International Journal of Advanced Technology in Engineering and Science
Vol. No.4, Issue No. 02, February 2016
www.ijates.com
ijates
ISSN 2348 - 7550

A GENETIC ALGORITHM OPTIMIZED MULTI-LAYER PERCEPTRON FOR SOFTWARE DEFECT PREDICTION

V.Jayaraj[1], N. Saravana Raman[2],

[1]*Bharathidasan University, Trichy, Tamil Nadu (India)*

[2]*Research Scholar, Bharathidasan University Trichy, Tamil Nadu (India)*

ABSTRACT

Predicting the defects in software is one of the significant issues in software engineering that contributes a considerable measure toward sparing time in software generation and maintenance process. Essentially, discovering the optimal models for Software Defect Prediction (SDP) has these days transformed into one of the primary objectives of software architects. Since details and restrictions of software development are growing and negative outcomes like, failure and errors diminish software quality and consumer loyalty, delivering error-free software is exceptionally hard. In this work, a Multi-Layer Perceptron (MLP) neural network is proposed to classify the software defects. To enhance the performance of the proposed MLP, the parameters of the neural network is optimized utilizing Genetic Algorithm (GA).

Keywords: Back-propagation (BP), Genetic Algorithm (GA), Multi-Layer Perceptron (MLP), Software Defect Prediction (SDP)

I. INTRODUCTION

Software Defect Prediction (SDP) assumes a fundamental part for software quality and software reliability. Software fault is an error, flaw, failure, defect or mistake,in a computer program that delivers an erroneous or unforeseen result, or makes it carry on in unintended ways. A software module is said to be fault-inclined in the event that it contains a substantial number of faults that genuinely interfere with its functionality. SDP locates defective modules in software. Unit testing, integration testing, code review, and system testing are the customary process for distinguishing defects.

A software defect is an error, in a computer program that may generate an erroneous or surprising result, or precludes the software from carrying on as expected. A project team dependably tries to reproduce a quality software item with zero or little defects. High hazard segments inside of the software project ought to be gotten as quickly as time permits, with a specific end goal to upgrade software quality. Software defects dependably cause cost regarding quality and time. In addition, finding and correcting defects is a time expending and costly software processes [1].

Multilayer Perceptron (MLP) neural networks trained with Back-Propagation (BP) algorithm have been helpful in comprehending a wide assortment of real world issues in different domains. In spite of various expansions and adjustments, for example, the increasing speed of the convergence speed (i.e. fast-back-propagation), distinct learning rules and data representation plans, diverse error functions, alternate transfer (activation) functions of the neurons, weight distribution (i.e. weight pruning) among others, to enhance the results or to accomplish some required properties of the trained networks, one key element i.e., the BP, still taking into account the gradient descent algorithm to minimize the network error, has been hardly changed.

Various endeavours have been made to prevent the gradient descent algorithm from becoming trapped in local minima as the training of the network advances. Genetic Algorithms (GAs) normally evade local minima via searching in a numerous areas concurrently (working on a population of trial solutions). The main data that GAs need is some performance value that decides how great a given set of weights is. They have no requirement for gradient data. GAs additionally put no limits on the network topology as they don't require backward propagation of an error signal [2].

Neural Networks aren't generally utilized as a part of credit scoring because of two principle reasons,

- The trouble with interpretability and
- The complexity in model development

While building up a MLP Neural Network, analysts need to address a few architectural issues. An optimization of architecture of the MLP Neural Network is made utilizing an implementation of GA in the SAS system. The objective function to improve is the Receiver Operating Characteristics (ROC) curve and the decision variables are the quantity of hidden layers and their activation function, the quantity of hidden units in every layer, the activation function of the target layer, and whether or not to utilize bias, or to have connection between the input and output layer [3].

Perceptron was the generic name given to a group of theoretical and experimental artificial neural net models by Frank Rosenblatt. Rosenblatt's work formed much eagerness, argument and awareness, for neural net models for pattern classification in that period and prompted critical models abstracted from his work in later years. As of now Perceptron (single-layer) and MLP are utilized to allude to particular artificial neural network structures based on Rosenblatt's perceptron.

