The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.47 e-ISSN: 2251-7545

Model for Software Errors Prediction Using Machine Learning to Improve The Software Reliability

Bonthu Kotaiah
Research Scholar
Babasaheb Bhimrao Ambedkar
Lucknow, India
kotaiah_bonthuklce@yahoo.com

Raees Ahmed Khan
Associative Professor
Babasaheb Bhimrao Ambedkar
University
Lucknow, India
khanraees@yahoo.com

Muralidhar Vejendla
Associate Professor
TEC,TENALI
Andhra Pradesh India
vmdharprof@gmail.com

Abstract — The Software projects become critical systems now a days. Measuring software reliability in a continuous and disciplined manner leads to accurate estimation of project costs and schedules, and improving product and process qualities. Also, detailed analysis of software metric data gives important clues about the locations of possible errors in a programming code. The objective of this paper is to establish a method for identifying software errors using machine learning methods. We used machine learning methods to construct a two step model that predicts potentially modules with errors within a given set of software modules with respect to their metric data by using Artificial Neural Networks. The data set used in the experiments is organized in two forms for learning and predicting purposes; the training set and the testing set. The experiments show that the two step model enhances error prediction performance to improve the Software Reliability.

I. INTRODUCTION

The results of the software reliability measurement are usually evaluated with naive methods like regression and correlation between values. Some recent models utilize machine-learning techniques for error predicting (Neumann, 2002). But the main drawback of using machine learning in software error prediction is the scarcity of data. Most of the companies do not share their software metric data with other organizations so that a useful database with great amount of data cannot be formed. However, there are publicly available well-established tools for extracting metrics such as size, McCabe's cyclomatic complexity, and Halstead's program vocabulary. These tools help automating the data collection process in software projects to measure the Software Reliability.

The software metric data gives us the values for specific variables to measure a specific module/function or the whole software. When combined with the weighted error/error data, this data set becomes the input for a machine learning system. A learning system is defined as a system that is said to learn from experience with respect to some class of tasks and performance measure, such that its performance at these tasks improve with experience (Mitchell, 1997). To design a learning system, the data set in this work is divided into two parts: the training data set and the testing data set. Some predictor functions are defined and trained with respect to Multi-Layer Perceptron and the results are evaluated with the testing data set.

The second section gives a previous work done and the third section deals with dataset used. The fourth section states the Research Problem and the fifth section explains our proposed model for error prediction. In the sixth section, the results of the experiments are shown. The last section concludes our work and summarizes the future work to be done.

II. RELATED WORK

A. METRİCS AND SOFTWARE RİSK ASSESMENT

Software metrics are mostly used for the purposes of product quality and process efficiency analysis and risk assessment for software projects. Currently there are numerous metrics for assessing software risks. The early researches on software metrics have focused their attention mostly on McCabe, Halstead and lines of code (LOC) metrics. Among many software metrics, these three categories contain the most widely used metrics. Also in this work, we decided to use an evaluation mechanism mainly based on these metrics.

Researchers have used neural network approach to generate new metrics instead of using metrics that are based on certain polynomial equations (Boetticher et al., 1993). Bayesian belief network is also used to make risk assessment in previous research (Fenton and Neil, 1999). Basic metrics such as LOC, Halstead and McCabe metrics are used in the learning process. There is not a similar relation between the number of errors for the pre- and post-release versions of the software and the cyclomatic complexity. To overcome this problem, Bayesian Belief Network is used for error modeling.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.47 e-ISSN: 2251-7545

B. ERROR PREDICTION AND APPLICATIONS OF MACHINE LEARNING

Error prediction models can be classified according to the metrics used and the process step in the software life cycle. Most of the error models use the basic metrics such as complexity and size of the software (Henry and Kafura, 1984).The main idea behind the prediction models is to estimate the reliability of the system, and investigate the effect of design and testing process over number of errors.

