Software Maintainability Prediction using Machine Learning Algorithms

Ruchika Malhotra[1] and Anuradha Chug[2]

[1]*Department of Software Engineering, Delhi Technological University, Delhi 110042, India*

[2]*University School of Information and Communication Technology, GGS IP University, Dwarka, New Delhi 110077*

ruchikamalhotra2004@yahoo.com, a_chug@yahoo.co.in

Abstract - **Software maintainability is one of the most important aspects while evaluating quality of the software product. It is defined as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to a changed environment. Tracking the maintenance behaviour of the software product is very complex. This is precisely the reason that predicting the cost and risk associated with maintenance after delivery is extremely difficult which is widely acknowledged by the researchers and practitioners. In an attempt to address this issue quantitatively, the main purpose of this paper is to propose use of few machine learning algorithms with an objective to predict software maintainability and evaluate them. The proposed models are Group Method of Data Handling (GMDH), Genetic Algorithms (GA) and Probabilistic Neural Network (PNN) with Gaussian activation function. The prediction model is constructed using the above said machine learning techniques. In order to study and evaluate its performance, two commercial datasets UIMS (User Interface Management System) and QUES (Quality Evaluation System) are used. The code for these two systems was written in Classical Ada. The UIMS contains 39 classes and QUES datasets contains 71 classes. To measure the maintainability, number of "CHANGE" is observed over a period of three years. We can define CHANGE as the number of lines of code which were added, deleted or modified during a three year maintenance period. After conducting empirical study, performance of these three proposed machine learning algorithms was compared with prevailing models such as GRNN (General Regression Neural Network) Model, ANN (Artificial Neural Network) Model, Bayesian Model, RT (Regression Tree) Model, Backward Elimination Model, Stepwise Selection Model, MARS (Multiple Adaptive Regression Splines) Model, TreeNets Model, GN (Generalized Regression) Model, ANFIS (Adaptive Neuro Fuzzy inference System) Model, SVM (Support Vector Machine) Model and MLR (Multiple Linear Regressions) Model which were taken from the literature. Based on experiments conducted, it was found that GMDH can be applied as a sound alternative to the existing techniques used for software maintainability prediction since it assists in predicting the maintainability more accurately and precisely than prevailing models**.

Keywords: **GMDH (Group Method of Data Handling), Genetic Algorithms, Probabilistic Neural Network (PNN), Software Maintainability, Software Maintainability Prediction Metrics and Modeling.**

1. INTRODUCTION

Software maintainability means the ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to a changed environment [1]. The change in the software is required to meet the changing requirements of customers which may arise due to many reasons such as change in the technology, introduction of new hardware or enhancement of the features provided etc. Producing software which does not need to be changed is not only impractical but also very uneconomical. This process of changing the software which has been delivered is called software maintenance. The amount of resource, effort and time spent on software maintenance is much more than what is being spent on its development. Thus, producing software that is easy to maintain may potentially save large costs and efforts. One of the main approaches in controlling maintenance cost is to monitor software metrics during the development phase. It is a matter of interest for researches to measure various attributes of software design in terms of inheritance, coupling, cohesion etc and predict its maintenance behaviour on the basis of their values. The problem of predicting the maintainability of software is widely acknowledged in the industry and much has been written on how maintainability can be predicted by using various tools and processes at the time of designing with the help of software design metrics [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Studies have been conducted and found the strong link between Object Oriented software metrics and its maintainability. They have also found that these metrics can be used as predictors of maintenance effort. Accurate prediction of software maintainability can be useful because of the following reasons:

(a). It helps project managers in comparing the productivity and costs among different projects.

(b). It provides managers with information for more effectively planning the use of valuable resources.

(c). It helps managers in taking important decision regarding staff allocation.
(d). It guides about maintenance process efficiency.
(e). It helps in keeping future maintenance effort under control.
(f). The threshold values of various metrics which drastically affect maintainability of software can be checked and kept under control so as to achieve least maintenance cost.
(g). It enables the developers to identify the determinants of software quality so that they can improve design and coding.
(h). It helps practitioners to improve the quality of software systems and thus optimize maintenance costs.