In Artificial Neural Networks (ANN)comprises of a progression of nodes (neurons) which have multiple connections with different nodes. Every connection has a weight connected with it which can be changed in strength, in similarity with neurobiology synapses. An ordinary, ANN architectureis known as MLP. The principalon how a neural network works is generally basic. Every neuron in the input layer holds a value, so that the input layer holds the input vector. Each of these neurons interfaces with each neuron in the next layer of neurons [4].

Most broadly utilized neural network model for classification is MLP in light of one or all the more consecutively joined layers of perceptron. MLP model considered in this paper fits in with the feed forward neural networks. In classification to start with, the network is trained on an arrangement of paired data to advance an arrangement of connection weights and then the network is prepared to test new set of data. The most common and broadly utilized training algorithm to estimate the values of the weights is the Back-Propagation (BP) algorithm, which follows the principle the guideline of gradient descent procedure.

Genetic algorithms (GAs) based learning is utilized to discover near-optimal solutions globally from search space without computing gradient data. Original GAs use binary string vector representation like the chromosome structure of biology. However, binary GA hasdrawbacksas its learning primary objective is to enhance accuracy of the network and no thought is given to the speed of convergence and local error. Here, the emphasis is to first apply GA learning and next the gradient descent algorithm, inMLP to optimise the best connection weights and minimize local errors with fast convergence [5].

GAs would like to imitate what nature does, and in this manner get a powerful optimization algorithm for the calculation of the "global" optimum of a given function. Theoretically the gene-pool is the solution space, nature is the function to be optimized (the objective or cost function), and the organisms are the trial functions (individuals) used to work out a solution. Having added to this similarity, one needs to recognize the mechanics of natural evolution and afterward make an interpretation of them into something scientifically concrete. This results in a powerful optimization technique [6].

The parameters of the MLP are optimized utilizing GA as a part of this work.Section 2 deals with literature related to this work, section 3 reveals the methods used in the work, section 4 deals with results and discusses obtained results and finally section 5 concludes the work.

II. LITERATURE REVIEW

Syed Mustafa and Swamy [7] proposed MLP optimized with Tabu search (MLP-TS) for learning. Experimental results exhibited that the proposed MLP-TS is better than Multi-Layer Perceptron-Levenberg-Marquardt (MLP-LM) and Multi-Layer Perceptron Back Propagation (MLP-BPP) for classification.

Mirjalili and Sadiq [8] utilized Magnetic Optimization Algorithm (MOA) as another training strategy for MLP to enhance the previously stated inadequacies. The proposed learning strategy was compared with Particle Swarm Optimization (PSO) and GA-based learning algorithms utilizing 3-bit XOR and function approximation benchmark issues. The results demonstrated the better performancefor this new learning algorithm for extensive quantities of training samples.

Ikuta et al., [9] proposed the MLP with the two distinctive pulse glial networks. The proposed MLP has the glial network which is inspiredfrom biological functions of the glia. These impacts are propagated into the networks. The glial impact gets to be complex, and influences the MLP learning performance. By the computer simulation, the authors affirm that the learning performance of the proposed MLP is superior to the ordinary MLP.

Yang et al., [10] concentrated on whether directly optimizing the model performance measure could assistSDP model development. The work included two viewpoints: one is a novel utilization of the learning-to-rank way to deal with real-world data sets for SDP, and the other is a comprehensive assessment and correlation of the learning-to-rank strategy against different algorithms that have been utilized for predicting the order of software modules as per the predicted number of defects.

Okumoto [11] presented a SDP model utilizing defect data from stability test. The authors showed that test run term in hours is a superior measure than time in days for predicting the quantity of defects in a software release. An exponential reliability development model is connected to the defect data as for test run length of time.

Can et al., [12] proposed another model for SDP utilizing PSO and Support Vector Machine (SVM) named P-SVM model which exploited non-linear computing capacity of SVM and parameters optimization ability of PSO. P-SVM model utilized PSO algorithm to compute the best parameters of SVM, and after that it received the optimized SVM model to predict software defect. P-SVM model and other three diverse prediction models are utilized to predict the software defects in JM1 data set as a test; the results demonstrated that P-SVM model’s prediction accuracy is higher than SVM model, GA-SVM model, BP Neural Network model,.

