Software Defect Prediction System using Multilayer Perceptron Neural Network with Data Mining

Gayathri M, A. Sudha

Abstract -- Fault prediction in software systems is crucial for any software organization to produce quality and reliable software. Faults (defects) or fault-proneness of software modules are to be predicted in the early stages of software life cycle, so that more testing efforts can be put on faulty modules. Various metrics in software like Cyclomatic complexity, Lines of Code have been calculated and effectively used for predicting faults. Techniques like statistical methods, data mining, machine learning, and mixed algorithms, which were based on software metrics associated with the software, have also been used to predict software defects. Many works have been carried out in the prediction of faults and fault-proneness of software systems using varied techniques. In this paper, an enhanced Multilayer Perceptron Neural Network based machine learning technique is explored and a comparative analysis is performed for the modeling of fault-proneness prediction in software systems. The data set of software metrics used for this research is acquired from NASA's Metrics Data Program (MDP).

Keywords -- Faults, Fault-proneness, Software Metrics, Software Defect Prediction, Multilayer Perceptron Neural Network.

I. INTRODUCTION

Software Defect Prediction plays a vital role in the field of software quality and software reliability. A software **fault** is an error, flaw, mistake, failure, or defect in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways. A software module is said to be **fault-prone** if it contains a large number of faults that seriously interfere with its functionality. **Software Defect Prediction (SDP)** is the process of locating defective modules in software. Code review, unit testing, integration testing and system testing are the traditional process for identifying defects. However, when projects' size grows in terms of both lines of code and complexity, finding and fixing faults gets more difficult and computationally expensive with the use of sophisticated testing and evaluation procedures. Also, Boehm observed that finding and fixing a problem after delivery is more expensive, in terms of cost and effort, than fixing it during the early stages of software life cycle. 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.

Software Defect Prediction models helps to tackle these problems. SDP models deals with predicting low-quality areas of the software product, thereby assisting the design and testing team in focusing their quality improvement tasks. Thus, SDP models helps to ensure reliability of the delivered products. SDP models generally focuses on either estimating the number of defects in software modules or classifying the fault-proneness of software components into two classes, namely fault-prone (fp) and not fault-prone (nfp). **Software metrics** refers to software measurement data that are collected at various phases of the software development life cycle from initiation to maintenance. Traditionally software metrics are used for the purposes of product quality and process efficiency analysis. Detailed analysis of software metric data gives a good indication of possible defects in the software being developed. For instance, complexity and size metrics have been used in an attempt to predict the number of defects a system will reveal during operation or testing. Statistical methods like CART, Poisson regression, binomial regression, logistic regression, etc. are also employed in software defect analysis. Another approach for defect prediction is the use of data mining techniques (Classification, Prediction, Clustering, and Association Rules) to predict the problematic areas in the software. Apart from the data mining techniques described above, researchers have recently started investigating the application of machine learning techniques (e.g. Artificial Neural Networks, Decision Trees, Naive Bayes, etc.) for predicting software quality. Researchers have also investigated methods to combine machine learning and statistical methods. For example, statistical methods like Principal Component Analysis and Feature Subset Selection can be used to enhance the performance of neural networks. 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 defect, regular machine learning algorithms are applicable to analyze errors [10]. Neural networks, based on machine learning approach, are non-linear sophisticated modeling techniques that are able to model complex functions. The aim of this work is to explore Multilayer Perceptron Neural Network (MLP-NN), for the modeling of fault-proneness prediction in software systems. The comparison of different algorithms is made on the basis of Root Mean Squared Error (RMSE) and Accuracy Values. The data set used in this research is acquired from NASA's Metrics Data Program – Metric Data Repository. 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 remainder of the paper is structured as follows: Section 2

Manuscript Received on May 2014.

Gayathri M, *II year student, M.E(CSE) Department of CSE, Al-Ameen Engineering College, College,Erode, Tamilnadu, India.*

A. Sudha, *Assistant Professor, Department of CSE, Al-Ameen Engineering Erode, Tamilnadu, India.*

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

discusses various proposals made in the literature for software defect prediction. Section 3 gives research background and description of techniques that are used in the study. Section 4 deals with the methodology explored in this study. Section 5 describes about the data set used in this work. Section 6 elaborates the proposed model. Section 7 explains the methods employed to compare algorithms. Section 8 analyses the results and the last section concludes the study.

