Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2016, 3(7): 78-84

Research Article ISSN: 2394 - 658X

Software Bug Prediction System Using Neural Network

Rakesh Kumar and DL Gupta

Department of Computer Science and Engineering, KNIT, UP, India
rkyknit@gmail.com

ABSTRACT

In modern scenario the IT industries are investing much capital on software quality testing. Software testing phase of software development life cycle provide the defects details to the software developer. The proposed model is capable to predict the defect before going to the software testing phase. It also minimize the time and cost of the project. In this paper object oriented defected dataset ANT 1.7 has been used which is provided by the PROMISE (Predict or Models in Software Engineering) repository of empirical software engineering data. In these data sets the metrics and bug are freely available to the researchers publically. The main objective of this work is to calculate the accuracy and performance of the proposed software bug prediction model based on different techniques.

The proposed model is based on GDA (Gradient descent with adaptive learning back propagation) techniques. The proposed model is providing 98.97 % accuracy which is better as compare to the other existing models for Ant dataset.

Key words: Bug prediction, Dataset, Gradient descent with adaptive learning back propagation techniques, Metrics

---

INTRODUCTION

The different types of software metrics like class level, method level, file level, process level are used to find fault in the software before the testing the software at early stage of software developments life cycle. There are different methods which are used to find the software faults, the methods are statistical method, machine learning method, and expert analysis system etc [7]. The number of faults in the software cause to problems in system performance and the software containing many faults delivered to the user. So there is need of automated model which takes less time and predict the approximate faults existing in the system. Here machine learning method has been used which is based on neural network. This provided the approximate result near to the actual result already given [16].

Neural network is the techniques based on the human brains. It is collection of the number of artificial neurons. It can be customized according to the need and problems. As each artificial neuron takes a number of inputs and provide the single output [5]. An artificial neuron performed the complex calculation on input on the basis of used transfer function to produce desired output. As the size of neural network increases the complexity of the system also increases. Therefore there is need to use the number of layers, number of neurons, transfer function as less as possible. Neural network is capable to perform complex function in many fields like in pattern recognition [5], classification, speech recognition, vision and control system. Neural network is capable to solve the problems which could not solve by the human brains or by the conventional computer system E.g. data mining [11], image processing [12] and diagnostic systems [13].

At last it has been analyzed that the software industries have a high level of challenging problems to deliver the fault free software to clients or users. The delivered software should be reliable; it should be more performable as size and complexity of the software increases. Software bug prediction system is cost effective and improving software quality [19].

REVIEW OF LITERATURE

In this study the number of papers which are published in conferences, journal, book, transaction from 1990 to 2015 have been analyzed. These published papers have been categorized according to the model based on machine learning approach. This literature review excluded the paper which has not experimental results. Perfer and Selby used the classification tree on method level metrics and produced 79.3% accuracy of the model [2]. But it not supported longer to improve this model. The logistic regression, optimization set reduction (OSR) and classification tree used by the Briand *et al*. This model was not appropriates, it produced accuracy to 90% using OSR technique. Khoshgoftar *et al* [4] used fuzzy logic subtractive clustering methods to classify the data in to faulty and non-faulty classes. This model is made for the 10 method level metrics. Refurment used fuzzy rule based model for finding number of faults. This model is applied on the 11 method level metrics of a medical image system. Wang *et al* used artificial neural network for software fault prediction used on a large telecommunication system developed on c language having product metrics provided by matrix analyzer [2]. Xiang *et al* used the support vector machine (SVM) on the 11 method level metrics of medical image software. The published paper evaluate that it was not better for class level public dataset.

Ma *et al* studied the performance of various machine learning approach on public data set, this is method level metrics provided by the NASA. However it was failed to find a software tool which consist this approach. Boesticher analyzed the effect of dataset on software engineering and applied Naïve Bayes technique [2]. This provides the 94% accuracy for determining same class and nasty neighbor test set in public dataset provided by NASA. Bibi *et al* used regression classification to solve software fault prediction problem with a confidence interval. It had provided better regression error than standard regression. Turhan and Bener showed the assumption in Naïve Bayes algorithm is not preprocessed with PCA so they used balance parameters [2]. Mende and Kuchke focused on lines of codes metrics based software fault prediction model on public dataset of NASA. This model performed better in term of area under ROC curve parameter. Singh and Salaria [15] introduced a model based on machine learning approach using neural network which providing the accuracy of the model 88.09% on the class level public dataset ANT1.7. Mahajan *et al* [9] proposed a software fault prediction model using BR technique of neural network. It was machine learning approach that provided 92.44% accuracy of the model for class level public dataset ANT1.7.