Pelayo and Dick [13] researched two noteworthy stratification choices (under-, and over-sampling) for SDP utilizing Analysis of Variance. The analysis covered a few current SDP datasets utilizing a factorial design. The authors find that the fundamental impact of under-sampling is noteworthy at a = 0.05, as is the interaction between under-and over-sampling. The principle impact of over-sampling is not noteworthy.

Gayathri and Sudha [14] investigated an improved MLP Neural Network based machine learning strategy and a comparative analysis was performed for the modelling of fault-inclination prediction in software systems. The data set of software measurements utilized for this research was procured from NASA's Metrics Data Program (MDP).

III. METHODOLOGY

The reports depend on datasets got from the NASA publicModular toolkit for Data Processing (MDP) repository. This is an open repository for NASA datasets. NASA datasets are made up of many static code attributes. Eight datasets are chosen from those accessible in the repository which will be utilized as a part of the study. Each dataset depicts the attributes of every project properties, for example, size, number of modules, and the quantity of defects. All data sets are accessible in the extension ".arff" extension to empower processing and analysing data sets by means of a data mining toolcalled Weka. Numerous classifiers are executed in openly accessible machine learning toolkitWeka. A few data pre-processing steps are directed to enhance data sets quality, which will in the end build performance of the predictor [15].

3.1 Multi-Layer Perceptron (MLP)

MLP neural network is a kind of feed-forward neural network. This network incorporates three layers: input, hidden and output. The quantity of cells every layer can have is controlled by trial and error technique. In MLP neural networks, every neuron in a layer is connected with the previous layer's all neurons. Such networks are called 'completely related' networks. Fig. 1 demonstrates a perceptron networks with 2 hidden layers and three neurons in each hidden layer and a neuron in the output layer.

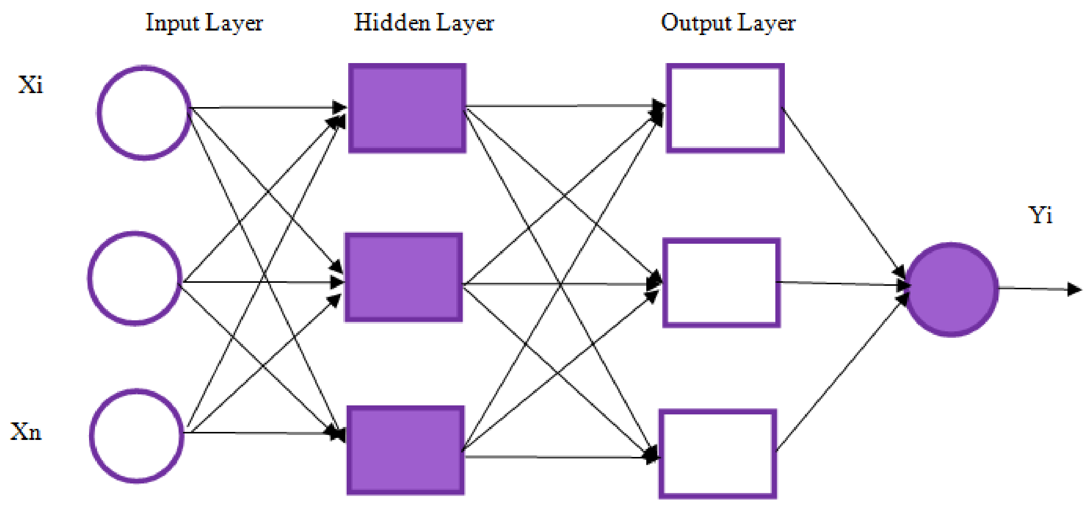

Figure 1 Multilayer Perceptron Network

The input layer is transformer and a device for preparing the data. The last layer-output layer-incorporates the values predicted by the network and registers model output. The centre layers-hidden layers-that are shaped by calculator neurons are the place the data is processed. Network output is given as (1):

$$Y_i = f_i\left(\sum_{j=1}^{n} X_j W_{ij} + b_i\right) \qquad (1)$$

Where Y_i represents network output, X_i looks like network input, W_{ij} is the connection weights amid input and output nodes, B_i is bias and F_i is the transfer function [16]

