International Journal of Engineering & Technology, 7 (2.6) (2018) 259-268

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

Empirical analysis of software quality prediction using a TRAINBFG algorithm

Saumendra Pattnaik[1]*, Binod Kumar Pattanayak[2]

[1] Saumendra Pattnaik, Department of Computer Science and Engineering, Institute of Technical Education and Research, Siksha 'O' Anusandhan Deemed to be University, Khandagiri, Odisha, India
[2] Prof.(Dr.) Binod Kumar Pattanayak, Department of Computer Science and Engineering, Institute of Technical Education and Research,Siksha 'O' Anusandhan Deemed to be University, Khandagiri, Odisha, India
**Email: saumendrapattnaik@gmail.com*

Abstract

Software quality plays a major role in software fault proneness. That's why prediction of software quality is essential for measuring the anticipated faults present in the software. In this paper we have proposed a Neuro-Fuzzy model for prediction of probable values for a predefined set of software characteristics by virtue of using a rule base. In course of it, we have used several training algorithms among which TRAINBFG algorithm is observed to be the best one for the purpose. There are various training algorithm available in MATLAB for training the neural network input data set. The prediction using fuzzy logic and neural network provides better result in comparison with only neural network. We find out from our implementation that TRAINBFG algorithm can provide better predicted value as compared to other algorithm in MATLAB. We have validated this result using the tools like SPSS and MATLAB.

Keywords: *MATLAB; Software quality metrics; Fuzzy Logic; SPSS; Neural Network.*

1. Introduction

A prediction is basically a forecasting of various things based on some existing facts and data. Software metric prediction [1] is the biggest concern for any software firm because through this we can analyze or estimates the failure rate or chances of getting failure. Metrics are used for prediction of various measures like failure rate, fault density of executable line of codes, final failure rate etc. the prediction outcome includes:

i) Number of faults in a fixed phase of life cycle.
ii) The failure rate of prediction when realizing the system.
iii) The different measures for software management.

The software metrics are basically collected at different phases of development cycle. These metrics contains the information about the software and used for prediction of software quality at the early stages of development cycle. The prediction of software quality generally dependent on the practical data and the techniques having the advantage of flexibility of controlling uncertain data. So generally people prefer machine learning techniques like fuzzy logic [2], NN network or genetic algorithm for prediction of software quality. As the machine learning techniques are used for handling uncertain data set, so the prediction performance can be better assured using these techniques.

The different sections of our paper are described as follows. Section 2 contains the background knowledge of various machine learning techniques and different software metrics used for software quality prediction. Section 3 gives the literature survey of previous work. Section 4 gives the idea on the various machines learning technique used for software quality prediction. Section 5 highlights the implementation using SPSS and MATLAB tools. Section 6 describes a brief discussion about the implementation of results. Section 7 concludes the paper along with future work.

2. Background study

2.1. Machine learning techniques

2.1.1. Fuzzy logic

Reasoning in Fuzzy logic replicates human reasoning. Decision making approach in Fuzzy Logic resembles to that in a human which subsequently involves all possible values in the range [0, 1] that refers to decision parameters YES or No.

The digital logic implemented in a computer can accept digital values 0 and 1 as input and provides TRUE or FALSE as output that refers to YES or NO as perceived by humans.

The inventor of fuzzy logic, as observed by Lotfi Zadeh, the inventor of Fuzzy Logic, human decision can take possible assumptions between YES and NO in contrast with computer decision making that can provide either TRUE or FALSE. Different assumed possibilities between YES and NO for a human can be given as:

- CERTAINLY YES
- POSSIBLY YES
- CANNON SAY
- POSSIBLY NO
- CERTAINLY NO

The fuzzy logic works on the levels of possibilities of input to achieve the definite output.

Utilization of fuzzy logic:

Fuzzy logic can be utilized in various sectors with multiple usages in small micro controller to large work station. It can be utilize in the field of hardware as well as software.

Utility of fuzzy Logic?

It is utilized in commercial as well as practical purpose. This is helpful in controlling machine and various consumer goods. It

provides approximate values but in the acceptable range. It is helpful in dealing with uncertainty factors.

Fuzzy logic systems architecture:

It has four main parts as shown below.

Fuzzification module- It transforms the system inputs that are crisp numbers into fuzzy sets. It splits the input signal into five steps shown in Table 1.

Table 1: Fuzzy Input Signals

LP	x is Large Positive
MP	x is Medium Positive
S	x is Small
MN	x is Medium Negative
LN	x is Large Negative

Knowledge base − It stores all the IF-THEN as recommended by experts.

Inference engine − It simulates the human reasoning behavior demonstrated by humans by virtue of application of Fuzzy inference procedure to the inputs used in the IF-THEN rules.

Defuzzification module − It is responsible for transforming the Fuzzy sets obtained by the inference engine into respective crisp values.

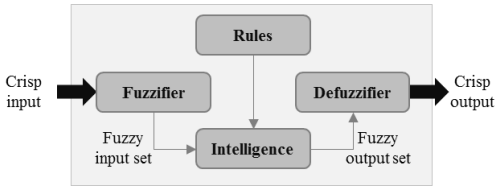

Fig. 1: The membership functions work on fuzzy set of variables

Membership function

Membership functions shown in Figure 1 can be used for quantification of a linguistic term and its representation in the form of a fuzzy set graphically. A membership function for a given fuzzy set, A defined over X, called as the universe of discourse, can be represented as $\mu A: X \rightarrow [0, 1]$.

Here, every element belonging to X is assigned with a value in the interval [0, 1]. It is called membership value or degree of membership. It quantifies the degree of membership of the element in X to the fuzzy set A.

x axis represents the universe of discourse.

y axis represents the degrees of membership in the interval [0, 1].

In order for fuzzification of a given numeric value, multiple membership functions can be used too. Simple membership functions need to be used for the reason that a complex function is incapable of adding more precision to the output.