DATA SET

The input dataset used in this model are object oriented class level public dataset. These dataset are taken from the PROMISE repository of empirical software engineering data *(http://promisedata.googlecode.com).* The dataset are available in nonnumeric or textual form so its need to prepare the data to use as input in neural network. It is known that neural network trained only with numeric data. Hence there is need to convert non numeric data in to numeric form. There are many techniques used for conversion like binary encoding, unary encoding and numbering classes. Here MS Excel has been used to convert textual form in to numeric data. The following table provides attribute information or metrics used in the public dataset [15].

Table -1 Attributes Information of Public DATASET

S. No.	Attributes / Inputs	Explanation	Suggested By
1.	WMC	Weighted methods per class	Chidamber and Kemerer [3]
2.	DIT	Depth of Inheritance Tree	Chidamber and Kemerer[3]
3.	NOC	Number of Children	Chidamber and Kemerer[3]
4.	CBO	Coupling between object classes	Chidamber and Kemerer[3]
5.	RFC	Response for a Class	Chidamber and Kemerer[3]
6.	LCOM	Lack of cohesion in methods	Chidamber and Kemerer[3]
7.	LCOM3	Lack of cohesion in methods	Henderson-Sellers12
8.	NPM	Number of Public Methods	Bainsy and Davis[1]
9.	DAM	Data Access Metric	Bainsy and Davis[1]
10.	MOA	Measure of Aggregation	Bainsy and Davis[1]
11.	MFA	Measure of Functional Abstraction	Bainsy and Davis[1]
12.	CAM	Cohesion Among Methods of Class	Bainsy and Davis[1]
13.	IC	Inheritance Coupling	Tang *et al*[16]
14.	CBM	Coupling Between Methods	Tang *et al*[16]
15.	AMC	Average Method Complexity	Tang *et al*[16]
16.	Ca	Afferent couplings	Martin[12]
17.	Ce	Efferent couplings	Martin[12]
18.	CC	Cyclomatic complexity	McCabe[13]
19.	Max(CC)	The greatest value of CC	McCabe[13]
20.	Avg(CC)	The arithmetic mean of the CC	McCabe[13]

IMPLEMENTATION TO DESIGN THE PROPOSED MODEL

In this work the neural network model has been created and designed to measure the performance using mean squared error (MSE) function which providing the better result in term of accuracy. As neural network works on the concept of accepting input, training the network, testing the sample by simulation. The 85% of Ant 1.7 dataset has been used for training and testing is done on the 15% 0f Ant dataset [9]. The ANT dataset classified in to two dataset training and sample (dataset for testing). The random 15% sample from Ant dataset has been ejected for testing dataset using the formula to generate the random index. Rand Array=ceil (1+ (745-1).* rand (100, 1))[9]. The remaining 85% of Ant dataset has been used for training.

Preprocessing and Post Processing

The training input dataset has been preprocessed to scale the input matrix in the range of -1 to 1 using the formula

$$Y_{ij} = (Y_{\max} - Y_{\min}) * (X_{ij} - X_{\min}) \div (X_{\max} - X_{\min}) + Y_{\min}$$

This model designed using GDA (Gradient descent with adoptive learning), BR (Baysian Regilation) and LM (Levenberg Marquardt) techniques to train the neural network. The transfer function tangent sigmoid and pure linear transfer functions have been used at hidden layer and output layer respectively. There are one hidden layer and size of hidden layer is 33.

The following Fig. 1 shows the neural network configuration information. The following Fig. 2 providing the execution and result in Matlab 2013.