II. LITERATURE REVIEW

Predicting defective software modules is of great interest among the software quality researchers and industry professionals. As a result of this, various efforts have been made for software fault prediction using varied methods. Rachna et al. [2] compares Hierarchical Clustering Technique and k-NN Neural Network which were applied in classifying software components into fault-prone or not fault-prone. It is found that the performance is better in case of neural network approach as compared to clustering based approach. Azeem et al. [3] discusses different data mining techniques for identifying fault prone modules as well as compares the data mining algorithms to find out the best algorithm for defect prediction. The study suggests that selection of better data mining algorithm depends on various factors like problem domain, type of data sets, nature of project, uncertainty in data set etc. Song et al. [4] propose and evaluate a general framework for software defect prediction that supports unbiased and comprehensive comparison between competing prediction systems. The results show that we should choose different learning schemes for different data sets (i.e., no scheme dominates).Yi Liu et al. [5] investigated the problem of software quality modeling using the metric dataset obtained from single software project, which is not adequate to build robust and an accurate model. To solve this problem, software quality modeling was done using multiple datasets sourced from different software projects. As the majority of faults are found in few of the modules, there is a need to investigate the modules that are affected severely as compared to other modules and proper maintenance has to be done on time especially for the critical applications. Ebru Ardil et al. [6] investigated the above problem using feed forward neural network. Lessmann et al. [7] proposed a framework for comparative software defect prediction experiments, which is applied to compare 22 classifiers over 10 public domain data sets from the NASA Metrics Data repository. High degree of predictive accuracy is observed in a SDP experiment which used metric-based classification. Catal et al. [8] modeled Artificial Immune System based on the Human immune system for defect prediction. The proposed classifier imitates the behavior of the antigen and the antibody during an attack by pathogens into the human biological system. The evolution of the immune system to new attacks is modeled to solve the software defect prediction problem. Challagulla et al. [9] evaluate different software defects predictor models on four different real-time software defect data sets. The results show that a combination of 1R and Instance-based Learning along with the Consistency-based Subset Evaluation technique provides a relatively better consistency in accuracy prediction compared to other models. Classification is one of the data mining problems receiving great attention recently in the database community. Lu et al. [11] presents an approach to discover symbolic classification rules using neural networks.

III. BACKGROUND AND DESCRIPTIONS

A software module has a series of metrics, some of which are related to its' fault-proneness. The relationship between software metrics and fault-proneness of the modules has been researched extensively over a long period of time. Our study exploits the said relationship, to build a SDP model to classify software modules into 'fp' and 'nfp'. Here, we provide a description of techniques that are used in the study.

A. Data Mining and Machine Learning Techniques

Data mining techniques and machine learning algorithms are useful in prediction of software defects. These techniques can be applied on the software repositories, to extract the defects of a software product. Knowledge Discovery in Databases (KDD) comprises of many steps namely, data selection, data preprocessing, data transformation, data mining and data interpretation and evaluation. Data mining forms a core activity in KDD. The above steps of KDD are employed by experts while building SDP models. Data mining entails the process of extracting knowledge from huge volume of data. Data mining technique comprises of classification, regression, clustering and association. The focus here is on classification technique, which is the task of classifying the data into a predefined class according to its predictive characteristics. In this work, software modules are classified into fp and nfp, based on software metrics. Machine learning methods are employed in defect prediction models to learn and predict potentially defected modules within the software. The software metric data of modules or software combined with defect data forms the input of the machine learning algorithms. A learning system is defined as a system that learns from experience, with respect to some class of tasks and performance measure, such that its' performance at these tasks, improve with experience. 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 training data set and the results are evaluated with the testing data set. The machine learning techniques popularly used for software defect prediction problems are decision trees, neural network and Bayesian belief network. Various machine learning approaches such as supervised, semi-supervised, and unsupervised have been used for building fault prediction models. Among these, supervised learning approach is widely used for predicting fault-prone modules. This study employs neural networks and uses supervised learning approach for software defect prediction.

B. Neural Networks

An Artificial Neural Network (ANN) or simply a Neural Network (NN) is an information-processing paradigm that is inspired by the way a biological nervous system in human brain works. Large number of neuron, present in the human brain form the key element of the neural network paradigm and act as elementary processing elements. These neurons

Published By:
Blue Eyes Intelligence Engineering &
Sciences Publication Pvt. Ltd.

are highly interconnected and work in union to solve complex problems. An artificial neuron is a small processing unit and performs a simple computation that is fundamental to the operation of a neural network. The model of a neuron contains the basic elements like inputs, synaptic weights and bias, summing junction and activation function. In a broader perspective, ANN can be divided into two major categories based on their connection topology: Feed forward and Feed backward neural networks. Feed-forward neural networks allow the signal to flow in the forward direction only. The signals from any neuron do not flow to any other neuron in the preceding layer. In Feed backward neural networks the signal from a neuron in a layer can flow to any other neuron whether it be preceding or succeeding layers. However, the drawback with these networks is that these are complex and are difficult to implement.