2.1.2. Neural networks

Artificial neural network (ANN) [3] is a form of electronic network that comprises of nodes called as "neurons" (Fig. 2). It has been devised keeping in view the structure of a human brain. Such a network is capable of processing one record at a time and thereby tend to "learn" by virtue of comparing their classification of the record (which at the outset is largely arbitrary) with a known classification of the record. Here, the errors that are obtained from the initial classification of the first record are further fed back into the network, subsequently used for modification of the network algorithm in the following round, and thus, several iterations are carried out.

Precisely, a neuron in an ANN comprises of:

1. A predefined set of input values (x_i) along with a set of associated weights (w_i)
2. A function (g) that is used to sum up the weights and map the results into an output (y).

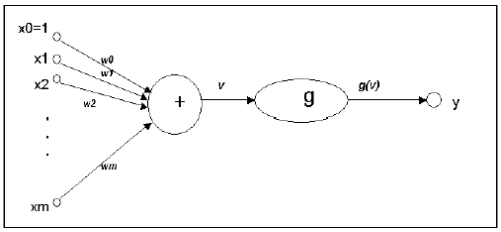

Fig. 2: Neurons are organized into layers

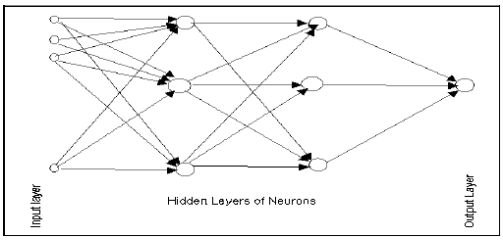

Fig. 3: Hidden Layers of Neurons

The input layer does not comprise of full neurons here, rather consists of the values incorporated in a data record that represents the inputs to the next layer of neurons. The next layer here is considered as a hidden layer (Fig. 3). It must be noted that there may be several hidden layers in an ANN. The last layer in the network represents the output layer, where there is one node representing each of the classes. A single sweep forward through the network results in assigning a specific value to each of the output nodes and the record thereby is assigned to a class node that possesses the highest value.

Training of an ANN model:

While training the ANN, the correct class for each of the records in already known and output nodes is assigned 1 as correct value at the node which corresponds the right class and 0 for rest (practically 0.9 and 0.1 are usually considered as better values). Comparing the calculated values of the output nodes of the network to the correct values are the error terms of each nodes (the "Delta" rule). The use of the error terms to further adjust the hidden layers weights [4], so that the output values in the next round will be closer to the correct values.

Structuring the network:

The number of layers and the processing elements per layer in the network need to be correctly chosen. The parameters of feed forward network with the back propagation topology are significantly important which depends on the design of the network. For any given application there is no quantifiably appropriate answer to the layer out of the network where most of the researchers and engineers follow the general rules chosen from time to time for applying this particular architecture to their desired application.

Rule 1: The number of the processing elements for the hidden layer to be increased with respect to the complexity of the relationship exists between the input and the required output.

Rule 2: A need of extra hidden layer maybe required if and only if the process that requires to be modeled separately into multiple stages. If it fails the additional layers may facilitate the memorization of the training set and not as true general solution effective with other data.

Rule 3: An upper bound to the number processing units is determined by the available training data and to be included in the hidden layers. For the calculation of the upper bound the training data set available in the number of cases needs to be divided by the sum of input nodes as well as output layers in the network. Again the result is divided by a scaling factor chosen from 5 to 10. Higher scaling factors need to require comparatively less noisy data. The training data set must be remembered where too many artificial neurons are used such that data generalization would not be viable for making the network of no use for the new data sets.

2.2. Software metrics

The software metrics follow different models but ISO 9126 model which holds six software quality metrics as functionality, reliability, usability, efficiency, maintainability [5] and portability holds good. For predicting the software quality using the metrics provides better prediction output in comparison with other models.
The software quality prediction [6] model takes the above stated six attributes as its input. According to the rule base the prediction [7] model works and provides the correct output. Here we have utilized some input predictors like Depth of inheritance tree (DIT), Number of children (NOC), Coupling between Objects (CBO), Response for a class (RFC), weighted methods per class (WMC) and Inherited Coupling (IC). Through this input predictors we try to predict the software quality of a given set of software quality parameters.

2.2.1. Functionality

It is defined with respect to the attributes those significantly determine the set of functions achieved along with their respective properties.
The different sub characteristics [8] of the metric functionality are as follows:
Suitability: It determines the conformation to the purposes.
Accuracy: It determines the level of conformance to the obtained results and at the same time it specifies how authentic are the attributes of the software in order for interacting with specific systems.
Interoperability: It identifies the attributes those necessarily define the capabilities of the software in order for interacting with specific systems.
Functionality compliance: It refers to the conformance of the software with the standards as well as the convention pertaining to the applications and the regulation as well.

2.2.2. Maintainability

It refers to incorporation of necessary modification in a software component when there need be.
The different sub characteristics of the metric maintainability [9] are as follows:
Analyzability refers to comfort ability of diagnosis of the errors, the reasons of failure and identification of the parts those need to be modified.
Changeability defines the convenience of modification of the software in accordance with the changes in the environment.
Testability defines the ease in testing of the software with an intention to identify the errors.
Stability defines the attributes of the software those determine the level of risk caused from unexpected changes.
Maintainability compliance [10] refers to the conformance of the software to specified standards meant for maintainability compliance.

2.2.3. Efficiency

It is defined with respect to the attributes that necessarily determine the level of performance of the software in the context of utilization of time and resources.
The different sub characteristics of the metric Efficiency are as follows:
Time behavior of a software deals with the attributes those provide a quantitative measure of characteristics like processing time, response time as well as throughput rates.
Resource behavior provides a measure of resources utilized along with the respective duration of their usage.
Efficiency compliance determines if the software conforms to the respective standards of efficiency.

2.2.4. Usability

Usability of software refers to the capabilities from a user's point of view in terms of learning, understanding, configuring, executing and using it as per requirement.
The different sub characteristics of the metric Usability are as follows:
Understandability incorporates the attributes those explicitly disclose the underlying logic and its applicability.
Learnability refers to the attributes those describe the convenience of a user for learning the applications.
Operability refers to the attributes of the software those make it easy to understand the operations incorporated in the software.
Attractiveness describes the level of attractiveness of the software.
Usability compliance defines if the software conforms to the specified standards of usability compliance.