Machine learning algorithms have been proven to be practical for poorly understood problem domains that have changing conditions with respect to many values and regularities. Since software problems can be formulated as learning processes and classified according to the characteristics of error, regular machine learning algorithms are applicable to prepare a probability distribution and analyze errors (Fenton and Neil, 1999; Zhang, 2000). Machine learning algorithms can be used over program execution to detect the number of the faulty runs, which will lead to find underlying errors. Fault relevant properties are utilized to generate a model, and this precomputed function selects the properties that are most likely to cause errors and errors in the software.

Clustering over function call profiles are used to determine which features enable a model to distinguish failures and non-failures (Podgurski et al., 2003). Dynamic invariant detection is used to detect likely invariants from a test suite and investigate violations that usually indicate erroneous state. This method is also used to determine counterexamples and find properties which lead to correct results for all conditions (Groce and Visser, 2003).

III. METRIC DATA USED

The data set used in this research is provided by the BELL TELEPHONE LABS IV&V S1 Program for Real Time Command and Control. The data repository contains software metrics and associated error data at the function/method level. The data repository stores and organizes the data which has been collected and validated by the Metrics Data Program.

The association between the error data and the metrics data in the repository provides the opportunity to investigate the relationship of metrics or combinations of metrics to the software. The data that is made available to general users has been sanitized and authorized for publication through the MDP website by officials representing the projects from which the data has originated. The database uses unique numeric identifiers to describe the individual error records and product entries. The level of abstraction allows data associations to be made without having to reveal specific information about the originating data.

Some of the product metrics that are included in the data set are, McCabe Metrics; Cyclomatic Complexity and Design Complexity, Halstead Metrics; Halstead Content, Halstead Difficulty, Halstead Effort, Halstead Error Estimate, Halstead Length, Halstead Level, Halstead Programming Time and Halstead Volume, LOC Metrics; Lines of Total Code, LOC Blank, Branch Count, LOC Comments, Number of Operands, Number of Unique Operands and Number of Unique Operators, and lastly Error Metrics; Error Count, Error Density, Number of Errors (with severity and priority information).

After constructing our data repository, we have cleaned the data set against marginal values, which may lead our experiments to faulty results. For each type of feature in the database, the data containing feature values out of a range of ten standard deviations from the mean values are deleted from the database.

Our analysis depends on machine learning techniques so for this purpose we divided the data set in two groups; the training set and the testing set. These two groups used for training and testing experiments are extracted randomly from the overall data set for each experiment by using a simple shuffle algorithm. This method provided us with randomly generated data sets, which are believed to contain evenly distributed numbers of error data.

IV. PROBLEM STATEMENT

Two types of research can be studied on the code based metrics in terms of error prediction. The first one is predicting whether a given code segment contain errors or not. The second one is predicting the magnitude of the possible error, if any, with respect to various viewpoints such as density, severity or priority. Estimating the error causing potential of a given software project has a very critical value for the reliability of the project. Our work in this research is primarily focused on the second type of predictions. But it also includes some major experiments involving the first type of predictions.

Given a training data set, a learning system can be set up. This system would come out with a score point that indicates how much a test data and code segment is defected. After predicting this score point, the results can be evaluated with respect to popular performance functions. The two most common options here are the Mean Absolute Error (mae) and the Mean Squared Error (mse). The mae is generally used for classification, while the mse is most commonly seen in function approximation.

In this research we used mse since the performance function for the results of the experiments aims second type of prediction. Although mae could be a good measure for classification experiments, in our case, due to the fact that our output values are zeros and ones we chose to use some custom error measures. We will explain them in detail in the results section.

V. PROPOSED MODEL AND METHODOLOGY

The data set used in this research contains error density data which corresponds to the total number of errors per 1-000 lines of code. In this research we have used the software metric data set with this error density data to predict the error density value for a given project or a module. Artificial neural networks approach is used to predict the error density values for a testing data set.

Multi-layer perceptron method is used in ANN experiments. Multilayer perceptrons are feedforward neural networks trained with the standard backpropagation algorithm. Feedforward neural networks provide a general framework for representing non-linear functional mappings between a set of input variables and a set of output variables. This is achieved by representing the nonlinear function of many variables in terms of compositions of nonlinear functions of a single variable, which are called activation functions (Bishop, 1995).