1. Multilayer Feed Forward Neural Networks

Multilayer feed forward neural networks (MLF-NN), consists of multiple layers of computational units, usually interconnected in a feed-forward way. The first layer is called the input layer, the last layer is called the output layer, and the layers between are hidden layers. Each neuron in one layer has directed connections to the neurons of the subsequent layer. MLF-NNs are non-parametric regression methods, which approximate the underlying functionality in data by minimizing the loss function. During training, specified items of data records are put as the input of neural network and its weights are changed in such a way that its output would approximate the values in the data set. After finishing learning process, the learned knowledge is represented by the values of neural network weights. For training, the algorithm of back-propagation of error is often used. Figure 1 depicts the structure of this class of neural network.

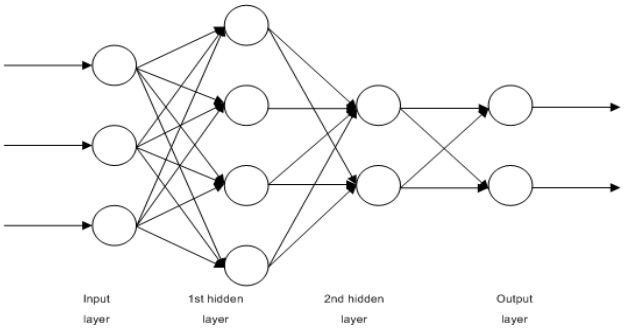

Fig. 1 Multilayer Feed Forward Neural Network

General Defect Prediction Process

To construct a defect prediction model, we must have defect and measurement data collected from actual software development efforts to use as the learning set. There exist compromise between how well a model fits to its learning set and its prediction performance on additional data sets. Therefore, we should evaluate a model's performance by comparing the predicted defectiveness of the modules in a test set against their actual defectiveness. Kim et al. [1] have described a common defect prediction process shown in Fig. 2.

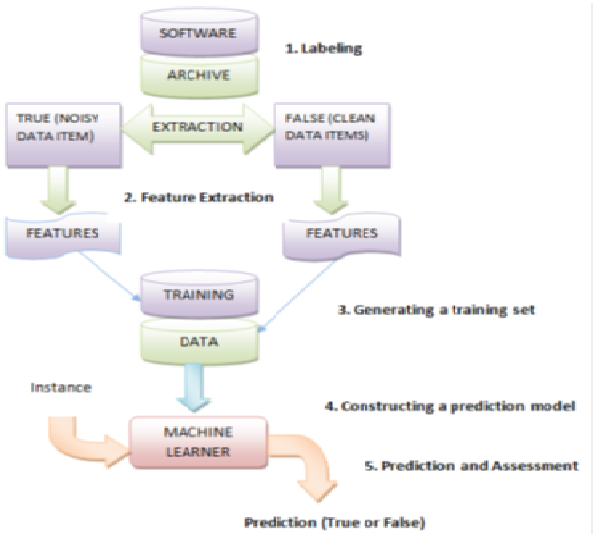

Fig.2 General Defect Prediction Process

General defect prediction process has the following steps:

Labeling: Defect data should be gathered for training a prediction model. In this process usually extracting of instances i.e., data items from software archives and labeling (TRUE or FALSE) is done.

Extracting features and creating training sets: This step involves extracting of features for prediction of the labels of instances. General features for defect prediction are complexity metrics, keywords, changes, and structural dependencies. By combining labels and features of instances, we can produce a training set to be used by a machine learner to construct a prediction model.

Building prediction models: General machine learners such as Neural Networks (NN) or Bayesian Network can be used to build a prediction model by using a training set. The model can then obtain a new instance and predict its label, i.e. TRUE or FALSE.

Assessment: The evaluation of a prediction model requires a testing data set besides a training set. The labels of instances in the testing set are predicted and the prediction model is evaluated by comparing the prediction and real labels. 10-fold cross-validation is broadly used to separate the training and testing sets.