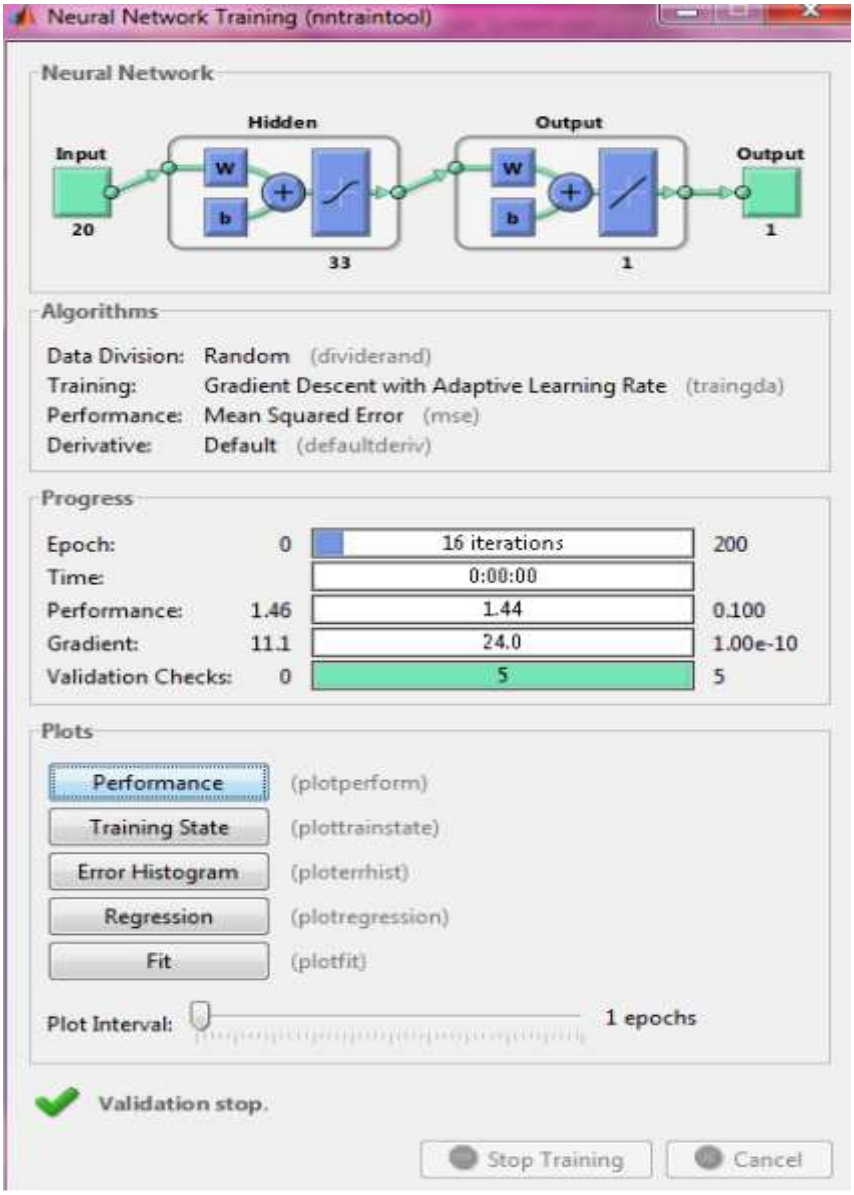

Fig. 1 Neural Network Training GUI

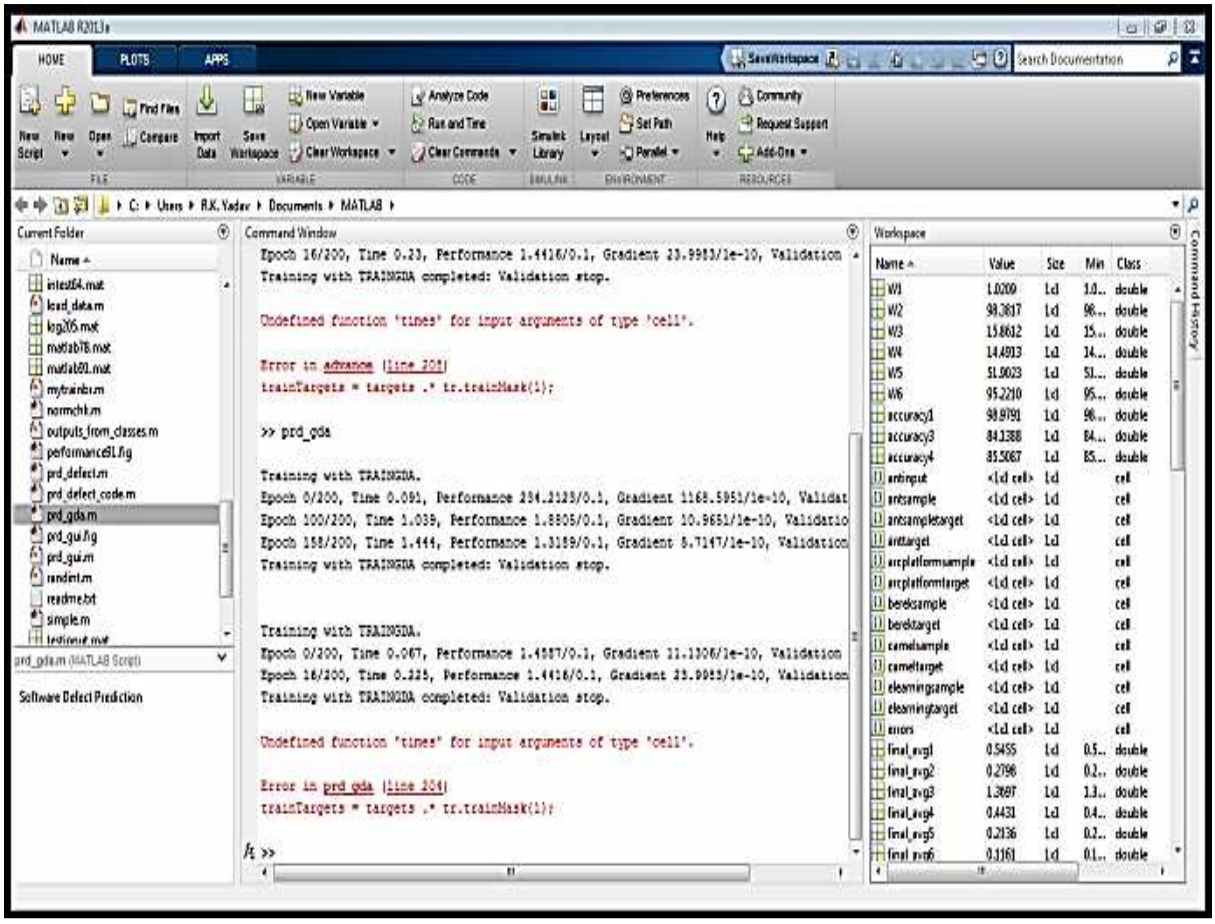

Fig. 2Execution process of the proposed model in MATLAB

RESULT AND COMPARISON ANALYSIS

For ANTver1.7 Datasets the Actual and Predicted Bug

Table -2 shows the predicted bug corresponding to the actual bug in the classes of the datasets. Record number is the random indices find out from the actual datasets with the help of random indices generator array of size 100 rows of 1 column.

From the above table it has been analyzed that the result of predicted bug output of GDA. BR and LM algorithm is calculated by the testing dataset of Ant1.7. Actual bug is the output which is already given in the Ant1.7 dataset. As a testing result the average predicted output using GDA, LM and BR are 0.545513, 0.5649195 and 0.6196424 respectively. Given that average actual output for testing dataset Ant1.7= 0.5400

General formula for calculating percentage accuracy of GDA, BR and LM algorithm are given below:

Percentage Error= |average predicted output-average actual output|/average actual output*100 [9]

(i) Accuracy of GDA algorithm:

Percentage Error = |0.545513 - 0.5400|/0.5400 * 100 = 1.024074%

Percentage Accuracy= 100 - 1.024074% = 98.97

(ii) Accuracy of BR algorithm:

Percentage Error = |0.5649195 - 0.5400|/0.5400 * 100 = 4.614722%

Percentage Accuracy= 100 - 4.614722% = 95.38%

(iii) Accuracy of LM Algorithm:

Percentage Error = |0.6196424 - 0.5400|/0.5400 * 100 = 14.748593%

Percentage Accuracy= 100 - 14.748593% = 85.25%

The following Table -3 shows the comparison result of the proposed model and it is analyzed that the accuracy of proposed model using GDA techniques is greater than the other techniques BR and LM for ant datasets. From the table -3 calculations, so it has been observed that designing of software bug prediction model using GDA technique provides the higher accuracy than techniques [9] like BR and LM.

Note: This is shown that the neural network based bug prediction system using GDA techniques is better system than BR and LM based models.