To measure the maintainability we first find out the "change effort". It is defined as "how much amount of average efforts are required to add, change or delete existing classes". Software maintenance is very important as it consumes 70% of the time of any product's life and indeed it is very challenging. Despite this fact it is poorly managed because we really do not have good measures of software maintainability. The fundamental reality is stated by Morco [17] in the year 1982 in his book "Controlling Software Projects" that "you cannot control what you cannot measure". It really highlights the importance of a good measurement of software maintainability so that we can control it.

To measure the various features of object oriented paradigm such as inheritance, cohesion, coupling, memory allocation etc different metrics are carefully selected. We have studied various metrics available in literature and selected only those software metrics that have a strong relationship with software maintainability and used them while constructing our model for prediction of object oriented software maintainability. These metrics are Weighted Methods per Class (WMC), Depth of Inheritance (DIT), Number of Children (NOC), Lack of Cohesion in Methods (LCOM), Response For a Class (RFC), Message Passing Coupling (MPC), Data Abstraction Coupling (DAC), Number of Local Methods (NOM), and (SIZE1) traditional line of code, (SIZE2) total number of attributes and methods of a class. We divided our data into three parts. 60% of the data is used for training i.e. machine learn from the data patterns using specified algorithm, 20% of the data is used for validation and 20% of the data is used for testing. From the literature it is verified that this is the commonly accepted proportion used by most researchers and practitioners [6, 8, 9, 11, 12, 13, 14, 16, 18, 19].

The relationships between static software metrics and its maintainability is very complex and non linear, hence conventional statistical techniques based models, which are purely based on quantity, would not help much to the problem. Instead, use of machine learning algorithms to establish the relationship between metrics and maintainability would be much better approach as these are based on quantity as well as quality. Three machine learning algorithms were proposed and evaluated in this study. First proposed method is very powerful architecture called Group Method of Data Handling (GMDH). The main idea behind GMDH is that it tries to build a function (called a polynomial model) which would behave in such a way that the predicted value of the output would be as close as possible to the actual value of the output. The GMDH network is implemented with polynomial terms in the links and a genetic component to decide how many layers are to be built. The result of training at the output layer can be represented as a polynomial function of all or some of the inputs. Next proposed model was Genetic Algorithms (GA) which was based on the principles of Darwin's evolution theory. Over many generations, the "fittest" individuals tend to dominate the population. In predictions based problems, GA try to discover an optimal solution by simulating the evolution theory. For Predicting the object oriented software maintainability, the genetic algorithms start their job by first selecting a set of software metrics, which is constituted with collection of genes (solutions). GA then uses natural selection and genetics as a basis to search for the optimal gene and a set of software metrics that give the best classification rate. Third proposed model in this study is Probabilistic Neural Network (PNN) which is based on neural network. Neural network technology mimics the human brain's own problem solving process. As the human beings use their knowledge from earlier experiences to solve new problems or face situations, the neural network also considers earlier solved examples to create a scheme of "neurons" which makes new choices, classifications and predictions.

In this study we have compared performances of above mentioned machine learning algorithms with other well known algorithms applied in the last decade for the purpose of prediction of software maintainability such as GRNN (General Regression Neural Network) model, ANN (Artificial Neural Network) Model, Bayesian Model, RT (Regression Tree) Model, Backward Elimination Model, Stepwise Selection Model, MARS (Multiple Adaptive Regression Splines) Model, Tree Nets Model, GN (Generalized Regression) Model, ANFIS (Adaptive Neuro Fuzzy inference System) Model, SVM (Support Vector Machine) model and MLR (Multiple Linear Regressions) Model etc in terms of MRE (Magnitude of Relative Error), MMRE (Mean Magnitude of Relative Error), Pred(0.25) and Pred(0.75).

The rest of the paper is organized as follows: Section 2 highlights the objectives of the study; Section 3 provides overview of the related research work conducted on prediction of software maintainability. Section 4 describes the machine learning algorithms proposed in this study i.e. GMDH, GA and PNN along with their advantages. Section 5 summarizes well thought-out selection of software design metrics and source of data considered in this empirical study. Section 6 includes the experimental setup, Results of the study, Analysis of results and Discussion. In Section 7, threats to validity have been discussed and finally Section 8 concludes the paper.

