International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:7, 2008

# Alternative Methods to Rank the Impact of Object Oriented Metrics in Fault Prediction Modeling using Neural Networks

Kamaldeep Kaur, Arvinder Kaur, and Ruchika Malhotra

***Abstract*****—**The aim of this paper is to rank the impact of Object Oriented(OO) metrics in fault prediction modeling using Artificial Neural Networks(ANNs). Past studies on empirical validation of object oriented metrics as fault predictors using ANNs have focused on the predictive quality of neural networks versus standard statistical techniques. In this empirical study we turn our attention to the capability of ANNs in ranking the impact of these explanatory metrics on fault proneness. In ANNs data analysis approach, there is no clear method of ranking the impact of individual metrics. Five ANN based techniques are studied which rank object oriented metrics in predicting fault proneness of classes. These techniques are i) overall connection weights method ii) Garson's method iii) The partial derivatives methods iv) The Input Perturb method v) the classical stepwise methods. We develop and evaluate different prediction models based on the ranking of the metrics by the individual techniques. The models based on overall connection weights and partial derivatives methods have been found to be most accurate.

***Keywords*****—**Artificial Neural Networks (ANNS)**,** Backpropagation**,** Fault Prediction Modeling.

## I. Introduction

OBJECT oriented (OO) paradigm provides viable solution to the problems existing in the software industry and is expected to result in development of software which is less fault prone, reusable, maintainable, easily modifiable and less brittle. The advent of OO paradigm requires new metrics for quantifying the software development process as traditional product metrics are inadequate to measure reusability, inheritance and polymorphism. Several OO metrics have been proposed in literature [1] and there is high research activity and interest in validating the usefulness of the proposed metrics in quality, maintainability and effort modeling. [2,3].Popular techniques for data analysis in these studies are multiple regression, logistic regression and Multivariate Adaptive Regression Splines(MARS). Machine learning techniques like decision trees and neural networks are also common. In this paper we use multi layer perceptrons neural networks trained with backpropagation algorithm for data analysis in fault prediction modeling of java projects.

Neural networks have a close analogy to non-parametric statistical inference and are very powerful in modeling non linear relationships. In contrast to some statistical techniques they do not require to specify the relationships between inputs and outputs a priori. ANNs tend to be useful in quality modeling because, the number of inputs(object oriented metrics) are fairly large, many of the metrics are relevant but predictive information lies in a lower dimensional subspace. A quality prediction neural model should contain only the most significant explanatory metrics which have an impact above a fixed threshold. This allows the size of the neural network to be reduced and improves generalization. However, the determination of significant input metrics is the most difficult and criticized aspect of ANN modeling. With neural networks, there is no clear method of ranking input metrics. We study here some alternative rational ways to rank the impact of individual OO metrics on fault proneness using neural networks. We use some basic methods for analyzing the weights or inputs of neural networks, although some advanced methods like Fuzzy logic and Bayesian analysis can be applied for neural network weight interpretation.

This paper is organized as follows. Section II presents related work in this area. Section III presents the research background. Section IV contains the neural network modeling method and architecture. In section V we present five approaches for ranking the importance of individual OO metrics. In section VI we develop and evaluate neural models based on the obtained rankings and discuss the results obtained.

## II. Related Work

We performed a review of literature on use of ANNs for predicting quality models using metrics both in procedural and object oriented paradigm. Khoshgoftarr et al. used one hidden layer MLP with backpropagation training algorithm as a data analysis tool for software quality modeling of a large telecommunication system[4]. Their quality model was based on call graph metrics and control flow graph metrics as predictor variables to predict the group membership of a software module. The modules were classified as fault prone

Manuscript received April 20, 2006

Kamaldeep.Kaur, is with Institute of Information Technology and Management, Affiliated to GGS Indrapratha Delhi, India (corresponding author; phone: 91-011-55388484; e-mail: kdkaur99@yahoo.co.in).

Dr. Arvinder Kaur, is with GGS Indraprastha University , Delhi, India (e-mail: arvinderkaurtakkar@yahoo.com.)