3.2 GA Optimized MLP

GA Optimized MLP is an algorithm that searches over initial weights, and hidden layer size of MLP, based on GA and BP. It will be demonstrated that it gets preferred results over BP alone, and only the learning constant has to be set by hand (it clearly needs to set GA constants, yet is sufficiently powerful to get great results under the default parameter settings). To utilize both algorithms: the aptitude of the GA to discover a solution near the global optimum, and the aptitude of the BP to tune a solution and achieve the closest local minimum by method for local search from the solution found by the GA. Rather than utilizing a pre-built up topology, the population is introduced with distinctive hidden layer sizes, with some particular operators intended to change it. As genetic operators, mutation, multi-point crossover, addition, elimination and substitution of hidden units have been chosen. Hence, the GA searches and optimizes the architecture (number of hidden units) and the initial weight setting for that architecture. Not like the different methodologies, the hidden layer size is not limited ahead of time.

- A GA is applied directly to a population of MLP, rather than some codification of the network. In this manner a lineal or binary representation is not required to evolve the population.
- GA is utilized to adjust the initial weights, while BP is utilized to train from those weights. This makes a perfect division amid global and local search.
- The hidden layer size is searched all through the application of some new GA operators: substitution, addition and elimination. These operators are created to perform incremental (including hidden neurons) or decremental (pruning hidden neurons) learning.
- The program does not need more time than BP to locate a superior solution [17].

Genetic algorithms are an iterative process, with cycle of a solution or a few solutions work. GA search begins with an introductory random population of solutions. In the event that the criteria aren't fulfilled three diverse methods, Mutation, crossover, selection, are utilized to upgrade the population. Every emphasis of these three methods is perceived starting a generation. Since solutions in Genetic Algorithms are like a characteristic of chromosome and genetic operators are like the GA operators, these algorithms are called genetic algorithms [18].

GA is an optimization system attempting to imitate normal evolution where individuals with best qualities adjusting to the environment are prone to reproduce and survive. Such profitable individuals mate among themselves and produce offspring with comparable qualities subsequently preserving positive attributes while those unfavourable are demolished, prompting species dynamic evolution.

GA iterates and develops a population shaping another population at each step. GA cycle incorporates the accompanying steps:

1. **Selection**: The initial step chooses individuals/chromosomes for reproduction. Fitness value is essential in selection and completely randomly. Individuals having better fitness values are picked regularly for reproduction.
2. **Reproduction**: Offspring are produced by chosen individuals. Recombination and mutation methods generate new chromosomes.
3. **Evaluation**: Fitness of new chromosomes is assessed.
4. **Replacement**: Here, old population individuals are evacuated and interchanged by new ones.

Besides mutation rate and number of crossing points, every operator's application needs to show number of individuals generated by genetic operators. Crossover operator does multipoint cross-over amid two chromosome sets, to obtain 2 networks whose hidden layer neurons have a combination of both the parents hidden layer neurons[19].

IV. RESULTS AND DISCUSSION

In this section, we present the classification accuracy, precision, recall, F measure results achieved.The KC1 Dataset is used for the performance evaluation of the proposed technique; 2107 samples was used of which 1391 samples are used as training set and 716 samples are used for testing. The software complexity measures Base Halstead measures, LOC measure, Cyclomatic complexity, and Derived Halstead measures are used to classify the software modules. The proposed MLPNN is made up of 20 input neurons and two hidden layers. The results obtained for classification accuracy, precision and recall are shown from Fig. 2-4.

Table 1 Classification Accuracy

Techniques	Classification Accuracy
MLP with back propagation training	0.925
MLP with GA training	0.9307
Decision stump	0.8357