2. STUDY OBJECTIVE

The biggest irony of the software industry is that the largest cost associated with any software product over its lifetime is actually its maintenance cost. Most suggested approaches by all researchers for controlling maintenance costs is to utilize software metrics during the development phase. Studies examining the link between OO software metrics and maintainability have found that in general these metrics can be used as predictors of maintenance effort [6, 8, 9, 11, 12, 13, 14, 16]. The result shows in almost all the studies that the prediction accuracy of one model is more accurate on one dataset but is less accurate for another dataset. Although a number of maintainability prediction models have been developed in last two decades, they have low prediction accuracies according to the criteria suggested by Conte *et al*. [20]. Therefore, it is necessary to explore new techniques, which are not only easy in use but also provide high prediction accuracy for the purpose of maintainability prediction.

The GMDH algorithm [21, 22] is ideal for complex, unstructured system where the investigator is only interested in obtaining a high order input-output relationship [23]. Also, the GMDH algorithm is heuristic in nature and not based on solid foundation as is regression analysis. For many end users it may be more convenient to have such a model, which is able to make predictions using familiar polynomial formulas which are widely understood. GMDH is formulated as neural network architecture, and is called a polynomial network however; the output of the model is in the form of standard polynomial function. In fact, the GMDH network is not like regular feed forward networks and was not originally represented as a neural network. The GMDH algorithm and its modified versions have been previously applied to wide array of problems to ascertain predictions [23]. In the year 2009, it was also used for the prediction for software reliability [24] where it was proved as one of the best available models. In the current study an attempt was made to apply this model perhaps for the first time for the task of software maintainability prediction using Object Oriented software design metrics. The background of proposing GMDH model in software maintainability was that if it had proven empirically to predict reliability with least errors compared with other techniques [24], than possibly it may be useful as a sound alternative to existing techniques for maintainability predictions. Moreover, it was presumed that better cohesion and benefit to industry would be gained, if the same model can effectively predict both reliability and maintainability of newly developed software since they work in tandem to achieve the overall goal of software quality. The objective of our study was to apply GMDH model along with two other models GA and PNN; all three are machine learning algorithms and compare them with prevailing prediction models proposed in last decade to ascertain their performance in software maintainability.

3. RELATED WORK

There are several models and metrics proposed in literature to predict the maintainability of the softwares. These methods vary from simple statistical models such as regression analysis to complex machine learning algorithm such as neural networks etc. Various methods proposed in the literature for the prediction of maintainability have been summarized in Table 1. We elaborate few important studies here. Multiple Linear Regression (MLR) Model was used by Li and Henry to predict maintenance effort in 1998 [25] in which they not only created MLR model for prediction but also successfully earmarked those metrics which have strong impact on prediction of object oriented software maintainability. In the year 2000 Muthanna *et al*. also used polynomial regression to establish the relationship between design level metrics [26] and the corresponding maintainability of Industrial software. The results have shown that predicted values using polynomial regression were quite close to actual values. Dagpinar *et al*. also based their study on empirical data to establish the relationship between software metrics and its maintainability however instead of design level metrics of structure languages, the metrics were replaced by object oriented metrics. They recorded significant impact of two metrics i.e. direct coupling metric and size metric on software maintainability while other parameters like cohesion, inheritance and indirect coupling were not considered significant by them [18]. Fioravanti and Nesi in 2001 [5] presented a metric analysis to identify which metrics would be better ranked for its impact on prediction of adaptive maintenance for object-oriented systems.