2.2.5. Portability

Portability of software is regarded as the flexibility of functionality of it from one environment to other.
The different sub characteristics of this are as follows:
Replace-ability defines the attributes of a software those describes the spectrum of adaptations of the software.
Adopt-ability defines the ability of the software in order for adoptability of it to several other environments without bringing any modification except for the specified purpose.
Install-ability determines the convenience of installation of software in a specified environment.
Co-existence defines the property of the software that makes it exist in the system without affecting the other processes.
Portability compliance refers to the attributes those enable the software to conform to the specified the standards & conventions.

2.2.6. Reliability

Reliability of software is the feature that can predetermine the probable failures of software or any component of it.
The different sub characteristics of this are as follows:
Maturity: It determines how frequently failures occur in the software resulting from errors.
Fault-tolerance: It refers to how robust relates to those attributes of the software. It describes the capability of the software to maintain a certain level of performance even if faults do occur.
Recoverability: It refers to the ability of the software to regain its desired level of performance in case of faults & to recover the affected data.
Reliability compliance determines if the software complies with the specified standards of reliability compliance.

3. Related work

According to Malhotra *et.al.,* [11] a suitable model using machine learning algorithm for prediction of software maintainability is used for reducing the prediction errors. The prediction performance are assessed and compared with the existing model. The result shows the proposed model is better than the existing model according to the error percentage. So this model is a useful modeling technique and can be used for the prediction of software maintainability.
According to Bhutani *et.al.,* [12] general neural network (GRNN) gives best result in comparison with probabilistic neural network and group method of data handling. So general neural network (GRNN) is the best prediction model for prediction software maintainability. This model can be added with other data mining model for more accurately predicting software maintainability.
According to Agarwal *et. al.,* [13] it examines the application of the neural network for predicting the software maintainability using object oriented metrics. Here the dependent variable is maintenance effort. The quality estimation includes estimating

maintenance effort. The independent variables are the principal component of eight object oriented metrics. Here the result shows that the mean absolute relative error is very less in compared with the other model i.e. 0.265.

According to Osama *et.al.,* [14] in the year 2016 the software quality prediction system is comprises of two parts. One is classification algorithm and other one is software quality measurement. The classification algorithm can be using the tools like WEKA and SPSS. The software quality measurement model depends on machine learning algorithm like fuzzy logic, neural network or genetic algorithm. The result from the experiment shows that multilayer perception network is having more balance prediction in comparison with other machine learning technique.

Authors Jain *et.al.,* [15] claim that software maintenance starts from delivering to the customer till the expiry of the software. If we can predict the maintainability accurately, the maintenance activity can be highly reduced. In this paper GA is used for prediction of maintainability and a comparison with different other machine learning techniques has been carried out. Based on the comparison authors conclude that the prediction using GA appears to be better than other machine learning algorithms.

Authors Zubair A. Baig *et.al.,* [16] address the AI techniques those can be used for estimation of the maintenance cost of software. They have used neural network for predicting the software maintainability. Their approach is a hybrid one where neural network is trained for prediction and Genetic Algorithm (GA) is used for implementation. They obtain an accuracy of prediction that equals to 91%.

4. Technique used

4.1. Fuzzy logic

Fuzzy Logic (FL) [17] is a concept used for handling uncertain less. This helps in decision making using different reasoning concept. The reasoning is based on a specific range with scalar inputs. The approach of FL imitates the way of decision making in humans that involves all intermediate possibilities between digital values YES and NO.

Example 1: let us consider a simple blood pressure (BP) machine whose values and range uses the following reasoning. We have a scalar input, as blood pressure reading scalar and output, measurement (y).

Example:

R1: If BP is greater than 0 and less than 60, then BP is low.

R2: if BP is greater than 60 and less than 120, than BP is Normal.

R3: If BP is greater than 120 and less than 180, than BP is High.

The set of antecedent linguistic terms:

A= {(r1, r2, r3) s.t. $0 < r1 < 60$

$60 < r2 < 120$

$120 < r3 < 180$}

B = {Low, Normal, High}

The qualitative relationship between the model input and output can be expressed by the following rules:

R1: If BP is greater than 0 and less than 80, than BP is Low.

R2: If BP is greater than 60 and less than 120, than BP is Normal.

R3: If BP is greater than 110 and less than 180, than BP is High.

The meaning of the linguistic terms is defined by their membership functions i.e. shown below in Figure 4.

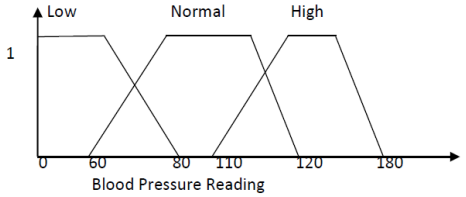

Fig. 4: Membership functions

4.2. Neural model

Our proposed model Fuzzy logic Neural Network (FLNN) is a combination of fuzzy logic concept along with neural network architecture. In this model we have applied the rule base system of fuzzy logic and three tier architecture of neural network.

The rule base system in fuzzy logic holds IF-THEN rules along with the outcomes of it. In this application, we have considered the rule base system as mentioned in Fig. 5. We have considered five input systems and 4 output nodes to predict the software quality [18] of the given software.

Earlier different authors have not checked the efficiency level of the prediction model. The fuzzy rule base system is basically used to handle the system input and output of the FLNN (Fuzzy Logic Neural Network) model in the given application as mentioned in Fig. 5. We have considered certain rules which may be more effective in comparison with the ones conducted by different authors earlier.

The Rule based system is depicted below (Neural network),

RULE 1 (R1) : If IC is High, RFC is Many, NOC is Few, DIT is Shallow and CBO is High, then Functionality is High, Maintainability is Low, Efficiency is High and Usability is High.

RULE 2 (R2) : If IC is Low, RFC is Many, NOC is High, DIT is Deep and CBO is High, then Functionality is Low, Maintainability is High, Efficiency is Low and Usability is Low.

RULE 3 (R3) : If IC is Low, RFC is Many, NOC is Many, DIT is Shallow and CBO is Low, then Functionality is High, Maintainability is Low, Efficiency is High and Usability is High.