IV. METHODOLOGY

Multilayer Perceptron Neural Networks are feed forward neural networks trained with the standard back-propagation algorithm. Multilayer Perceptrons (MLPs) can be trained to learn to transform input data into a preferred response, and are widely used for modeling prediction problems. The MLP is an example of a supervised learning artificial neural network that is used extensively for the solution of a number of different problems, including pattern recognition, identification, classification, vision, speech, control systems, etc. In back-propagation, the actual output values are compared with the desired output to compute the value of some predefined error-function. The algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

would say that the network has learned a certain target function. To adjust weights properly, one applies a general method for non-linear optimization that is called gradient descent. For this, the derivative of the error function with respect to the network weights is calculated, and the weights are then changed such that the error decreases. The error computed is the squared Euclidean distance between the actual output of the network and the desired output. The algorithm for multilayer perceptron neural network is given below:

i. Present input and desired output

Present input Y*p = y0 ,y1 ,y2 ,...,yn-1* and target output ***C****p = c0 ,c1 ,...,cm-1* where n is the number of input nodes and m is the number of output nodes.

ii. Calculate the actual output

Each layer calculates, *fxpj = f* [*w0y0 + w1y1 + ... + wyn*]*j*. This is then passed to the next layer as an input. The final layer outputs values *opj*.

iii. Adapts weights, starting from the output we now work backwards.

wij(t+1*) = wij(t) + ñþpjopj*, where *ñ* is a gain term and *þpj* is an error term for pattern *p* on node *j*.

For output units, *þpj = kopj*(*1 - opj*)(*t - opj*)

For hidden units, *þpj = kopj*(*1 - opj*)[(*þp0wj0 + þp1wj1 + ...+þpkwjk*)], where the sum is over the *k* nodes in the layer above node *j*.

Feed forward 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. The hidden layer of a MLP neural network typically consists of sigmoid function. The proposed MLP neural network consists of two hidden layers, with the first layer employing a tanh activation function and the second layer implementing fuzzy bell-shaped activation function.

The fuzzy bell membership function is given by

$$F(x) = \frac{1}{1+\left(\frac{x-w_2}{w_0}\right)^{2w_1}}$$ where x is the input and w_i is the weight.

The tanh member applies a bias and tanh function to each neuron in the layer. The tanh will squash the range of each neuron between -1 and 1. Such nonlinear elements provide a network with the ability to make soft decisions. The tanh activation function is given by

$$\tanh(i) = \frac{(e^{i})-(e^{-i})}{(e^{i})+(e^{-i})}$$ where i is the sum of the input patterns.

V. DATA SET

This study makes use of KC1 data set which is obtained from the NASA's Metric Data Program (MDP) data repository. KC1 is a project that is comprised of logical groups of computer software components (CSCs) within a large ground system. KC1 is made up of 43,000 lines of code, coded in C++. The data set contains 2,107 instances (modules), and of these instances, 325 have one or more faults and 1,782 have zero faults. The maximum number of faults in a module is seven. Different types of predictor software metrics of KC1 data set that are used in our analysis are listed in the Table 1. Defect prediction models have independent variables captured in the form of product and process metrics and one dependent variable which indicates whether there could be a fault or no fault in the module. In our set of data, class_label is the dependent variable and the rest are independent variables.

Table I – Software Metrics Inside KC1 Project

Metrics	Description
LOC_blank (BLOC)	No. of blank lines in a module
LOC_code_and_comment (CCLOC)	No. of lines which contain both code & comment in a module
LOC_comments (CLOC)	No. of lines of comments in a module
LOC_executable (ELOC)	No. of lines of executable code for a module (not blank or comment)
LOC_total (TLOC)	Total no. of lines for a given module
cyclomatic_complexity (CC)	Cyclomatic complexity of a module
design_complexity (DC)	Design complexity of a module
essential_complexity (EC)	Essential complexity of a module
Halstead_Content (I)	Halstead intelligent content of a module
Halstead_Difficulty (D)	Halstead difficulty metric of a module
Halstead_Effort (E)	Halstead effort metric of a module
Halstead_Error_Est (B)	Halstead error estimate metric of a module
Halstead_Length (N)	Halstead length metric of a module
Halstead_Level (L)	Halstead level metric of a module
Halstead_Prog_Time(T)	Halstead programming time metric of a module
Halstead_Volume (V)	Halstead volume metric of a module
num_operands (NOD)	No. of operands contained in a module
num_operators (NOT)	No. of operators contained in a module
num_unique_operands (UNOD)	No. of unique operands contained in a module
num_unique_operators (UNOT)	No. of unique operators contained in a module
branch_count (BC)	No. of branches of flow graph for each module
class_label (Target Metric)	Indicates that a module has/has not one or more faults with the values {false,true}