In the experiments we first applied ANN approach to perform a regression based prediction over the whole data set. According to the experiment results we calculated the corresponding mse values. Mse values provide the amount of the spread from the target values. To evaluate the performance of each algorithm with respect to the mse values, we compared the square root of the mse values with the standard deviance of the testing data set. The standard deviation of the data set is in fact the mse of it when all predictions are equal to the mean value of the data set. To declare that a specific experiment's performance is acceptable, its mse value should be fairly less than the variance of the data set. Otherwise there is no need to apply such sophisticated learning methods, one can obtain a similar level of success by just predicting all values equal to mean value of the data set.

The first experiments that are done using the whole data set show that the performance of both algorithms are not in acceptable ranges as these outcomes are detailed in the results section. The data set includes mostly non-defected modules so there happens to be a bias towards underestimating the error possibility in the prediction process. Also it is obvious that any other input data set will have the same characteristic since it is practically likely to have much more non-defected modules than defected ones in real life software projects.

The three type of experiments explained above guided us in proposing the novel model for error prediction in software projects. According to the results of these experiments, better results are obtained when first a classification is carried out and then a regression type prediction is done over the data set which is expected to contain errors. So the model has two steps, first classifying the input data set with respect to whether it contain errors or not. After this classification, a new data set is generated with the values that are predicted as defected. And a regression is done to predict the error density values among the new data set.

The novel model predicts the possibly modules contain errors in a given data set. So the model helps concentrating the efforts on specific suspected parts of the code so that significant amount of time and resource can be saved in software quality process.

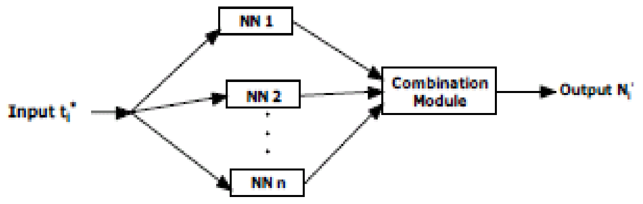

Figure 1. Prediction system

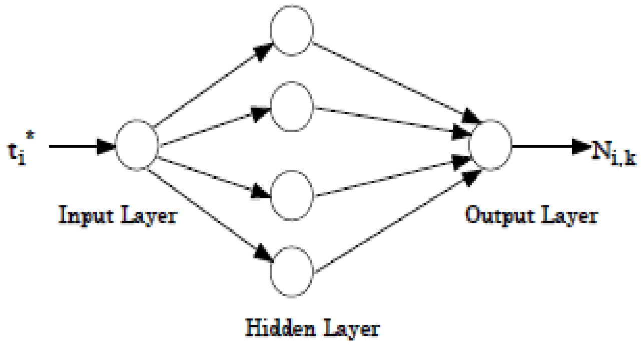

Figure: Architecture of Neural Network

VI. RESULTS

In this research, the training and testing are made using MATLAB's MLP based on a model for classification and regression. The data set used in the experiments contains 6-000 training data and 2-000 testing data. The resulting values are the mean values of 30 separately run experiments.

In designing the experiment set of the MLP algorithm, a neural network is generated by using linear function as the output unit activation function. 32 hidden units are used in network generation and the alpha value is set to 0.01 while the experiments are done with 200 training cycles.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.47 e-ISSN: 2251-7545

A. *REGRESSION(DATA SET)*

The average variance of the data sets which are generated randomly by the use of a shuffling algorithm is 1-402.21 and the mean mse value for the ANN experiments is 1-295.96. This value is far from being acceptable since the method fails to approximate the error density values. Figure 1 depicts the scatter graph of the predicted values and the real values. According to this graph, it is clear that the method potentially does faulty predictions over the non defected values. The points laying on the y-axis show that there are unacceptable amount of faulty predictions for non defected values. Also apart from missing to predict the non defected ones, it is obvious that the method is biased towards smaller approximations on the predictions for defected items because vast amount of predictions lay under the line which depicts the correct predictions.