The model and metrics proposed have been validated against real data by using MLR (Multilinear Regression Analysis) Model. The validation has identified that several metrics can be profitably employed for the prediction of software maintainability. Misra used linear regression in 2005 [7] and presented an empirical study which was based on intuitive and experimental analyses. It used a suite of twenty design and code measures to obtain their indications on software maintainability. Thwin and Quah [8] used neural networks to build object oriented software maintainability prediction models. Koten and Gray [11] used Bayesian Belief Network (BBN) to predict object-oriented software maintainability in year 2006. Many researchers [6, 8, 9, 11, 12, 13, 14, 16] have used two datasets (UIMS and QUES) proposed by Li and Henry [2] for model building and its evaluation. Zhou and Leung [12] used Multivariate Adaptive Regression Splines (MARS) for predicting object-oriented software maintainability in year 2007. They compared the performance of the MARS model with other very popular four models Multivariate linear regression (MLR), Support Vector Regression (SVR), Artificial Neural Network (ANN) and Regression Tree (RT). The results provided by them [12] suggest that MARS can predict maintainability more accurately and precisely with less error when compared with other models for the QUES dataset, and as accurate as the best model for the UIMS dataset. In the last decade some machine learning algorithms have also been proposed and evaluated. It has been verified empirically that the machine learning algorithms can predict maintainability more accurately and precisely. Aggarwal *et al*. suggested the use of Fuzzy model; Kaur *et al*. [14] stated the use of soft computing approaches such as ANN, FIS and ANFIS. Elish *et al*. [13] used Tree Nets and proved that they also provide competitive results when compared with other models. Recently Cong and Liu [19] have used Support Vector Machine. They conducted their study where the code was written to implement detection for the "temper proof HTML web page" that can be either used standalone or embedded as component within other layers of applications. The code was written in C++. Hidden Markov Model (HMM) was used by Ping [15] to define health index of a product in literature and suggested that it works as a weight on the process of maintenance behaviour over a period of time. The detail classification is given in Table 1.

TABLE 1
METHODS CLASSIFICATION

S.No.	Author	Year	Methods Used
1.	G. M. Berns.	1984	Maintainability Analysis Tool (Kind of Lexical Analyzer)
2.	D Kafura and R Reddy	1987	Static Analyzers (To count)
3.	Wake, S. and S. Henry	1988	Multiple Linear Regression Model
4.	Li W., Henry S.	1993	A Classic Ada Metric Analyzer [Based on LEX and YACC of UNIX
5.	RD Banker, S M Datar, CF Kemerer and Dani Zweig	1993	Statistical Model which assign weight to each metric
6.	F Niessink,HV Vliet	1997	Regression models
7.	D Stavironoudis, M Xenos, D Christodolakis,	1999	Experts Judgements
8.	F Fioravanti, PNesi	2001	Multiple linear Regression Model
9.	K.K. Aggarwal, Y Singh, JK Chhabra	2002	Fuzzy Model
10.	M Dagpinar, JH. Jhanke	2003	Multivariate Analysis, Correlation, Best subset Regression Model
11.	LA Stamelos, DE Sakellaris	2003	Bayesian belief networks

12.	SC Misra	2005	Linear Regression, co relation and multiple regression
13.	MMT Thwin , Quah,	2005	General Regression Neural Network (GRNN)
14.	K.K. Aggarwal, YSingh, P Chandra and M Puri	2005	Fuzzy Model
15.	K.K. Aggarwal, YSingh, A Kaur, and R Malhotra	2006	Artificial Neural Network
16.	C.V Koten, A.R. Gray	2006	Of Bayesian network, Neural network
17.	Y Zhou, H Leung	2007	Multiple adaptive regression splines (MARS)
18.	N.N Prasanth, S.Ganesh, G. Dalton	2008	Fuzzy repertory table (FRT) and Regression analysis
19.	N.N Prasanth, S.P.Raja, X.Birla, K.Navaz, SAA Rahuman	2009	Used Static Analysers and set threshold values of these metrics
20.	WANG Li-jin, HU Xin-xin, NING Zheng-yuan KE Wen-hua	2009	Projection Pursuit Regression (nonparametric multivariate regression technique)
21.	MO. Elish and KO Elish	2009	TreeNets
22.	A Kaur, K Kaur, R Malhotra	2010	Artificial Neural Network, Fuzzy Inference System (FIS), Adaptive Neuro Fuzzy Inference System (ANFIS)
23.	L Ping	2010	Hidden Markov Model (HMM) is used to simulate the maintenance behaviors shown
24.	C Jin, JA Liu	2010	Support Vector Machine
25.	F Marzoughi, MM Farhangian, A Marzoughi, ATH Sim	2010	Bayesian network theory

4. PROPOSED MODELS

This section is further divided into three sub-sections in which all the three proposed models are discussed in detail:

1. Group Method of Data Handling (GMDH)
2. Genetic Algorithm (GA)
3. Probabilistic Neural Network (PNN) with Gaussian Activation Function