VI. PROPOSED MODEL

The proposed MLP neural network model is a modification of existing MLP-NN model, with the introduction of fuzzy

Published By:
Blue Eyes Intelligence Engineering &
Sciences Publication Pvt. Ltd.

logic based bell-shaped function in its hidden layer exploiting the advantages of membership function. The proposed model consists of two hidden layers with the first layer being a tanh activation function and the second layer containing the fuzzy bell activation function. Fig. 3 depicts the block diagram of the proposed model architecture. The enhanced MLP neural network proposed in this paper uses the criteria specified in Table II.

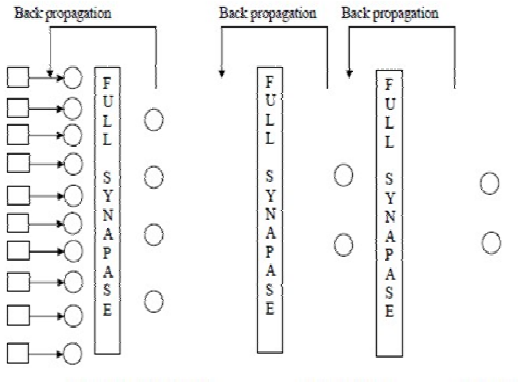

Fig. 3 Proposed Model Architecture

The main advantage of fuzzy logic is that it mimics human decision making to handle vague concepts and ability to deal with imprecise or imperfect information. But it needs experts for rule discovery (data relationships). The major disadvantages are lack of self-organizing & self-tuning mechanisms. The main advantage of neural networks is that there is no need to know data relationships. It has self-learning capability and self-tuning capability. The limitations faced by neural network include that it cannot manage imprecise or vague information. Neuro-fuzzy refers to the combination of fuzzy set theory and neural networks with the advantages of both.

Table II Design Metrics of The Proposed MLP-NN Model

Parameters	Values
Input Neuron	20
Output Neuron	1
Number of Hidden Layer	2
Number of processing elements – first layer	6
Transfer function of first hidden layer	tanh
Learning rule of second hidden layer	momentum
Number of processing elements-second layer	2
Transfer function of second hidden layer	fuzzy bell-shaped
Learning Rule of hidden layer	Momentum

VII. ANALYSIS OF RESULTS

The proposed Fuzzy Bell MLP neural network algorithm was implemented using Visual Studio. The data set discussed in section 5 is used to train and test the SDP system. Classification was done using 65 percent of the data as the training set and the remaining as the test set. The classification accuracy obtained on KC1 dataset is 98.2%.

The result obtained by our proposed methodology is improved over the multilayer perceptron model with sigmoid hidden function by 3.92%. The proposed fuzzy based neural model was able to classify better than other existing techniques like Random Tree, CART and Bayesian logistic regression. Table III gives the classification accuracies and RMSE of the methods investigated. Fig. 4 shows the chart which depicts the classification accuracy attained by existing and proposed methods.

Table III – Results of Investigated Methods

Method	% Correctly classified	RMSE
Random tree	94.55	0.43
Logistic Regression	95.67	0.37
CART	96.79	0.35
Existing MLP-NN	94.28	0.34
Proposed MLP-NN	98.22	0.29