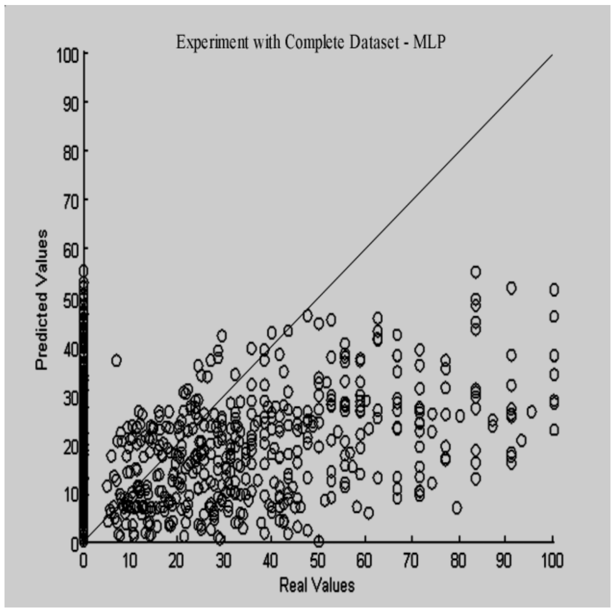

Figure 1. The predicted values and the real values in ANN experiments

B. *REGRESSION OVER THE DATA SET(ONLY DEFECTED ITEMS)*

The second type of experiments are done with input data sets which contain only defected items.

The average variance of the data sets used in the ANN experiments are 1-637.41 and the mean mse value is 262.61. According to these results the MLP algorithm approximates the error density values well when only defected items reside in the input data set. It also shows that the dense non defected data effects the prediction capability of the algorithm in a negative manner. Figure 2 shows the predicted values and the real values after an ANN experiment run. The algorithm estimates the error density value better for smaller values as seen from the graph, where the scatter deviates more from the line that depicts the correct predictions for higher values of error density.

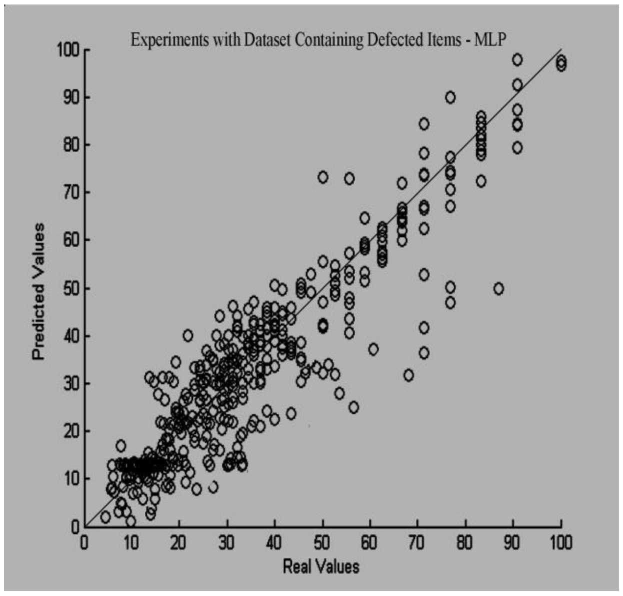

Figure 2. The predicted values and the real values in ANN experiments where the input data set contains only defected items

C. *CLASSIFICATION(DEFECTEDNESS)*

In the ANN experiments the clustering algorithm is partly successful in predicting the defected items. The mean percentage of the correct predictions is 88.35% for ANN experiments. The mean percentage of correct defected predictions is 54.44% whereas the mean percentage of correct non defected predictions is 97.28%. These results show that the method is very successful in finding out the really defected items. It is capable of finding out three out of every four defected items.

As a result, it can be deduced that we divide the error prediction problem into two parts. The first part consists of predicting whether a given module contains errors or not. And the second part is predicting the magnitude of the possible error if it is labeled as defected by the first type. We understand that predicting the error density value among a data set containing only defected items brings much better results than the case that the whole data set is used where an intrinsic bias towards lessening the magnitude of the error arises. Also by dividing the problem into two separate problems, and knowing that second part is successful enough in predicting the error density, it is possible to improve the overall performance of the learning system by improving the performance of the classification part.