RULE 4 (R4) : If IC is High, RFC is Few, NOC is Few, DIT is Shallow and CBO is High, then Functionality is High, Maintainability is Low, Efficiency is High and Usability is Low.

5. Implementation

5.1. Using SPSS

SPSS is a statistical tool which is used for calculating various types of analytical concepts. Through this tool one can analyze the data source. This statistical analysis includes correlations between variables, regression line, neural network model generation and other analyses. SPSS is used for getting a report in the output window which provides various analyses in an organized way.

In SPSS the hidden layers and input layers bias weights along with input weights of different layer are evaluated through SPSS multilayer perceptron. The synaptic weights are shown in Fig. 6. We find out the neural network of the variables present in the input layer as shown in Fig. 7. We also evaluated the 3 different dependent variables DIT, NOC, MPC along with its sensitivity shown in Fig. 8, Fig. 9 and Fig. 10. The area under the curve of the 3 dependent variables is provided in Fig. 11. In Table 2 the training data, sample data and the holdout data are given in order to check the validity percentage.

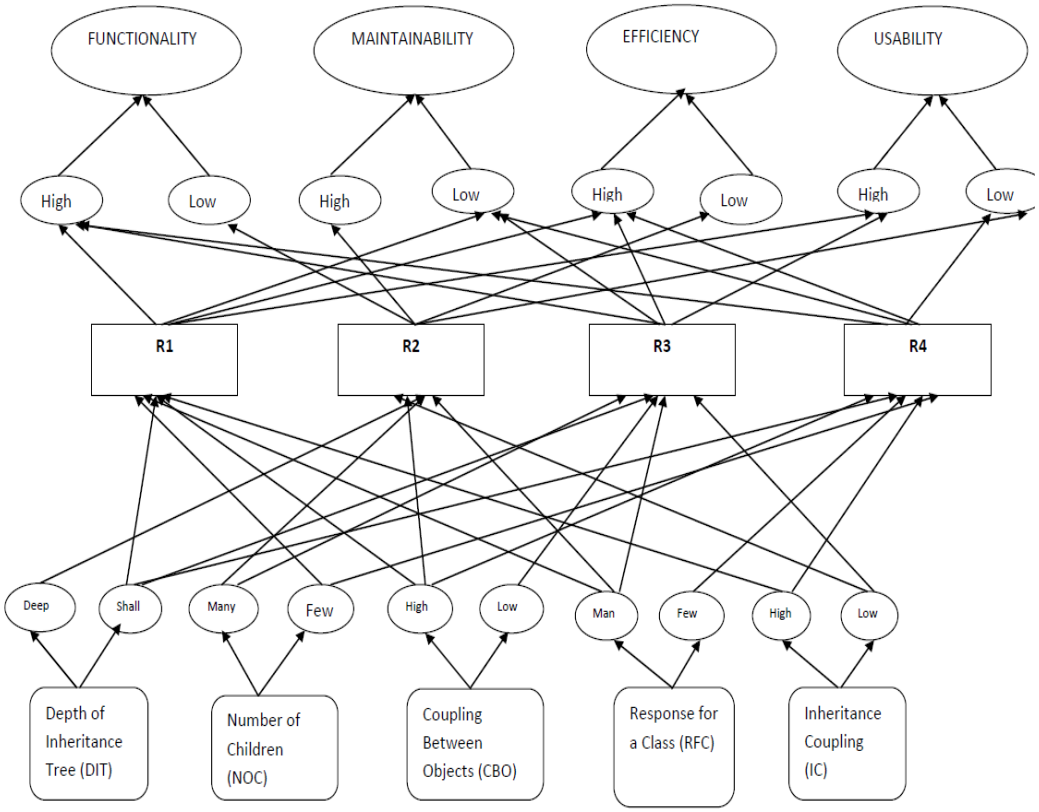

Fig. 5: FLNN Software Quality Prediction Model

		Predicted																
		Hidden Layer 1			Hidden Layer 2			Output Layer										
Predictor		H(1:1)	H(1:2)	H(1:3)	H(2:1)	H(2:2)	H(2:3)	[DIT=0]	[DIT=1]	[DIT=2]	[DIT=3]	[DIT=4]	[NOC=0]	[NOC=1]	[NOC=3]	[NOC=4]	[NOC=6]	[NOC=8]
Input Layer	(Bias)	.514	.412	-.350														
	DAC	.141	-.395	.575														
	WMC	-.277	.440	.498														
	NOM	.118	.233	-.396														
	ICOM	-.011	-.150	-.144														
	RFC	.383	-.140	.281														
	MPC	.347	-.222	-.465														
Hidden Layer 1	(Bias)				.430	.238	.133											
	H(1:1)				.671	-.257	.086											
	H(1:2)				-.882	.306	.446											
	H(1:3)				-.890	-.645	.415											
Hidden Layer 2	(Bias)							.111	.176	.398	.038	-.159	.203	.081	.154	.163	.160	.037
	H(2:1)							-.108	.298	-.699	.535	.192	.430	-.116	-.065	.277	-.171	-.093
	H(2:2)							.057	-.464	.343	.267	.162	.607	.091	-.250	-.458	-.011	.084
	H(2:3)							.072	.270	.496	-.348	.284	.305	.020	.342	-.011	-.039	-.034

Fig. 6: Parameter estimates

.