4.1 Group Method of Data Handling (GMDH)

Russian Scientist A.G. Ivakhnenko introduced a technique in 1966 [21, 22], for constructing an extremely high order regression type model termed as GMDH. The algorithm, GMDH builds a multinomial of degree in hundreds, whereas standard multiple regression Boggs down in computation and linear dependence. The GMDH model has been described in Section 2 however few inherent advantages with GMDH approach are briefly highlighted as under:

(i). GMDH can predict the outcome even with smaller training sets.
(ii). The computational burden is reduced with GMDH model.
(iii). The procedure automatically filters out input properties that provide little information about location and shape of hyper surface.
(iv). A multilayer structure maintained in GMDH model is a computationally feasible way to implement multinomial of high degree.

The GMDH model has a forward multi-layer neural network structure. Each layer consists of one or more units

wherein two inputs arcs and one output arc is attached with every unit. Each unit corresponds to Ivakhnenko polynomial form [21, 22].

$$Z = a + bx + cy + dx^2 + exy + fy^2 \quad (1)$$

Or

$$Z = a + bx + cy + dxy \quad (2)$$

Where variables x and y are input variables and Z is output variable and a, b, c …f are the parameters. Fujimoto et al. [27] described that the basic technique of GMDH learning algorithm is a self-organization method and it fundamentally consists of the following steps:

(i). Given a learning data sample including a dependent variable y and independent variables x1, x2, ... , xm ; split the sample into a training set and a checking set.
(ii). Feed the input data of m input variables and generate combination (m, 2) units from every two variable pairs at the first layer.
(iii). Estimate the weights of all parameters 'a' to 'f' in formula as per either using equation (a) or equation (b). Applying it on training dataset in the next step. In this study, stepwise regression method with formula given in equation (b) was employed.
(iv). Compute mean square error between actual and predicted value of each unit.
(v). Sort out the unit by mean square error in decreasing order and eliminate bad units.
(vi). Set the prediction of units in the first Layer to new input variables for the next layer, and build up a multi-layer structure by applying Steps (ii) (v).
(vii). When the mean square error become larger than that of the previous layer, stop adding layers and choose the minimum mean square error unit in the highest layer as the final model output. Steps (iv) and (v) describe an important and basic technique of GMDH algorithm. It is called regularity criteria and leads to achieving the minimum error at Step (vii) [22].

Some GMDH networks can be large. When it has around 7 or more input variables, we have to separate it into several sub-networks with 6 or less input variables since the rule extraction process becomes too complex with many input variables. Using GMDH network structure developed from the dataset of nominal dependent variable y and independent variables x the following model is obtained [27]:

Output Layer:

$$\hat{x}_i(k) = a(i) + b(i)[x_i(k-1)] = a(i) + b(i)[a(i) + b(i)[x_i(k-1)]] \quad (3)$$

Hidden Layers

$$x_i(k) = a_i(k) + b_i(k)x_{n-1}(k-1) + c_i(k)x_n(k-1) + d_i(k)x_{n-1}(k-1) \quad (4)$$

Input Layer

$$x_i(k) = a_i(k) + b_i(k)x_n(0) + c_i(k)x_n(0) + d_i(k)x_n(0)x_n(0) \quad (5)$$

Where G is the maximum layer, $X_i(k)$ is the output, $a_i(k)$ ……. $d_i(k)$ are the parameters of unit i in layer k and $x_n(0)$ is input variable n. After rule extraction for each sub-network of GMDH, several sub-rules are obtained. The action part of the sub-rule for the lower sub-network is corresponds to the condition part of the sub-rule for the upper sub-network. Then, by integrating all of the sub-rules, we can obtain a rule set for the whole GMDH network as shown in Figure 1.