Ruchika Malhotra is with Indira Gandhi Institute of Technology , GGS Indraprastha University, Delhi, India (e-mail: ruchikamalhotra2004@yahoo.com)

and not fault prone. Thwin and Quah have used Ward and Generalized Regression Neural Networks(GRNN) for predicting software development faults[5] and software readiness[6]. There models are based on OO metrics[5,6] as predictor variables and number of faults as response variables. More recently, Gyimothy et al. [7] have used a neural network developed at their university to predict the fault proneness of classes in an open source software. Except for [6] none the above studies have considered the ranking of impact of individual metrics on fault proneness when conducting a data analysis using neural networks. In [6] analysis of weights method is followed for finding the impact of individual metrics but the algorithmic details are not divulged . The general approach in these papers [4,5] is to identify orthogonal metrics by Principal Component Analysis for dimensionality reduction but impact of individual metrics using neural technique is not studied. The input metrics in our study are decorrelated and we look for other variable selection techniques for ANN modeling.

## III. Research Background

### A. Metrics Studied

We choose suite of Chidamber and Kemerer metrics[1] in our study as independent variables.

WMC (Weighted Methods per Class). The WMC is the number of methods defined in each class, weighted by their complexity. More precisely, WMC is defined as being the number of all member functions and operators defined in each class.

DIT (Depth of Inheritance Tree).The DIT is defined as the length of the longest path from the class to the root in inheritance hierarchy.

RFC (Response For a Class).This the response for a class coupling metric. The response set of a class consists of the set M of the methods of the class and the set of methods invoked directly by methods in M (i.e., the set of methods that can potentially be executed in response to a message being received by that class). RFC is the number of methods in the response set of the class.

NOC (Number Of Children). The NOC is the number of direct descendants for each class.

CBO (Coupling Between Object classes). A class is coupled to another one if it uses its member functions and/or instance variables. CBO provides the number of classes to which a given class is coupled.

LCOM(Lack of Cohesion on Methods).The number of pairs of member functions without shared instance variables, minus the number of pairs of member functions with shared instance variables. However, the metric is set to zero whenever this subtraction is negative.

LOC (Lines Of Code). The LOC of a class is the number of all nonempty, non comment lines of the body of the class and all of its methods.

### B. Data Collected

To analyze metrics chosen for this work, their values are computed for twelve different systems. These systems are developed by under graduate engineering students and Masters of Computer Application students at School of Information Technology, of our University. The systems were developed using Java programming language over duration of four months. The aim was to teach students system analysis and design techniques as part of their course curriculum. All students had experience with Java language and thus they had basic knowledge necessary for this study.

The students were divided into 12 teams of four students each. Each team developed a medium-sized system such as flight reservation, chat server, proxy server etc. The development process used waterfall model. Documents were produced at each phase of software development. Faults were reported to the developers. A separate group of students having prior knowledge of system testing under the guidance of senior faculty were assigned the task of testing systems according to test plans.

The following relevant data was collected:

1. The design and source code of the java programs
2. The faulty data found by the testing team.

The 12 systems under study consist of 136 classes (39 KLOC) out of which 85 are system classes and 51 standard library classes available in java language. These classes contain functions to manipulate files, strings, lists, hash tables, frames, windows, menus, threads, socket connection etc.

Table I shows the descriptive statistics of the data for 85 system classes. The quantities of interest are mean, median, Standard Deviation, Inter-quartile range and number of classes in which a given metric is greater than zero.

TABLE I
DESCRIPTIVE STATISTICS FOR OUR DATA OF 85 SYSTEM CLASSES

| Metric | Mean | Median | Standard Deviation | IQR | N>0 |
|---|---|---|---|---|---|
| DIT | 0.353 | 0 | 0.612 | 1 | 25 |
| NOC | 0.282 | 0 | 0.959 | 0 | 10 |
| LOC | 114.423 | 64 | 151.658 | 85 | 85 |
| CBO | 1.059 | 1 | 1.313 | 1 | 54 |
| WMC | 5.964 | 3 | 6.889 | 7 | 80 |
| RFC | 13.4 | 7 | 16.357 | 17 | 76 |
| LCOM | 25.564 | 0 | 65.662 | 13.25 | 35 |

## IV. Neural Network Modeling

### A. Architecture Selection

The multi-layer feed-forward network was used in this experiment. The independent variables in this study are the seven object oriented metrics from 85 system classes, which are inputs to the neural network and dependent variable is the fault proneness of a class which is the network output. Outliers found to be influential were removed from the data set. The neural network was trained using backpropagation algorithm. This algorithm generally works best when the network inputs and targets are normalized all the input metrics and actual outputs were scaled with min-max normalization technique. The network architecture consisted of a single

hidden layer. The number of hidden neurons was empirically determined by partitioning the data into training, validation and test subsets in the ratio 3:1:1. A network with two hidden neurons gave the best prediction results. Mu was set equal to 0.001. The training algorithm used was backpropagation with Bayesian Regularization, which has already been found the best for software engineering applications in existing studies [8].