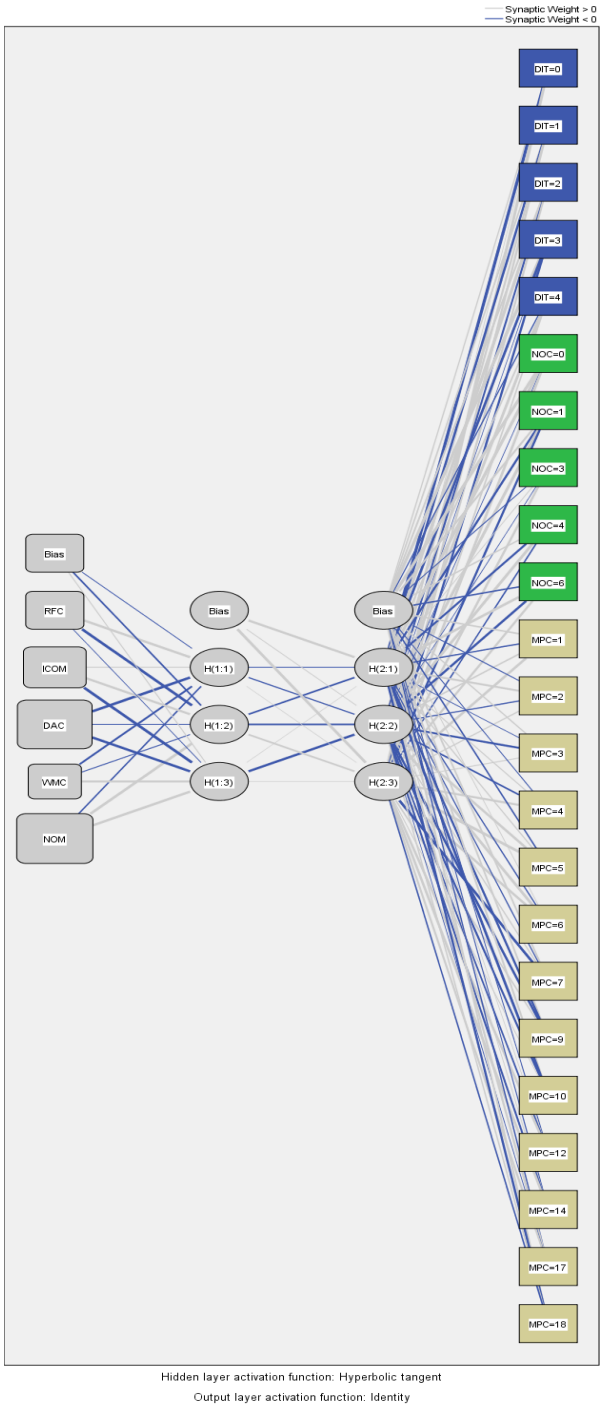

Fig. 7: Hidden layer Activation functions

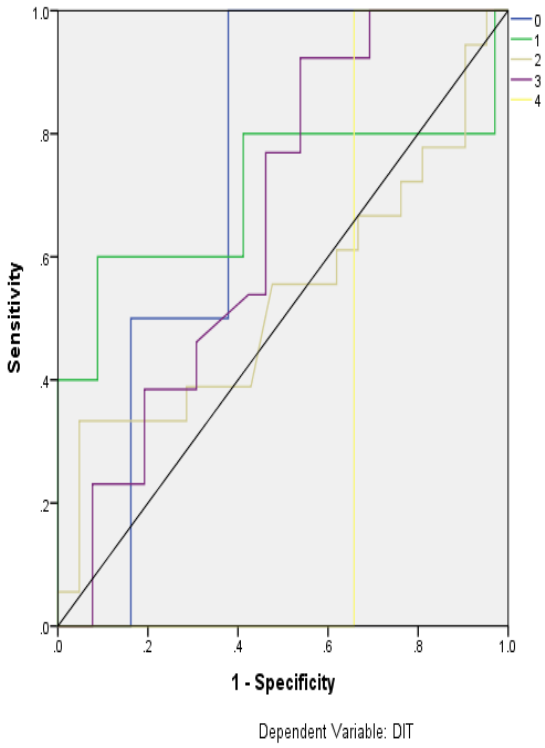

Fig. 8: Dependent variable DIT

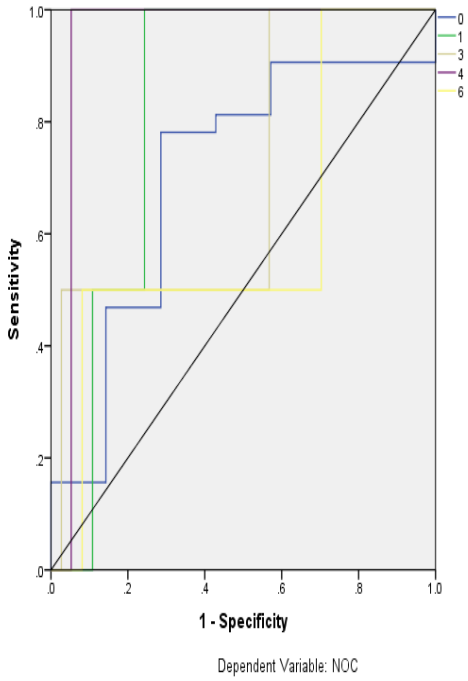

Fig. 9: Dependent Variable NOC

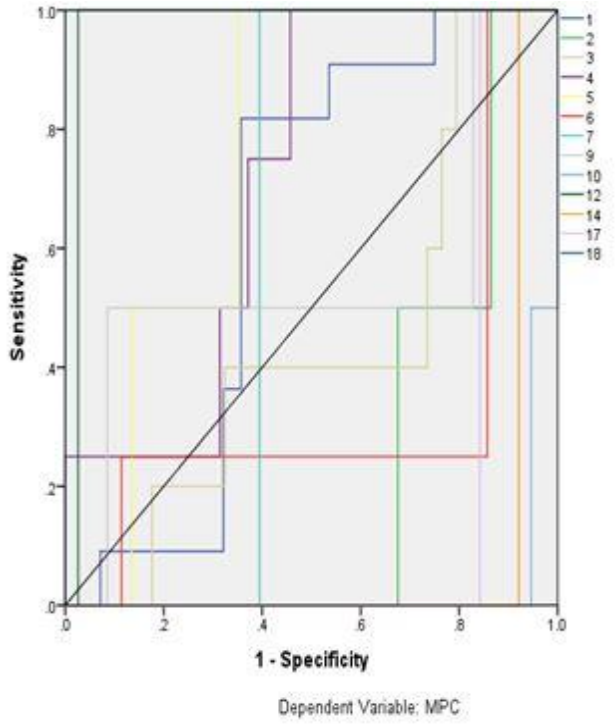

Fig. 10: Dependent value MPC

		Area
DIT	0	.730
	1	.706
	2	.533
	3	.658
	4	.342
NOC	0	.705
	1	.824
	3	.703
	4	.947
	6	.608
MPC	1	.627
	2	.230
	3	.441
	4	.714
	5	.757
	6	.329
	7	.605
	9	.543
	10	.027
	12	.974
	14	.079
	17	.158
	18	1.000