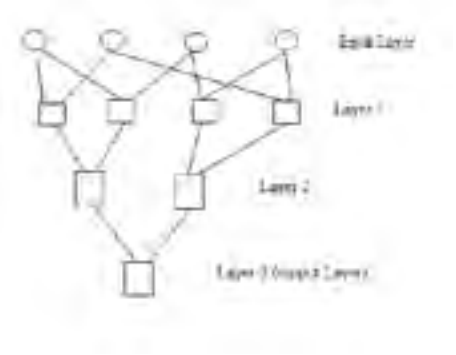


FIGURE 1 : STRUCTURE OF MULTI-LAYER NETWORK

4.2 Genetic Algorithms (GA)

A Genetic Algorithm is an adaptive system motivated by biological system proposed in Charles Darwin's evolution theory. It is a high level simulation. The GA starts with a set of solutions (represented by chromosomes) called population. GA is a search heuristic and it mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions for optimization and search problems. Best solutions from one population are then taken and used to form a new population which will be better than the old one. While choosing the solutions, their fitness function is evaluated. Those solutions which are more close to fitness function have more probability to be selected. We say that the more suitable solutions have more chances "to survive". This process is repeated until some condition is satisfied such as achievement of best solution. Hence the population is improved over generations to accomplish the best solution. Indeed, GA is the methods designed to optimize the solutions

of prediction problem by simulating the "evolution behavior". The following processes are repeatedly applied until an optimized solution to the given problem is found:

(i). Natural selection
(ii). Crossover
(iii). Mutation

When this process is repeated over time, the better-fit individuals are the ones who survived; hence the genetic algorithms are also called as function optimizers. While implementing GA we first create a population with or without fixed size; First time usually, this population is randomly generated. Each individual of this population is then tested against "fit function". Reproductive opportunities are given to those individuals who have a better solution to the target problem and they have better chances of survival. Those individual solutions of the populations which are poorer and produce "weaker" solutions, they have less chances of survival. The "goodness" of a solution is defined in terms of the problem which needs to be solved. While solving any issue using GA the researchers first break the given issue into two problems i.e. the encoding problem and the evaluation problem. While designing the evaluation function, utmost care has to be kept in mind. The entire algorithm would fail if the evaluation function is poorly designed. It should be capable of measuring correctly the solution to the given problem. Evaluation function need not to be a mathematical expression and it could be a complete simulation.

The following are general steps implemented when using GA algorithms:

(i). Generate a random initial population.
(ii). Create the new population by applying the selection and reproduction operators to select pairs of strings. The number of pairs will be the population size divided by two, so the population size will remain constant between generations.
(iii). Apply the crossover operator to the pairs of the strings of the new population.
(iv). Apply the mutation operator to each string in the new population.
(v). Replace the old population with the newly created population.
(vi). Copy the best-fitted individual(s) to the newly created population to warrantee evolution.
(vii). (If the number of iterations is less than the maximum go to step two, else stop) OR (If the fitness of the best result does not get better over certain number of iteration, then stop).

4.3 *Probabilistic Neural Networks (PNN)*

This network has been originated from Neural Networks [16, 28, 37]. In the neural networks, it looks for patterns in training sets of data, learn these patterns, and develop the ability to correctly classify new patterns or to make forecasts and predictions. Neural networks excel at problem diagnosis, decision making, prediction, and other classifying problems where pattern recognition is important and precise computational answers are not required. Neural network starts its job by first recognizing patterns and trains the network. Training continues until the network reaches the conditions set in the 'Training' and 'Stop Training Criteria' module. This module calls different learning sub programs depending upon the paradigm and architecture we select. PNN is a feed forward neural network created by Specht [16] around 1990. It is based on Bayesian network and Kernel Fisher discriminate analysis. In a PNN, the operations are organized into a multilayered feed forward network with four layers:

(i). Input layer
(ii). Hidden layer
(iii). Pattern layer/Summation layer
(iv). Output layer

First layer is input layer where one neuron is present for each independent variable. The next layer is the hidden layer. This layer contains one neuron for each set of training data. It not only stores the values of the each predictor variables but also stores each neuron along with its target value. Next is the Pattern layer. In PNN networks one pattern neuron is present for each category of the output variable. Last layer is output layer. At this layer weighted votes for each target category is compared and selected. PNN are known for their ability to train quickly on sparse datasets as it separates data into a specified number of output categories. The network produces activations in the output layer corresponding to the probability density function estimate for that category. The highest output represents the most probable category. In the proposed study the number of neurons in the input layer at Slab 1 is equal to the number of inputs in our problem i.e. we have selected 11 independent variables summarized in Table 2 as inputs. The number of neurons in the output layer i.e. Slab 4 corresponds to the number of outputs. In the proposed study "Change" is taken as output variable. The number of neurons in the hidden layer defaults to the number of patterns in the training set because the hidden layer consists of one neuron for each pattern in the training set. We inspected smoothing factor for each link and apply it to all links. The smoothing factor that is defined during the design stage is default but we changed it in the training