TABLE II
NEURAL NETWORK ARCHITECTURE

| | |
|---|---|
| Hidden Layers | 1 |
| Hidden neurons | 2 |
| Activation function hidden layer | Tansig |
| Output neurons | 1 |
| Activation function output neuron | Purelin |
| Mu | 0.005 |

After the architecture ha been selected the network is trained with data from all 85 system classes to rank the impact of metrics.

*B. Metric Ranking Methods*

1. Univariate Analysis

In this approach we used only one metric in the neural network at a time and created seven models to find out the metrics that are most important in predicting fault proneness. The following table shows the results of correlation of network outputs with actual fault proneness for individual metrics.

TABLE III
CORRECTNESS OF INDIVIDUAL METRICS IN PREDICTING FAULT PRONENESS

| Metric | r | p-value | Correctness |
|---|---|---|---|
| DIT | 0.1974 | 0.0701 | All classes non faulty |
| NOC | 0.1382 | 0.0207 | All classes non faulty |
| LOC | 0.6142 | 0.0000 | 67.57% |
| CBO | 0.3735 | 0.0004 | 40.5% |
| WMC | 0.5275 | 0.0000 | 54.05% |
| RFC | 0.7169 | 0.0000 | 78.37% |
| LCOM | 0.4829 | 0.0000 | 45.9% |

Table IV gives the sum squared training error, sum squared weights and effective number of parameters for individual metrics by training the network with a Bayesian Regularization Algorithm.

TABLE IV
UNIVARIATE MODELING WITH INDIVIDUAL METRICS AS PREDICTORS

| Metric | Sum Squared Error | Sum squared weights | Effective number of parameters |
|---|---|---|---|
| DIT | 20.152 | 0.8401 | 1.813 |
| NOC | 20.885 | 0.183 | 0.985 |
| LOC | 13.016 | 19.209 | 3.909 |
| CBO | 17.911 | 2.577 | 2.408 |
| WMC | 15.082 | 2.817 | 2.579 |
| RFC | 10.159 | 20.394 | 3.687 |
| LCOM | 16.038 | 9.336 | 3.107 |

The metrics that have high correctness in predicting fault proneness result in lower sum squared error and high sum squared weights.

The univariate analysis method considers the impact of metrics when they are isolated from other metrics. Although univariate analysis provides some insight regarding the usefulness of individual metrics it does not capture the interaction between various metrics. Rest of the methods we consider are multivariate methods wherein all metrics are input to the neural network.

2. Over all Connection Weights based Wrapper Method

This method is also called ANNIGMA[9]. This method calculates the product of raw input-hidden and output-hidden connection weights between each input neuron and output neuron and sums the products across all hidden neurons. The metric having the lowest overall connection weight is considered to be the least significant and is eliminated from the neural network. The order of metric elimination is considered to be the ranking of metrics from lowest to highest. The confusion matrix is analyzed after each elimination. If the sensitivity and specificity does not improve further the algorithm can be stopped for model prediction.

3. Garson's Method

The method for ranking the relative importance of input variables was first proposed by Garson[10]. This method essentially involves partitioning the hidden output connection weights of each hidden neurons into components associated with each input to the neural network. As contrast to overall connection weights method this method uses the absolute values of connection weights.

i) For each hidden neuron i , multiply the absolute value of the hidden –output layer connection weight by the absolute value of the hidden input layer connection weight. Do this for each input metric j.

ii) For each hidden neuron , divide each $P_{ij}$ obtained in step i) by the sum of all $P_{ij}$ s to obtain Qijs

iii) For each input obtain the sum $S_j$ by summing the $Q_{ij}$s

iv) Divide the sum $S_j$ by the sum of $S_j$ for all input variables . This gives the relative importance or contribution of each of the metrics.