Table -2 Predicted Bug corresponding Actual Bug

Sr.	Record	Actual	Predicted	Predicted	Predicted	Sr.	Record	Actual	Predicted	Predicted	Predicted
1	608	0	-0.13965	0.207984	0.097988	51	207	0	-0.11941	0.151031	-0.07361
2	675	0	-1.21596	-0.0495	-2.87044	52	507	0	-0.24676	0.114256	0.068069
3	96	0	-0.05277	0.247892	0.275674	53	331	0	0.701153	0.228145	-0.21602
4	681	0	-0.11811	0.285726	-0.17133	54	122	0	-0.09474	0.254947	0.136278
5	472	0	0.285041	0.212626	0.094878	55	90	0	0.025808	0.310194	0.170748
6	74	0	-1.53603	0.317994	-0.25844	56	372	4	1.538865	0.562457	1.047118
7	209	0	-0.22445	0.076408	0.140634	57	716	0	2.434129	0.46279	-0.03102
8	408	0	0.029051	0.292781	0.167947	58	255	0	0.740494	0.335037	-0.54654
9	64	6	3.581547	1.397309	1.735874	59	437	1	0.210388	0.745827	0.862717
10	719	0	-0.28106	-0.10414	-0.60366	60	168	0	0.361601	0.957596	-0.26016
11	119	0	0.886165	0.292749	0.001648	61	560	6	3.267603	2.06538	0.739828
12	724	0	-0.31521	0.038991	0.130673	62	191	0	-0.44862	-0.00344	0.259179
13	714	0	0.181777	0.674014	0.054281	63	378	0	1.085374	0.946486	0.865572
14	363	0	-0.25176	0.111971	0.067472	64	522	3	0.310988	1.469272	1.874704
15	597	1	0.104797	3.857747	0.584896	65	664	1	-0.61054	-0.41488	0.206739
16	107	0	8.776178	0.337989	1.645744	66	-5	1	0.816135	0.428421	-0.01306
17	315	0	0.113252	0.242712	-0.00369	67	409	0	-0.14028	0.25732	0.336718
18	683	1	0.348626	1.39504	0.36483	68	105	0	0.549674	0.605076	2.480168
19	591	0	-0.31736	0.132873	0.001049	69	113	0	-0.34709	0.183026	0.055194
20	715	0	-0.61349	0.431657	0.148418	70	193	3	1.731835	1.559254	3.182096
21	489	1	0.734295	1.079813	0.537585	71	627	0	-0.47256	-0.02015	0.950938
22	28	0	1.105606	1.261218	1.442753	72	110	0	0.656704	0.230562	-0.01864
23	633	0	0.409662	0.317317	-0.16703	73	607	0	0.452434	0.233531	0.174338
24	696	0	-0.22445	0.076408	0.140634	74	183	1	-0.41804	0.353103	0.60126
25	506	0	2.455554	0.011636	-0.1736	75	693	1	1.523191	0.042523	0.684113
26	565	0	0.650624	0.414242	0.293202	76	262	4	1.512755	0.96698	-1.77521
27	554	0	1.454326	1.072593	0.838033	77	148	0	-0.08509	0.319187	0.176864
28	293	1	0.002737	0.617582	15.25707	78	188	0	-0.33207	0.020443	-0.26828
29	558	0	1.163242	0.557905	0.176552	79	460	0	0.297795	1.147923	0.39675
30	129	0	-0.32408	0.431525	-0.04377	80	354	1	0.173184	1.438076	1.278197
31	527	0	0.1587	0.094566	0.235554	81	263	0	0.33324	0.214441	0.084025
32	25	0	1.892478	0.639642	0.176114	82	620	0	0.25723	0.11507	0.346055
33	208	3	2.601005	-0.53971	11.61586	83	300	0	0.16683	0.027812	0.345309
34	36	0	1.428806	0.475444	0.779296	84	410	0	0.296097	0.386845	-0.65296
35	395	1	1.116454	1.153601	1.259305	85	684	0	0.104498	0.141806	0.027432
36	614	0	-0.21917	0.078755	0.141554	86	214	1	0.065931	1.479849	0.910492
37	518	0	2.375533	-0.13306	-0.17967	87	180	0	-0.02454	0.207553	0.081312
38	237	0	-0.4702	1.004961	-0.10142	88	562	1	-1.19087	0.456473	0.139036
39	708	0	0.838772	0.136294	0.359103	89	93	0	0.074013	0.157137	0.053273
40	27	0	-0.63521	0.849762	0.03574	90	424	0	1.433585	0.499563	-0.05633
41	328	2	-1.43437	0.165375	-0.03463	91	58	3	3.947953	0.679799	3.53968
42	285	0	0.811302	0.282689	0.101281	92	42	0	-0.17297	0.168506	0.697581
43	571	0	0.240642	0.787866	0.3722	93	396	0	-0.14783	0.201637	0.091267
44	593	1	1.299295	1.911706	2.355454	94	581	0	0.113623	0.21556	0.226597
45	141	1	1.272101	2.376585	2.699214	95	312	1	0.097784	0.719909	1.080108
46	366	0	0.394207	1.486041	1.139526	96	98	0	1.562014	1.136674	0.454579
47	333	4	0.595494	1.679585	-0.013	97	425	0	-0.19022	0.16109	-0.55119
48	482	0	1.455601	0.405238	-0.05881	98	351	0	0.872553	0.262891	0.100087
49	529	0	1.288392	2.655653	1.862932	99	10	0	0.572335	0.459189	-2.34548
50	563	0	0.045455	0.745916	0.795299	100	252	0	-0.41827	0.351805	0.601557
							AVG	0.54	0.545513	0.5649195	0.6196424