Fig. 11: Area under the Curve Dependent Variable

Table 2: Case Processing Summary

		N	Percent
Sample	Training	32	74.4%
	Testing	7	16.3%
	Holdout	4	9.3%
Valid		43	100.0%
Excluded		10	
Total		53	

5.2. Using NNTOOL in MATLAB

Nntool in MATLAB provides various advantages such as predefined algorithm, pre-trained models and application for creating, training or simulating the different types of neural networks. In this tool it provides the facility of classification, regression, factor analysis, clustering and dynamic system modeling etc. when the network is created in MATLAB, by default it generates the code for performing the task given to it. The various algorithm supported by the nntool are like TRAINLM, TRAINBFG, TRAINGDA etc. Nntool helps in modeling the algorithms that is integrated within the software. At the time of training we can provide errors in order to check the importance of the output. We can access the training algorithm according to our requirement and can visualize the training process i.e. Fig. 12, training state i.e. Fig. 13, performance plot i.e. Fig. 14 and regression i.e. Fig. 15. The work flow of a neural network can be described as follows.

Step 1: Collect the data
Step 2: Create the network
Step 3: Configure the network with input and outputs.
Step 4: Provide the values of synaptic weights and biases
Step 5: The neural network training process will continue.
Step 6: Validate of the neural network is done.
Step 7: Utilize the neural network.

In MATLAB we are training the data till the minimized gradient is reach as shown in Fig. 12. The performance training state and regression are evaluated and shown in Fig. 14 and Fig. 15. In the training process for checking the validation we have got epoch graph shown in Fig. 13. After training we have got the predicted synaptic weights shown in Fig. 16.