sessions in order to make predictions more accurate and precise. We have experimented with different smoothing factors to discover which works best for our problem. We have applied the trained network to our training set, and to a test set too, using different smoothing factors and selected which was giving us the best answers. Then we used this module to train PNN networks. Unlike back propagation networks, which require feedback of errors and subsequent adjustment of weights and many presentations of training patterns, training a PNN network is very fast because it requires that each pattern be presented to the network only once during training. During the training session we can see the number of learning events completed during training which is also called as "epoch". Training can be done in real time since training is almost instantaneous. When data is sparse, training is superior to other network types. The success of PNN networks is dependent upon the smoothing factor. The adaptive PNN network is very powerful as during the building of neural networks, it uses genetic algorithms. Initially, we developed a GA based network algorithm that uses the GA directly with Calibration to improve the network's generalization. There are three ways for calibration of PNN which are as under:-

(i). Iterative Calibration Proceeds in Two Parts. The first part trains the network with the data in the training set. The second part uses Calibration to test a whole range of smoothing factors, trying to hone in on one that works best for the network created in the first part.
(ii). Genetic Adaptive: Uses a genetic algorithm to find appropriate individual smoothing factors for each input as well as an overall smoothing factor. The input smoothing factor is an adjustment used to modify the overall smoothing factor to provide a new value for each input.
(iii). None: In this calibration technique simply trains the network and we do not find an overall smoothing factor. The value for the smoothing factor is default chosen and applied. The user will have to manually adjust the smoothing factor by entering a new one in the edit box while using this module.

Even though PNN are slower and require more memory space, there are several advantages of PNN such as they are much faster, more accurate, and relatively insensitive to outliers, use Bayes optimal classification approach and generate accurate predicted target probability.

5. METRICS AND DATA

First we define the goal of this empirical study which is as follows:

Model : Evaluate GMDH model, Genetic model and PNN model (with Gaussians activation function) for the purpose of predicting object oriented software maintainability with respect to its prediction accuracy against the prevailing models like GRNN model, ANN Model, Bayesian Model, MARS Model, TreeNets, SVM model, Generalized Regression Model and ANFIS Model proposed by various researchers and practitioners during previous decade.

Metrics: We have worked on the set of metrics initially proposed by Chidamber et al. [29], and later by Li and Henry [2, 25, 30] and revised in Aggarwal et al. [36] as given in Table -2.

TABLE 2
METRICS DEFINITION

Metrics	Definition
WMC (Weighted Methods per Class)	The sum of McCabe's cyclomatic complexities of all local methods in a class. Let a class K1 with method M1…… Mn that are defined in the class. Let C1…….Cn be the complexity of the methods. W $WMC = \sum_{i=1}^{n} Ci$
DIT (Depth of Inheritance Tree)	The depth of a class in the inheritance tree where the root class is zero.
NOC (Number of Children)	The number of child classes for a class. It counts number of immediate sub classes of a class in a hierarchy.
RFC (Response For a Class)	The number of local methods plus the number of non local methods called by local methods.

LCOM (Lack of Cohesion of Methods)	The number of disjoint sets of local methods. Each method in a disjoint set shares at least one instance variable with at least one member of the same set.
MPC (Message Passing Coupling)	The number of messages sent out from a class.
DAC (Data Abstraction Coupling)	The number of instances of another class declared within a class.
NOM (Number of Methods)	The number of methods in a class.
SIZE1 (Lines of code)	The number of lines of code excluding comments.
SIZE2 (Number of properties)	The total count of the number of data attributes and the number of local methods in a class.
CHANGE (Number of lines changed)	The number of lines added and deleted in a class, change of the content is counted as two.