Table -3 Comparison for Accuracy of Different Bug Prediction Models

Sr. No.	Bug Prediction Model using different techniques	Input Data set	Accuracy (%)
1	Levenberg Marquardt (LM) Based Model	Ant 1.7	85.25
2	Baysian Regilation (BR) algorithm Based Model	Ant 1.7	95.38
3	Gradient Descent Adoptive Learning Back-propogation Algorithm (GDA) Based Model	Ant 1.7	98.97

Performance Graph of Neural Network

The train structure keeps track of several variables during the course of training, such as the value of the performance function, the magnitude of the gradient, etc. It can use the training record to plot the performance progress by using the plot performance command.

The property training best epoch indicates the iteration at which the validation performance reached a minimum. The training continued for 16 more iteration before the training stopped. This Fig. does not indicate any major problems with the training. The validation and test curves are very similar. If the test curve had increased significantly before the validation curve increased, then it is possible that some over fitting might have occurred.

Validation define as the post analysis of the network means testing on the trained data that the system achieving the goal or not if achieving then this is stop the training and show the result of MSE [9]. The values of mean squared error for each iteration shown in the Fig. 3.

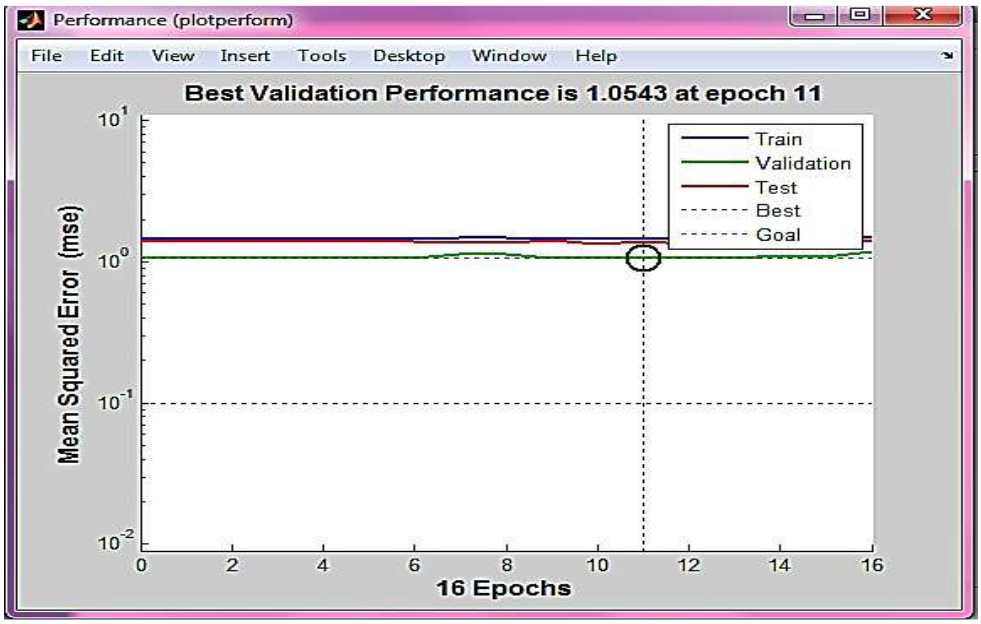

Fig. 3Performance measure v/s number of iteration

CONCLUSION

It is concluded that the proposed software bug prediction model using GDA techniques providing the better result than BR and LM techniques based models. It has been seen in the experimental result that the predicted accuracy of the model is 98.97% using Gradient Descent Adoptive learning Back-propagation techniques (GDA). As a result, it is found that machine learning models based on GDA technique of neural network is providing the superior results. It has been analyzed that this techniques is faster techniques than other existing techniques [14] and [8]. It is also observed that this providing the simulation result in less number of iteration than other existing models [14] and [8]. It provides the result only after 200 iterations which is smaller than the 500 no of iteration used by the existing models.