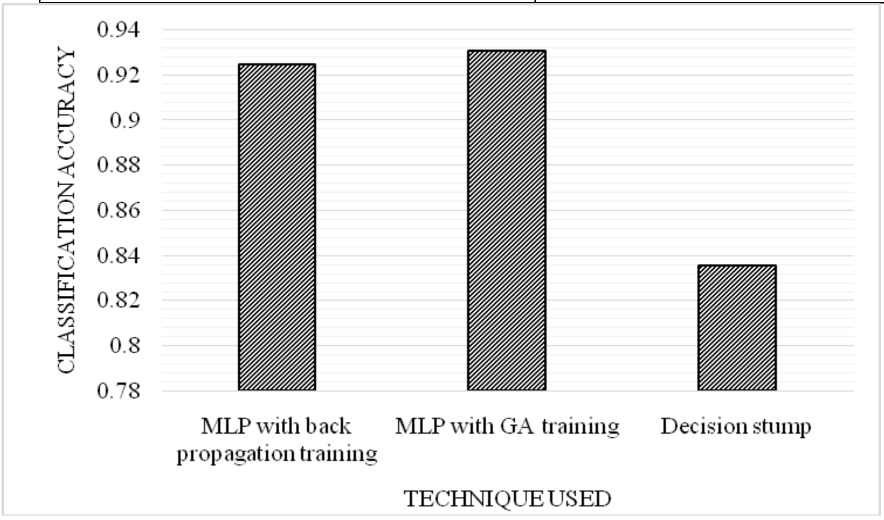

Figure 2 Classification Accuracy

From Fig. 2, it is observed that the proposed MLP with GA training increased classification accuracy by 0.61% when compared with MLP with back propagation training.

Table 2 Precision

Techniques	Precision for Defect	Precision for No Defect
MLP with back propagation training	0.835	0.9344
MLP with GA training	0.7355	0.9644
Decision stump	0.4471	0.9599

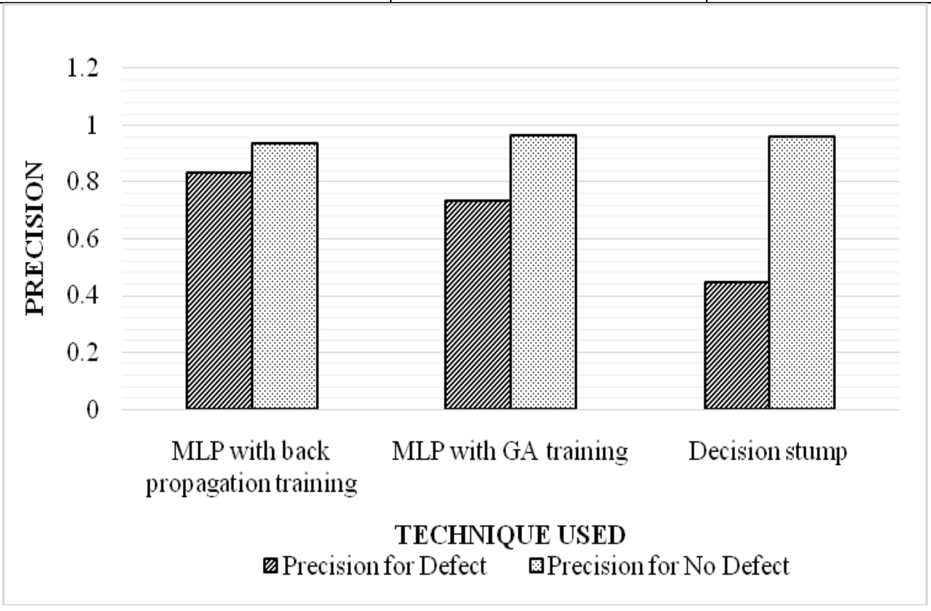

Figure 3 Precision

From Fig. 3, it is observed that the proposed MLP with GA training increased precision by 12.67% and 3.16% with defect and no defect respectively when compared with MLP with back propagation training.

Table 3 Recall

Techniques	Recall for Defect	Recall for No Defect
MLP with back propagation training	0.5719	0.9818
MLP with GA training	0.7808	0.9548
Decision stump	0.7808	0.8445