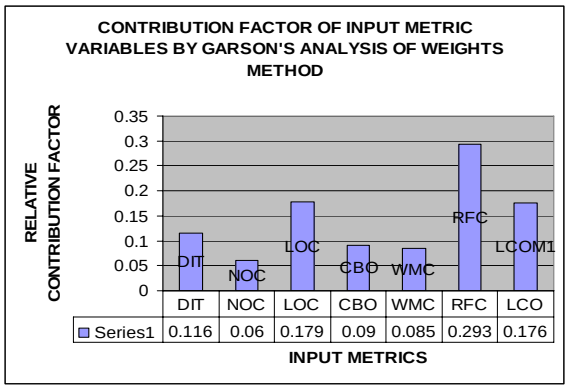



Fig. 1 Metric Ranking by Garson's Algorithm

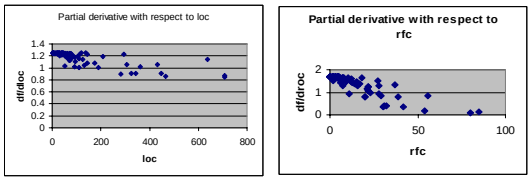
4. Partial Derivative Method

This is also a neural network analysis technique given by Ruck [11]. In this technique the partial derivative of the network output with respect to each of the input metric is calculated. The sum squared of each of the partial derivatives is obtained. The metrics that have large positive partial derivative values at a large number of exemplar points derivative value is considered to be the one that influences fault proneness the most. The partial derivative of the neural network output $y_i$ with respect to input $x_i$ is given by

$$\frac{\partial y_i}{\partial x_i} = \sum_{h=1}^{nh} w_{ho} * (1 + I_h) * (1 - I_h) * w_{hi} \qquad (1)$$

$I_h$ is the response of $h^{th}$ neuron in the hidden layer , who is the connection weight between $h^{th}$ hidden neuron and output neuron, $w_{hi}$ the connection weight between $i^{th}$ input and $h^{th}$ hidden neuron

(The above equation is valid only when there is a single hidden layer and the hidden layer activation function is hyperbolic tangent and output layer activation function is purely linear.) The partial derivative is obtained at all the 85 exemplars for each of the seven inputs and scatter plots are drawn to observer the influence of input variable on output. The sum of the squares of partial derivatives of input variables gives us the impact of each input variable.



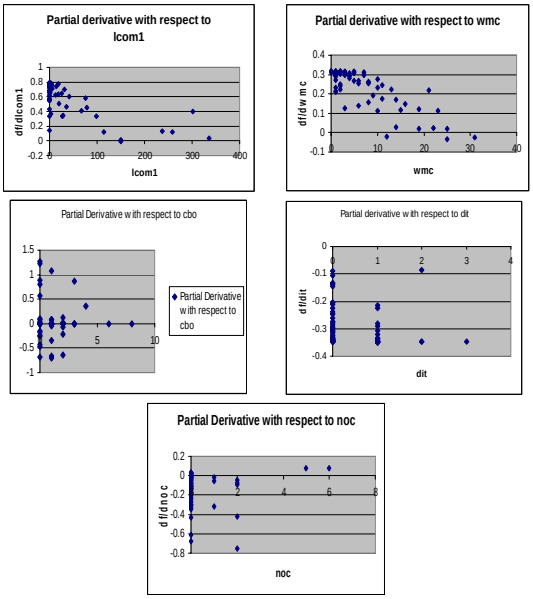




Fig. 2 Scatter Plots of Partial derivative of Neural Network response with respect to each of the explanatory OO metrics.

From the scatter plots for partial derivatives of each of the following metrics the following observations can be made.

i) The partial derivative of network output with respect to DIT and NOC is negative.

ii) The partial derivative of network output with respect to LOC, RFC, WMC and LCOM has positive values.

iii) The partial derivative of network output with respect to CBO does not have a precise positive or negative direction and cannot explain the output

5. Perturb Method

This method is also called sensitivity analysis in some research literature. It is basically a method for extracting cause and effect relationship between inputs (metrics in our case) and outputs (fault proneness).The method involves adding white noise or dither to one of the input metrics at a time while keeping all other inputs untouched. The network learning is disabled during this operation such that the network weights are not affected. The following table shows the raw sensitivity for different percentages of dither. We consider 50% perturbation values for ranking.