In future, the further study about the other fast and higher performing techniques are required to improve the performance of the model. The different techniques may produce better performance in terms of accuracy. By using class level metrics, more studies can be conducted on fault prediction models. So that this improved model may become capable to simulate the different similar type of wild public datasets and another private datasets.

REFERENCES

[1] J Bansiya and CG Davis, A Hierarchical Model for Object-Oriented Design Quality Assessment, *IEEE Transactions on Software Engineering*, **2002**, 28 (1), 4-17.
[2] C Catal and B Diri, A systematic Review of Software Fault Prediction Studies, *Journal of Expert Systems with Applications*, **2009**, 36 (4), 7346- 7354.
[3] SR Chidamber and CF Kemerer, A Metrics Suite for Object Oriented Design, **1994**, 20(6), 476-493.
[4] KO Elish and MO Elish, Predicting Defect-Prone Software Modules Using Support Vector Machines, *Journal of Systems and Software, ACM*, **2008**, 81(5), 649-660.

[5] M Egmont-Petersen, D de Ridder and H Handels, Image Processing with Neural Networks - A Review, *Pattern Recognition*, **2002,** 35(10), 2279-2301.
[6] NE Fenton and M Neil, A Critique of Software Defect Prediction Models, *IEEE Transactions on Software Engineering*, 1999, 25(5), 675- 689.
[7] I Gondra, Applying Machine Learning to Software Fault-Proneness Prediction, *ACM Journal of Systems and Software*, 81(2), **2008**, 186- 195.
[8] Y Jiang, B Cukic and T Menzies, Fault Prediction using Early Lifecycle Data, *18th IEEE International Symposium on Software* Reliability, Trollhattan, **2007**, 237-246.
[9] R Mahajan, SK Gupta and RK Bedi, Design of Software Fault Prediction Model using BR Technique, *Elsevier Procedia Computer Science*, **2015**, 46, 849 – 858.
[10] A Mahaweerawat, P Sophasathit and C Lursinsap, Software Fault Prediction using Fuzzy Clustering and Radial Basis Function Network, *Proceedings of International Conference on Intelligent Technologies*, Chulalongkorn University, Bangkok, Thailand, **2002**, 304-313.
[11] Mahaweerawat, P Sophasathit, C Lursinsap and P Musilek, Fault Prediction in Object-Oriented Software using Neural Network Techniques, *Advanced Virtual and Intelligent Computing Center (AVIC)*, Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand, **2004**, 1-8.
[12] R Martin, Object –Oriented Design Quality Metrics-An Analysis of Dependencies, *Workshop on Pragmatic and Theoretical Directions in Object –Oriented Software Metrics, Cranbrook Road Green Oaks, IL*, **1994**, 1-8.
[13] TJ McCabe, A Complexity Measure, *IEEE Transactions on Software Engineering*, **1976**, 2(4), 308-320.
[14] BH Sellers, *Object-Oriented Metrics, Measures of Complexity*, Prentice Hall, **1996**.
[15] R Shatnawi, Improving Software Fault-Prediction for Imbalanced Data, *IEEE Proceedings of International Conference on Innovations in Information Technology*, Abu Dhabi, **2012**, 54-59.
[16] M Singh and DS Salaria, Software Defect Prediction Tool based on Neural Network, *International Journal of Computer Applications*, **2013**, 70(22), 22-27.
[17] MH Tang, MH Kao and MH Chen, An Empirical Study on Object-Oriented Metrics, Proceedings of Metrics, 1999, 242-249.
[18] F Xing, P Guo and MR Lyu, A Novel Method for Early Software Quality Prediction Based on Support Vector Machine, *16th IEEE International Symposium on Software Reliability*, *Beijing*, **2005**, 37(6), 4537-4543.
[19] J Zheng, Cost-Sensitive Boosting Neural Networks for Software Defect Prediction, *Journal of Expert Systems with Applications: ACM Digital Library*, *Proceedings of Sixth International Symposium on Software Metrics*, **1999**, 242-249.