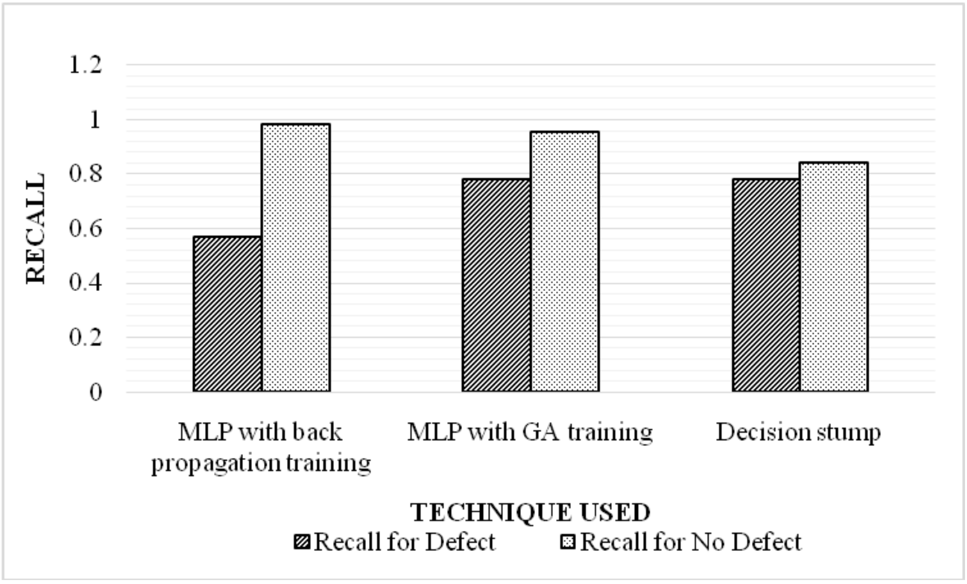

Figure 4 Recall

From Fig. 4, it is observed that the proposed MLP with GA training increased recall by 30.89% and 2.79% with defect and no defect respectively when compared with MLP with back propagation training.

Table 4 F Measure

Techniques	F Measure for Defect	F Measure for No Defect
MLP with back propagation training	0.9575	0.6788
MLP with GA training	0.9596	0.7575
Decision stump	0.8985	0.5686

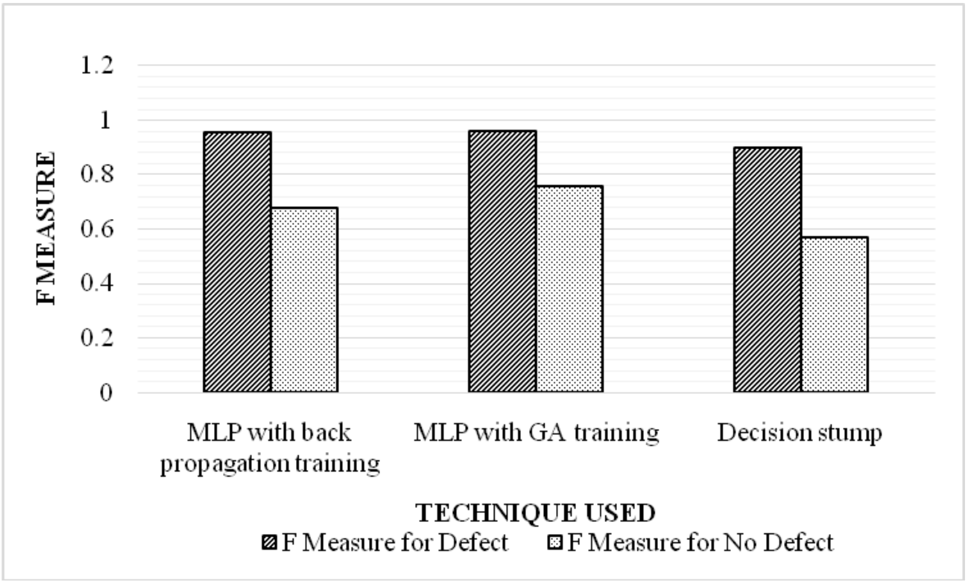

Figure 5 F Measure

From Fig. 5, it is observed that the proposed MLP with GA training increased f measure by 6.58% and 28.49% with defect and no defect respectively when compared with Decision stump.

V. CONCLUSION

SDP means to enhance software quality and testing efficiency by constructing predictive classification models from code attributes to ensure timely identification of fault-prone modules. In this system, another learning methodology was utilized as a part of MLP neural networks algorithm which improved network efficiency. Neural Network experiences the ill effects of parameter optimization issues which have not been addressed for the SDP problem. To beat the poor convergence of GA, an enhanced algorithm to train the Neural Network parameter was proposed. It is seen that the proposed MLP with GA training improved classification accuracy by 0.61% when compared with MLP with back propagation training.

REFERENCE

[1] Rawat, M. S., & Dubey, S. K. (2012). Software defect prediction models for quality improvement: a literature study. *International Journal of Computer Science*, *9*, 288-296.