TABLE V
RAW SENSITIVITY OF INPUT METRICS TO WHITE NOISE

| Dither Percentage Metric | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| DIT | 0.102 | 0.187 | 0.259 | 0.319 | 0.372 |
| NOC | 0.148 | 0.313 | 0.492 | 0.685 | 0.882 |
| LOC | 0.772 | 0.301 | 0.702 | 0.034 | 0.325 |
| CBO | 1.593 | 2.912 | 3.862 | 4.795 | 5.677 |
| WMC | 1.818 | 3.533 | 4.447 | 6.113 | 8.143 |
| RFC | 5.349 | 10.443 | 15.349 | 18.453 | 20.724 |
| LCOM | 0.781 | 1.623 | 2.336 | 2.858 | 3.269 |

6. Classical Stepwise Methods

We followed two classical stepwise methods:

a) Stepwise forward addition method

In the first step seven models are generated as in univariate analysis each using only one of the available metrics. Then six models are generated, combining the metric that resulted in smallest error (for a single input metric) with each of the remaining metrics. This procedure is repeated until all the metrics are added to the model. The order of integration of the metrics is their ranking.

b) Stepwise backward elimination

Seven models are generated using only six of the available metrics. The seventh missed out metric for which the resulting models gave the largest error, is the most important. Then six models are generated with five metrics, eliminating the most significant metric plus one of the six remaining metrics and so on. The order of metric elimination is their ranking.

## V. RESULTS

### *A. Ranking of the Metrics According to the Methods Employed*

TABLE VI
RANKING OF OO METRICS BY THE FIVE METHODS

| Method Metric | M1 | M2 | M3 | M4 | M5( a) | M5(b) |
|---|---|---|---|---|---|---|
| DIT | 7 | 4 | 7 | 7 | 5 | 7 |
| NOC | 6 | 7 | 6 | 6 | 3 | 6 |
| LOC | 2 | 2 | 2 | 5 | 2 | 3 |
| CBO | 5 | 5 | 5 | 3 | 7 | 2 |
| WMC | 3 | 6 | 3 | 2 | 4 | 4 |
| RFC | 1 | 1 | 1 | 1 | 1 | 1 |
| LCOM | 4 | 3 | 4 | 4 | 6 | 5 |

M1 : Overall Connection weights based wrapper approach
M2:Garson's Method
M3: Partial Derivatives Method
M4:Perturb Method
M5(a): Stepwise forward addition
M5(b): Stepwise backward elimination

### *B. Model Prediction and Evaluation*

We performed a 9 cross validation of the proposed neural network model. The 85 data exemplars were randomly split into 9 partitions (five partitions of 9 data points and four partitions of 10 data points).The classification results for the full model and for each of the five methods are given are given in Tables VII-XII. Table XIII gives sensitivity, specificity, proportion correct, true positive rate and J coefficient [13].

TABLE VII
CLASSIFICATION RESULTS BASED ON FULL MODEL

| Actual | Predicted | |
|---|---|---|
| | Low Risk | High Risk |
| Low Risk | 41 | 7 |
| High Risk | 11 | 26 |

TABLE VIII
CLASSIFICATION RESULTS BASED ON OVERALL CONNECTION WEIGHTS AND PARTIAL DERIVATIVES METHODS

| Actual | Predicted | |
|---|---|---|
| | Low Risk | High Risk |
| Low Risk | 44 | 4 |
| High Risk | 5 | 32 |

TABLE IX
CLASSIFICATION RESULTS BASED ON GARSON'S METHOD

| Actual | Predicted | |
|---|---|---|
| | Low Risk | High Risk |
| Low Risk | 39 | 9 |
| High Risk | 7 | 30 |

TABLE X
CLASSIFICATION RESULTS BASED ON PERTURB METHOD

| Actual | Predicted | |
|---|---|---|
| | Low Risk | High Risk |
| Low Risk | 34 | 14 |
| High Risk | 6 | 31 |

TABLE XI
CLASSIFICATION RESULTS BASED ON STEPWISE FORWARD ADDITION

| Actual | Predicted | |
|---|---|---|
| | Low Risk | High Risk |
| Low Risk | 37 | 11 |
| High Risk | 6 | 31 |