VII. CONCLUSION

In this research, we proposed a new error prediction model based on machine learning methods. Since most modules in the input data have zero errors (80% of the whole data), applied machine learning methods fail to predict scores within expected performance. Even if an algorithm claims that a test data doesn't contain errors though it did not try to learn at all, the 80% success is guaranteed. Therefore logic behind the learning methodology fails. Different methodology which can manage such data set for software metrics is required.

By using our two step approach, along with predicting which modules contain errors, the model generates estimations on the error magnitudes. The software practitioners may use these estimation values in making decisions about the resources and effort in software quality processes such as testing. Our model constitutes to a well risk assessment technique in software projects regarding the code metrics data about the project.

As a future work, different machine learning algorithms or improved versions of the used machine learning algorithms like decision trees and neuro-fuzzy systems may be included in the experiments. Also this model can be applied to other risk assessment procedures which can be supplied as input to the system. Certainly these risk issues should have quantitative representations to be considered as an input for our system.

REFERENCES

Bertolino, A., and Strigini, L., 1996. On the Use of Testability Measures for Dependability Assessment, IEEE Trans. Software Engineering, vol. 22, no. 2, pp. 97-108.
Bishop, M., 1995, Neural Networks for Pattern Recognition, Oxford University Press.
Boetticher, G.D., Srinivas, K., Eichmann, D., 1993. A Neural Net-Based Approach to Software Metrics, Proceedings of the Fifth International Conference on Software Engineering and Knowledge Engineering, San Francisco, pp. 271-274.
CHAOS Chronicles, The Standish Group - Standish Group Internal Report, 1995.
Cusumano, M.A., 1991. Japan's Software Factories, Oxford University Press.
Diaz, M., and Sligo, J., 1997. How Software Process Improvement Helped Motorola, IEEE Software, vol. 14, no. 5, pp. 75-81.
Dickinson, W., Leon, D., Podgurski, A., 2001. Finding failures by cluster analysis of execution profiles. In ICSE, pages 339– 348.
Fenton, N., and Neil, M., 1999. A critique of software error prediction models, IEEE Transactions on Software Engineering, Vol. 25, No. 5, pp. 675-689.
Groce, and Visser, W., 2003. What went wrong: Explaining counterexamples, In SPIN 2003, pages 121–135.
Jensen, F.V., 1996. An Introduction to Bayesian Networks, Springer.
Henry, S., and Kafura, D., 1984. The Evaluation of Software System's Structure Using Quantitative Software Metrics, Software Practice and Experience, vol. 14, no. 6, pp. 561-573.
Hudepohl, P., Khoshgoftaar, M., Mayrand, J., 1996. Integrating Metrics and Models for Software Risk Assessment, The Seventh International Symposium on Software Reliability Engineering (ISSRE '96).

Mr. Bonthu Kotaiah obtained his Bachelor's degree in Computer Applications fromNagarjuna University in 2001 and M.C.A from Nagarjuna University in 2008. During the
period from September, 2001 to 2011, he has been involved in various aspects of Information Technology-an engineer (L-Cube Innovative Solutions), a Corporate Trainer (SyncSoft&Datapro (Vijayawada),COSS(Hyd.)), a Programmer (Acharya Nagarjuna University). His research interests include software Engineering, Neural networks, Fuzzy Systems. Presently, he is a Full-Time Research Scholar in Babasaheb Bhimrao Ambedkar University (A Central University) Lucknow, UP in the Department of Information Technology.

Dr. R.A. Khan, Presently Working with Babasaheb Bhimrao Ambedkar University(A Central University) Lucknow, UP as an Associate Professor & Head in the Department of Information Technology, School for Information Science & Technology since December 2006. He is having More than Ten Years of teaching experience He obtained Ph.D. from Jamia Millia Islamia (A Central

University) New Delhi (2004).

MURALIDHAR.V is Post graduated in M.C.A and M.Tech (C.S.E) from Acharya Nagarjuna University- Guntur. He is working as Associate Professor in Computer Science Engineering Department for the last 4 years and also Persuing Research from Acharya Nagarjuna University on Software Reliability. His Area of Interested is Software Engineering, Image Processing, Data Mining and Neural Networks.