[2] Joy, C. U. (2011). Comparing the Performance of Backpropagation Algorithm and Genetic Algorithms in Pattern Recognition Problems. *International Journal of Computer Information Systems*, *2*(5), 7-12.

[3] Correa, A., Gonzalez, A., & Ladino, C. (2011). Genetic Algorithm Optimization for Selecting the Best Architecture of a Multi-Layer Perceptron Neural Network: A Credit Scoring Case. In *SAS Global Forum 2011 Data Mining and Text Analytics*.

[4] Abdellah, M. B. (2013). Architecture Optimization Model For The Multilayer Perceptron And Clustering. *Journal of Theoretical and Applied Information Technology*, *47*(1).

[5] Sarangi, P. P., Mishra, B. S. P., Majhi, B., Dehuri, S., & Mohan, F. (2013). Hybrid Supervised Learning in MLP using Real-coded GA and Back-propagation. *International Journal of Computer Applications*, *62*(21).

[6] Andersen, H. C., & Tsoi, A. C. (1993). A constructive algorithm for the training of a multilayer perceptron based on the genetic algorithm. *Complex Systems*, *7*(4), 249-268.

[7] Syed Mustafa, A., &Swamy, K. (2015, June). Web Service classification using Multi-Layer Perceptron optimized with Tabu search. In *Advance Computing Conference (IACC), 2015 IEEE International* (pp. 290-294). IEEE.

[8] Mirjalili, S., & Sadiq, A. S. (2011, May). Magnetic optimization algorithm for training multi-layer perceptron. In *Communication software and networks (ICCSN), 2011 IEEE 3rd international conference on* (pp. 42-46). IEEE.

[9] Ikuta, C., Uwate, Y., Nishio, Y., & Yang, G. (2012, December). Improvement of learning performance of multi-layer perceptron by two different pulse glial networks. In *Circuits and Systems (APCCAS), 2012 IEEE Asia Pacific Conference on* (pp. 356-359). IEEE.

[10] Yang, X., Tang, K., & Yao, X. (2015). A Learning-to-Rank Approach to Software Defect Prediction. *Reliability, IEEE Transactions on*, *64*(1), 234-246.

[11] Okumoto, K. (2011, June). Software defect prediction based on stability test data. In *Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011 International Conference on* (pp. 385-387). IEEE.

[12] Can, H., Jianchun, X., Ruide, Z., Juelong, L., Qiliang, Y., &Liqiang, X. (2013, May). A new model for software defect prediction using particle swarm optimization and support vector machine. In *Control and Decision Conference (CCDC), 2013 25th Chinese* (pp. 4106-4110). IEEE.

[13] Pelayo, L., & Dick, S. (2012). Evaluating stratification alternatives to improve software defect prediction. *Reliability, IEEE Transactions on*, *61*(2), 516-525.

[14] Gayathri, M., &Sudha, A. (2014). Software Defect Prediction System using Multilayer Perceptron Neural Network with Data Mining. *International Journal of Recent Technology and Engineering (IJRTE) ISSN*, 2277-3878.

[15] Najadat, H., &Alsmadi, I. (2012). Enhance Rule Based Detection for Software Fault Prone Modules. *International Journal of Software Engineering and Its Applications*, *6*(1), 75-86.

[16] Askari, M. M., &Bardsiri, V. K. (2014). Software Defect Prediction using a High Performance Neural Network. *International Journal of Software Engineering and Its Applications*, *8*(12), 177-188.

[17] Castillo, P. A., Merelo, J. J., Prieto, A., Rivas, V., & Romero, G. (2000). G-Prop: Global optimization of multilayer perceptrons using GAs.*Neurocomputing*, *35*(1), 149-163.

[18] Balochian, S., Seidabad, E. A., & Rad, S. Z. (2013). Neural Network Optimization by Genetic Algorithms for the Audio Classification to Speech and Music. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, *6*(3), 47-54.

[19] Kumari,V. S. R. and Rajesh Kumar, P. (2015). Optimization of Multi-Layer Perceptron Neural Network Using Genetic Algorithm for Arrhythmia Classification. Communications. Vol. 3, No. 5, pp. 150-157.