TABLE XII
CLASSIFICATION RESULTS BASED ON STEPWISE BACKWARD ELIMINATION

| Actual | Predicted | |
|---|---|---|
| | Low Risk | High Risk |
| Low Risk | 39 | 9 |
| High Risk | 5 | 32 |

TABLE XIII
ACCURACY OF PREDICTION MODELS

| Method | Metrics Selected | Sensitivity | Specificity | Proportion Correct | True positive Rate | J Coefficient |
|---|---|---|---|---|---|---|
| M1 & M3 | RFC, LOC ,LCOM, WMC | 86.40% | 91.60% | 89.40% | 88.80% | 0.78 |
| M2 | RFC , LOC, LCOM1 | 81.08% | 81.25% | 81.17% | 76.90% | 0.62 |
| M4 | RFC, WMC, CBO | 83.78% | 70.80% | 76.47% | 68.88% | 0.55 |
| M5 (a) | RFC, LOC ,NOC | 89.19% | 77.08% | 82.35% | 75% | 0.66 |
| M5 (b) | RFC,CBO,LOC,WMC | 86.40% | 81.25% | 83.5% | 78.05% | 0.67 |

## VI. CONCLUSION

The partial derivatives method and overall connection weights based wrapper approach give the best results. Both methods are found to be stable. These methods reduce the size and number of neural networks to be built and trained. The perturb method gives a low ranking to LOC which is an important metric. Stepwise methods give poor results as they select metrics found to be insignificant in univariate analysis , like NOC . Four methods identify RFC and LOC as the most significant metrics i.e., highly related to fault proneness. DIT and NOC receive a low rank from most of the five methods. Thus, ANNs can be used to rank the most significant metrics and build accurate prediction models based on subset of object oriented metrics.

## REFERENCES

[1] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, pp 476-493,1994

[2] V.R. Basili, et al,"A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, vol. 22, pp 751-761,1996

[3] L.C. Briand , Jurgen Wust, "Modeling Development Effort in Object Oriented Systems using Design Properties," *IEEE Transactions on Software Engineering*, vol. 27, no. 11, 2001

[4] T.M. Khoshgoftaar, E.B.Allen, J.P. Hudephol and S.J. Aud," E.B.Allen, J.P. Hudephol and S.J. Aud," ," Application of neural networks to quality modeling of a very large telecommunication system*", IEEE Transactions on Neural Network*s, vol.8,pp. 902-909,1997

[5] M. M. T. Thwin and T.-S. Quah,, "Application of Neural Networks for predicting Software Development faults using Object Oriented Design Metrics*", Proceedings of the 9th International Conference on Neural Information Processing*, November 2002, pp. 2312 – 2316.

[6] M. M. T. Thwin and T.-S. Quah,,, "Prediction of Software Readiness using Neural Networks," *ICITA2002*, ISBN:1-86467-114-9

[7] Tibor Gyimothy, Rudolf Fernec, and Istvan Siket , " Empirical Validaion of Object –Oriented Metrics on Open Source Software for Fault Prediction" , *IEEE Transactions on Software Engineering,* vol. 31, no. 10, October 2005

[8] K.K. Aggarwal, Y. Singh. P. Chandra,M.Puri, "Evaluation of various training Algorithms for Sofware Engineering Applications," *ACM SIGSOFT Software Engineering Notes* ,vol 30, no.4 July 2005.

[9] Dietrich Schusche, Chun Nan Hsu, Hung-Ju Huang , "A weight analysis based wrapper approach to Neural Nets feature subset selection," *IEEE Transactions on Systems, Man and Cybernetics*, vol.32,pp. 207-21,2002

[10] G.D. Garson, "Interpreting Neural Network Connection Weights," *AI Expert 6*, pp. 47–51, 1991.

[11] D.W. Ruck, S.K Rogers, M. Kabrisky, "Feature Selection using Multi Layer Perceptrons," in *Journal of Neural Network Computing*, vol 2, no . 2 , pp. 40-48 ,1990.

[12] S. Haykins, "A Comprehensive Foundation on Neural Networks ," Prentice Hall, 1999

[13] Khaled El Emam, "A Methodology for Validating Software Product Metrics," National Research Council Canada, Institute for Information Technology, ERB-1076.