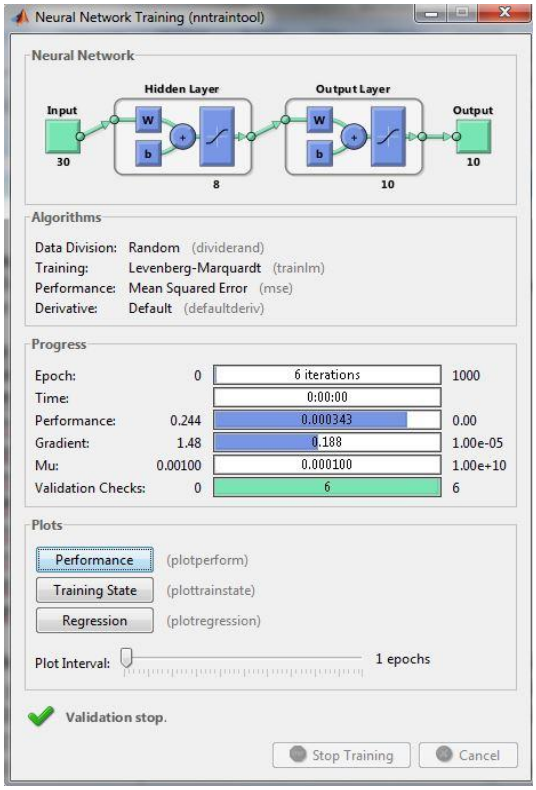

Fig. 12: Snapshot generated during the Training of Neural Network

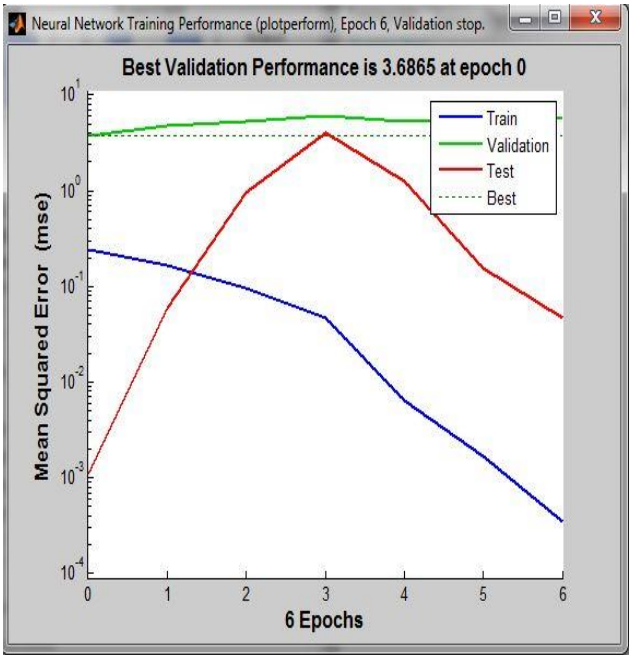

Fig. 14: Snapshot of Neural network Training Performance

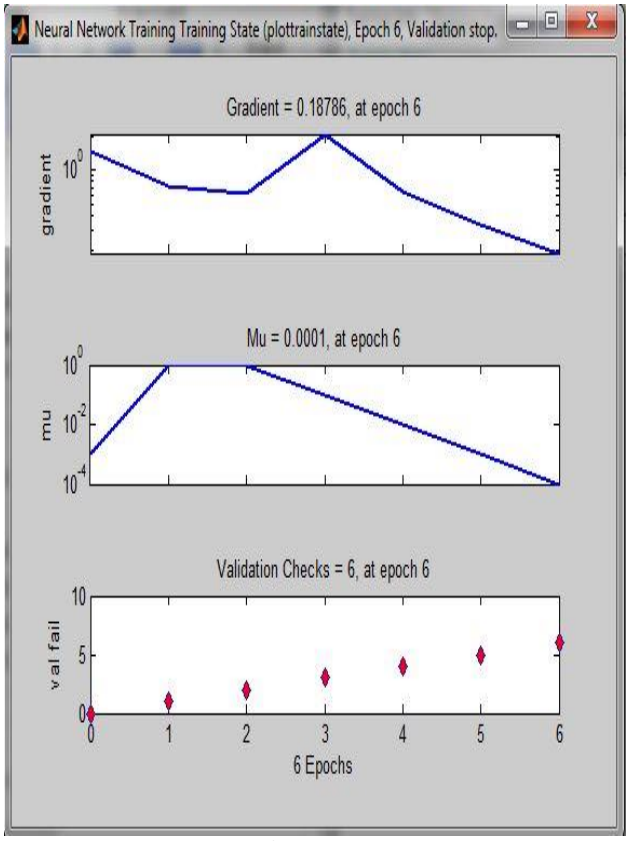

Fig. 13: Snapshot of Neural Network Training State

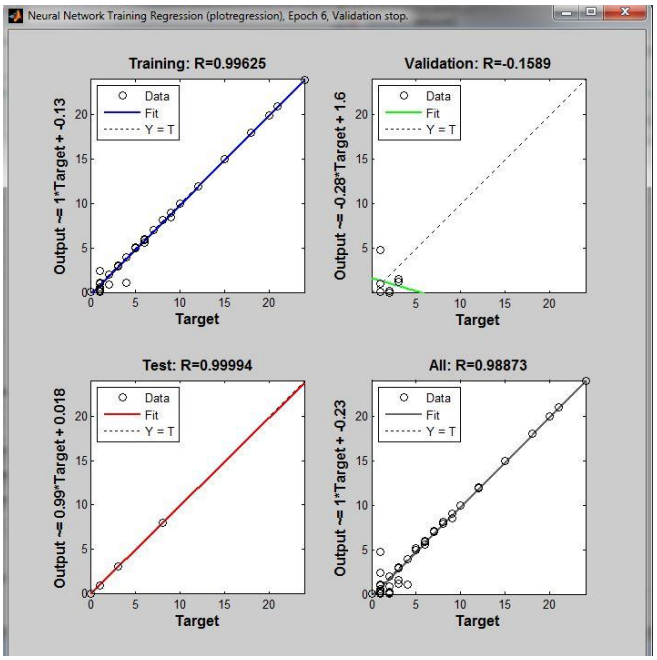

Fig. 15: Snapshot of Neural network Training Regression

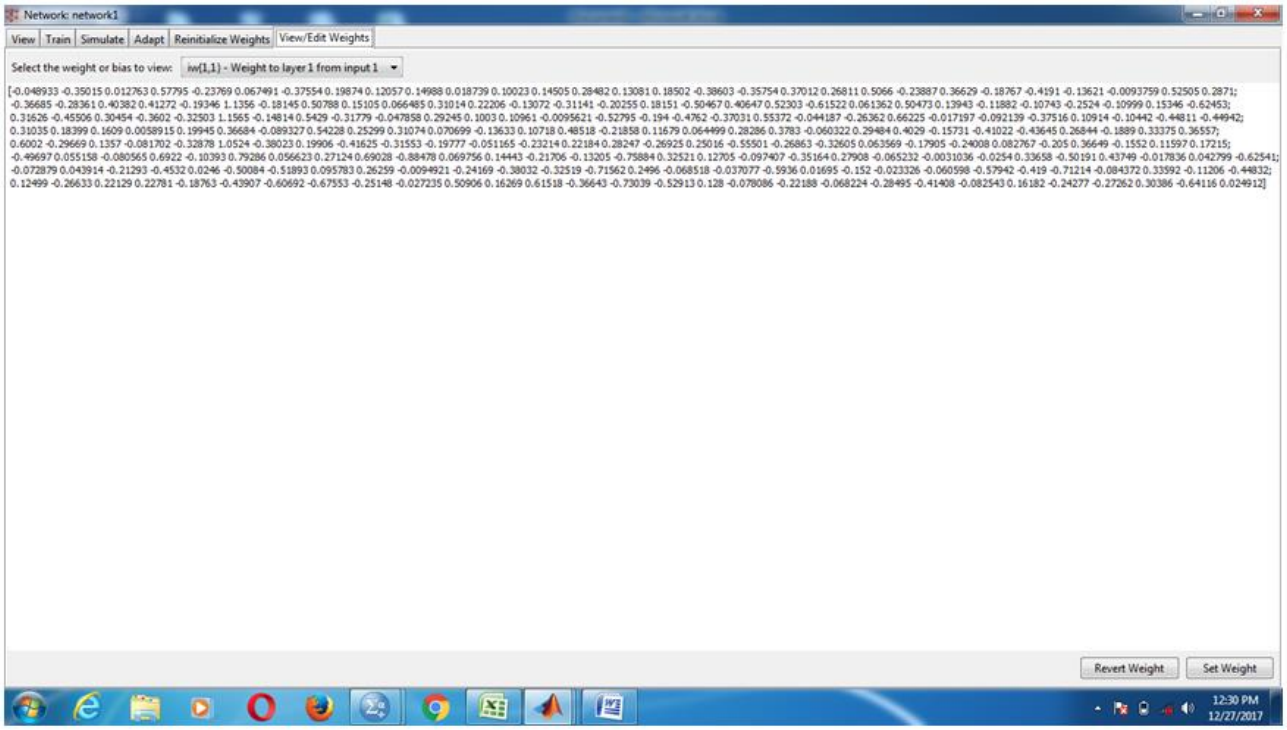

Fig. 16: Snapshot of Synaptic Weight Prediction

Training algorithm results:

Using the nntool in MATLAB we have applied eight different training algorithms with input, target and four different samples from the data set. The best results from the training process are shown in Table 3.

Table 3 Results of Training Process

TRAINING ALGORITHMS	TEST 1	TEST 2	TEST 3	TEST 4
TRAINLM	0.244	0.101	0.298	0.398
TRAINGDM	2.81	3.41	3.95	2.32
TRAINSCG	0.364	3.39	0.879	0.677
TRAINBFG	6.15	4.27	4.80	7.98
TRAINRP	0.437	1.22	0.552	0.736
TRAINOSS	5.17	6.28	5.07	3.60
TRAINGDX	0.327	0.658	0.981	0.656
TRAINGDA	2.68	2.98	1.57	1.21

From the comparison of different training algorithm of nntool we find out that TRAINBFG algorithm is much suited algorithm for software metric prediction [19]. We have analyzed from the synaptic weights obtained from SPSS tool and nntool of MATLAB. We find out that TRAINGDA algorithm are more fitting for prediction but TRAINBFG is suited most for the prediction of software quality metrics.