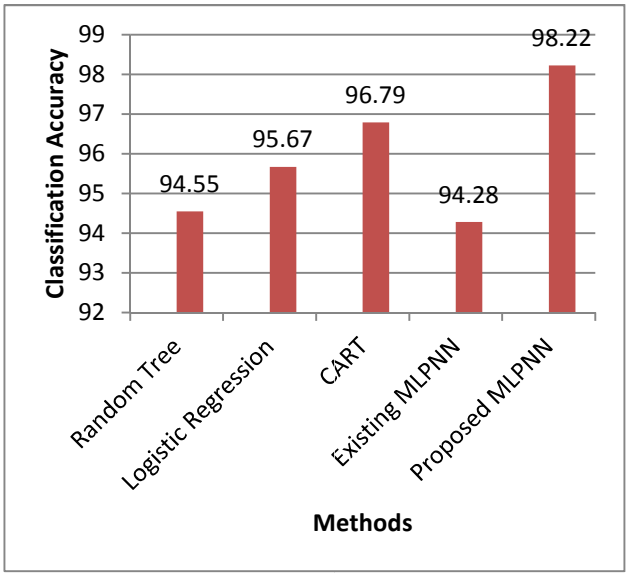

Fig. 4 Classification Accuracy of Existing and Proposed Methods

VIII. CONCLUSION

Software defect prediction is used to improve software process control and achieve high software reliability. By analyzing the results it is clear that the proposed MLP neural network model gives the better result, in predicting software defects, when compared with the other methods previously studied. However the proposed method needs to be evaluated with other datasets to better test the performance in terms of consistency. Thus, it can be concluded that the proposed algorithm, bell function based MLP neural network, is the best for modeling fault-proneness prediction in software systems.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

REFERENCES

[1] *Sunghun Kim, Hongyu Zhang, Rongxin Wu and Liang Gong, 'Dealing with Noise in Defect Prediction', CSE'11, Waikiki, Honolulu, HI, USA, ACM 978-1-4503-0445-0/11/05, 2011.*

[2] *Rachna Ratra, Navneet Singh Randhawa, Parneet Kaur, and Dr. Gurdev Singh, 'Early Prediction of Fault Prone Modules using Clustering Based vs. Neural Network Approach in Software Systems', IJECT Vol. 2, Issue 4, Oct. - Dec. 2011*

[3] *Naheed Azeem and Shazia Usmani, 'Analysis of Data Mining Based Software Defect Prediction Techniques', Global Journal of Computer Science and Technology Volume 11 Issue 16 Version 1.0 September 2011*

[4] *Qinbao Song, Zihan Jia, Martin Shepperd, Shi Ying, and Jin Liu, 'A General Software Defect-Proneness Prediction Framework', IEEE Transactions on Software Engineering, Vol. 37, No. 3, May/June 2011*

[5] *Yi (Cathy) Liu, Member, IEEE Computer Soceity, Taghi M. Khoshgoftaar, Member, IEEE, and Naeem Seliya, Member, IEEE, 'Evolutionary Optimization of Software Quality Modeling with Multiple Repositories', IEEE Transactions On Software Engineering, Vol. 36, No. 6, November/December 2010, pp 852-864*

[6] *Ebru Ardil, Erdem Ucar, and Parvinder S. Sandhu, 'Software Maintenance Severity Prediction with Soft Computing Approach', Proceedings of World Academy of Science, Engineering and Technology (2009), 38, pp. 139-143.*

[7] *Stefan Lessmann, Bart Baesens, Christophe Mues, and Swantje Pietsch, 'Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings', IEEE Transactions On Software Engineering, Vol. 34, No. 4, July/August 2008 pg : 485*

[8] *Cagatay Catal and Banu Diri, 'Software Defect Predicition Using Artificial Immune Recognition System', Proceedings of the 25th IASTED International Multi – Conference Software Engineering (2007), Innsbruck, Austria, ISBN Hardcopy:978-0-88986-641- 61/CD:978-0-88986-643-0.*

[9] *Venkata U.B. Challagulla, Farokh B. Bastani, I-Ling Yen, and Raymond A. Paul, 'Empirical Assessment of Machine Learning based Software Defect Prediction Techniques', Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (2005).*

[10] *Fenton N.E. and Neil M., 'A critique of software defect prediction models', IEEE Transactions on Software Engineering, Volume: 25 Issue: 5, Sept. - Oct. 1999, Page(s): 675 -689.*

[11] *Hongjun Lu, Member, IEEE Computer Society, Rudy Setiono and Huan Liu, Member, IEEE, 'Effective Data Mining Using Neural Networks', IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, December 1996*

Mrs. Gayathri M. obtained Master of Computer Applications from Bharathiyar University, Tamilnadu. She is pursuing Master of Engineering in Computer Science and Engineering in Al-Ameen Engineering College (Anna University), Erode. Her research interests are Software Reliability Engineering and Neural Networks.

Mrs. A. Sudha obtained Master of Technology in Computer Science and Engineering from VIT University, Tamilnadu. She is currently working as an Assistant Professor in the Department of Computer Science and Engineering, Al-Ameen Engineering College, Erode. Her research area includes Data Mining and Machine Learning.

Published By:
Blue Eyes Intelligence Engineering &
Sciences Publication Pvt. Ltd.