. The original non-java version of WEKA was a TCL/TK front-end to (mostly third-party) modeling algorithms implemented in other programming languages, plus data preprocessing utilities in C, and a Make file-based system for running machine learning experiments. This original version was primarily designed as a tool for analyzing data from agricultural domains, but the more recent fully Java-based version (WEKA 3). For which development started in 1997, is now used in many different application areas, in particular for educational purposes and research. Advantages of WEKA include:

- Free availability under the GNU General Public License.
- probability, since it is fully implemented in the Java programming language and thus runs on almost any modern computing platform
- a comprehensive collection of data preprocessing and modeling techniques
- Case of use due to its graphical user interfaces.

III. PROPOSED METHOD

In this proposed method two techniques are applied by using machine learning algorithms and fault prediction is done on the basis of these two techniques. Two parameters namely mean absolute error (MAE) and root mean squared error (RMSE) are used to evaluate the algorithms. Mean absolute error, MAE is the average of the difference between predicted and actual value in all test cases. It is the average prediction error. The formula for calculating MAE is given as

$$\frac{|a_1-c_1|+|a_2-c_2|+\ldots+|a_n-c_n|}{n}$$

Where the actual output is ‘a’ and expected output is ‘c’.
Root mean squared error (RMSE) is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated. It is just the square root of the mean square error as shown in equation

$$\sqrt{\frac{(a_1-c_1)^2+(a_2-c_2)^2+\ldots+(a_n-c_n)^2}{n}}$$

The mean-squared error is one of the most commonly used measures of success for numeric prediction. This value is computed by taking the average of the squared differences between each computed value and its corresponding correct value. The root mean-squared error is simply the square root of the mean-squared error. The root mean-squared error gives the error value the same dimensionality as the actual and predicted values

IV. RESULTS AND DISCUSSIONS

The results are calculated for the fault prediction of software using statistical attributes. These attributes may be qualitative or quantitative attributes. These are calculated by using machine learning algorithms decision trees and M5 rules classifiers. Every classifier is divided into as per two parameters; root mean squared error and mean absolute error. The fault prediction of software system is done on the basis of both of these values by using iterations which increases the accuracy of the software and comparisons is done between both of the algorithms and previous algorithms. The comparisons are made on the basis of the more accuracy and least value of MAE and RMSE error values. Accuracy value of the prediction model is the major criteria used for comparison. The mean absolute error is chosen as the standard error. The technique having lower value of mean absolute error is chosen as the best fault prediction technique.

M5 Rule result.

- Mean absolute error 252.5203
- Root mean squared error 374.7671

Decision Table Rule result.

- Mean absolute error 211.6015
- Root mean squared error 311.6675

V. CONCLUSIONS

Software Fault prediction is an important topic in software engineering. Software Fault prediction models have the potential to improve the quality of software systems and reduce the expenditure related with delivering those systems. The study evaluated the performance of various machine learning techniques for the fault dataset. Techniques have shown better results than other algorithms with lower values of MAE, RMSE and accuracy is implemented using WEKA, and with these values it is concluded that decision trees is the best algorithm based on these two techniques. Despite a set of fault prediction

studies, there is need to explore and have more research studies with reliable methodology and practical applicability in other dimension like programming languages and for mapping the relation of features and fault tolerance so that software defects could be forecasted and mitigated at the very genesis

REFERENCES

[1] Catal., Diri, "A fault prediction model with limited fault data to improve test process", manmara research centre, information technologies institute Kocaeli, TURKEY.

[2] Seliya N., Khoshgoftaar T.M. and Zhong S. (2005), "Analyzing software quality with limited fault-proneness defect data", in proceedings of the Ninth IEEE international Symposium on High Assurance System Engineering, Germany, pp.89-98.

[3] Jiang Y., Cukic B. and Menzies T. (2007), "Fault Prediction Using Early Lifecycle Data". ISSRE 2007, the 18th IEEE Symposium on Software Reliability Engineering, IEEE Computer Society, Sweden, pp. 237-246.

[4] Bezdek J.C., Ehrlich R., and Full W. (1984) "FCM: Fuzzy c-means algorithm". Computers and Geoscience, Volume: 10, pp. 191-203.

[5] Sarah Beecham, David Bowes, David Gray and Steve Counsell "A Systematic Literature Review on Fault Prediction Performance in Software Engineering" Tracy Hall,

[6] Challagula, Bastani B. and Yen (2006). "A Unified framework for Defect Data Analysis using the MBR Technique". 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), Washington, pp. 39-46.

[7] Katrin Niglas Tallinn., Pedagogical University Dept. of Math. & Comp. Science Narva mnt 25, 10120, Tallinn Estonia

[8] Norman Fenton, Martin Neil, William Marsh, Peter Hearty, Lukasz Radlinski, Paul Krause, "Project Data Incorporating Qualitative Factors for Improved Software Defect", Proceedings of the PROMISE workshop, Year: 2007.