6. Discussion

Out of these tests using MATLAB we have utilized 8 different algorithms. These algorithms holds good for various samples. But TRAINBFG algorithm is the most appreciated algorithm with respect to the comparison with the synaptic weight factors. So TRAINBFG algorithm can provide better prediction with numerous samples in the software metric data set. Neural network technique is thus the better prediction technique in compression with Fuzzy logic and Genetic algorithm. But the combination of both Fuzzy logic and neural network [20] holds even well in compression with only neural network. We have checked these using two different tools SPSS and MATLAB.

7. Conclusion and future work

Prediction of software quality is a tedious job as different metrics can be influenced by different factors. Prediction using machine learning techniques can provide us much better results as it handles the uncertainties. Neural network and fuzzy logic are the two techniques which are utilized in our research work for providing better accuracy in prediction of software quality. Neural network uses various training algorithms for prediction, but TRAINBFG is observed to be the most accurate training algorithm for our data set. As genetic algorithm provides fitness function, so we can check the threshold value of algorithms. Also, in order to reduce the time complexity as well as the overall cost of the software system, genetic algorithm along with NN (Neural Network) could be more effective which can be a future extension to our investigation.

References

[1] Pattnaik, Saumendra, and Binod Kumar Pattanayak, "A survey on machine learning techniques used for software quality prediction", *International Journal of Reasoning-based Intelligent Systems*, Vol.8, No.1-2, (2016), pp.3-14.

[2] Hema, K., and S. Muruganandam, "A New Object Oriented Software Development Genetic Algorithm to Fuzzy Assignment Problem", 2016, *Transylvanian Review 1*.

[3] Attarzadeh, I., & Ow, S. H., "Proposing a new software cost estimation model based on artificial neural networks", *In Computer Engineering and Technology (ICCET), 2010 2nd International Conference IEEE*, Vol. 3, (2010, April), pp. V3-487.

[4] Han, Song, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network", *In Advances in neural information processing systems*, 2015, pp. 1135-1143.

[5] Dubey, S. K., & Rana, A., "Assessment of maintainability metrics for object-oriented software system", *ACM SIGSOFT Software Engineering Notes*, Vol.36, No.5, 2011, pp. 1-7

[6] Khoshgoftaar, Taghi M., Kehan Gao, and Amri Napolitano, "An empirical study of feature ranking techniques for software quality prediction", *International Journal of Software Engineering and Knowledge Engineering*, Vol.22, No.2, 2012, pp. 161-183.

[7] Yadav, Dilip Kumar, S. K. Charurvedi, and R. B. Mishra, "Early software defects prediction using fuzzy logic", *International Journal of Performability Engineering*, Vol.8, No.4, 2012, pp. 399-408.

[8] Sarrab, M. and Rehman, O.M.H., "Empirical study of open source software selection for adoption, based on software quality characteristics", *Advances in engineering software*, Vol. 69, 2014, pp.1-11.

[9] Jeet, K., Dhir, R. and Verma, H., "A comparative study of Bayesian and fuzzy approach to assess and predict maintainability of the software using activity-based quality model", *ACM SIGSOFT Software Engineering Notes*, Vol.37, No.3, 2012, pp.1-9.

[10] Jain, Ashu, Sandhya Tarwani, and Anuradha Chug., "An empirical investigation of evolutionary algorithm for software maintainability prediction", *In Electrical, Electronics and Computer Science (SCEECS), IEEE Students' Conference*, 2016, pp.1-6.

[11] Malhotra, Ruchika, and Anuradha Chug., "Software maintainability prediction using machine learning algorithms", *Software Engineering: An International Journal (SEIJ)*, Vol.2, No.2, 2012, pp.19-36.

[12] Bhutani, Samridhi, and Anuradha Chug., "Prediction of Software Maintainability using Neural Networks", *International Journal of Computer Science & Communication Networks*, Vol.5, No.2, 2011, pp.92-95.

[13] Aggarwal, K. K., Yogesh Singh, Arvinder Kaur, and Ruchika Malhotra., "Application of artificial neural network for predicting maintainability using object-oriented metrics", *Transactions on Engineering, Computing and Technology*, Vol.15, 2006, pp.285-289.

[14] Alshareet, Osama, Awni Itradat, Iyad Abu Doush, and Ahmad Quttoum., "A novel software quality prediction system based on incorporating ISO 9126 with machine learning", *Journal of Theoretical and Applied Information Technology*, Vol.94, No.2, 2016, pp.283-293.

[15] Jain, Ashu, Sandhya Tarwani, and Anuradha Chug., "An empirical investigation of evolutionary algorithm for software maintainability prediction", *In Electrical, Electronics and Computer Science (SCEECS), IEEE*, 2016, pp.1-6.

[16] Baqais, Abdulrahman Ahmed Bobakr, Mohammad Alshayeb, and Zubair A. Baig., "Hybrid intelligent model for software maintenance prediction", *Proceedings of World Congress on Engineering, London, U.K. Springer*, 2014, pp.358-362.

[17] Shunmuganathan, K. L., "An intelligent temporal adaptive genetic fuzzy classification algorithm for effective intrusion detection", 2016, *Transylvanian Review 3*.

[18] Zulzalil, Hazura, Abdul Azim Abdul Ghani, Mohd Hasan Selamat, and Ramlan Mahmod., "Using fuzzy integral to evaluate the web-based applications", *Malaysian Software Engineering Interest Group (MSEIG)*, 2011.

[19] Sharawat, Mr Sandeep.,"Software Maintainability Prediction Using Neural Networks", *Environment 3*, Vol.2, No.2, 2012, pp.750-755.

[20] Khosravi, Abbas, Saeid Nahavandi, Doug Creighton, and Amir F. Atiya., "Comprehensive review of neural network-based prediction intervals and new advances", *IEEE Transactions on neural networks*, Vol.22, No. 9, 2011, pp.1341-1356.