Datasets: In our study we use two most popular object-oriented maintainability datasets which are also published by Li and Henry [2]: UIMS and QUES datasets. Their Descriptive statistics is given in Table -3 and Table - 4 followed by the interpretation. These datasets were chosen mainly because they have been recently used by many researchers to evaluate the performance of their proposed model in predicting object-oriented software maintainability [6, 8, 9, 11, 12, 13, 16, 17, 19] and hence we wanted to be able to compare our results against this published work. The UIMS dataset contains class-level metrics data collected from 39 classes of a user interface management system, whereas the QUES dataset contains the same metrics collected from 71 classes of a quality evaluation system. Both systems were implemented in Ada. Both datasets consist of eleven class-level metrics: ten independent variables and one dependent variable. The independent variables are taken as follows:

(i) Five variables are taken from Chidambar et al. [30] : WMC, DIT, NOC, RFC, and LCOM;
(ii) Four variables are taken from Li and Henry [2, 25]: MPC, DAC, NOM, and SIZE2;
(iii) One variable is taken from traditional lines of code metric (SIZE1).
(iv) The dependent variable is a maintenance effort surrogate measure (CHANGE), which is the number of lines in the code that were changed per class during a 3-year maintenance period. A line change could be an addition or a deletion. A change in the content of a line is counted as a deletion and an addition. Table 2 defines each metric in the datasets.

TABLE 3
DESCRIPTIVE STATISTICS OF UIMS DATASET

Metric	Minimum	Maximum	Mean	Standard Deviation
WMC	0	69	11.38	15.90
DIT	0	4	2.15	0.90
NOC	0	8	0.95	2.01
RFC	2	101	23.21	20.19
LCOM	1	31	7.49	6.11
MPC	1	12	4.33	3.41
DAC	0	21	2.41	4.00
NOM	1	40	11.38	10.21
SIZE1	4	439	106.44	114.65
SIZE2	1	61	13.47	13.47
CHANGE	2	253	42.46	61.18

TABLE 4
DESCRIPTIVE STATISTICS OF QUES DATASET

Metric	Minimum	Maximum	Mean	Standard Deviation
WMC	1	83	14.96	17.06
DIT	0	4	1.92	0.53
NOC	0	0	0.00	0.00
RFC	17	156	54.44	32.62
LCOM	3	33	9.18	7.31
MPC	2	42	17.75	8.33
DAC	0	25	3.44	3.91
NOM	4	57	13.41	12.00
SIZE1	115	1009	275.58	171.60
SIZE2	4	82	18.03	15.21
CHANGE	6	217	64.23	43.13

From the descriptive statistics we noticed some observations and accordingly actions were taken. Some of them are mentioned as follows:

(i) For DIT Median and Mean value are minimum in both the system, so we draw conclusion that the use of inheritance in both systems is limited.

(ii) The values for median and mean for CHANGE (dependent variable) in the UIMS dataset is lesser than those in the QUES, which means UIMS seems to be more maintainable.

(iii) We had removed NOC from the QUES dataset because it was observed that all data points for NOC are zeros in the QUES dataset.

(iv) We had observed that the coupling between classes in QUES was higher than those in the UIMS because the medians and means values for RFC and MPC in the QUES dataset were larger than UIMS dataset.

(v) Values of Mean and Median of LCOM were almost same in both systems that mean both have almost similar cohesion.

(vi) The similar medians and means for NOM and SIZE2 in both datasets suggest that both systems had similar class sizes at the design level. However, there was a significant difference in SIZE1.

6. DISCUSSION OF RESULTS

This section consists of four subsections. In section 6.1 we have discussed experiment setup, values of various parameters initialized and values of various important parameters received after training the machine for prediction using GMDH algorithm on given pattern dataset and processing. Section 6.2 discusses the various prediction accuracy measures to compare the results of our studies with other proposed models available in literature. In section 6.3 we have summarized the values of the parameters selected in section 6.2 followed by their analysis. Section 6.4 does the interpretation of the results.

6.1 Experiment Setup and Results

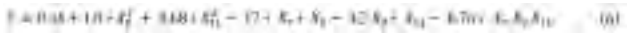
In this section, the results of GMDH model were analyzed using UIMS and QUES datasets. We employed GMDH algorithm available in Neuroshell2 tool [31, 32, 33] to predict the maintainability of software. We set the parameters as shown in Table 5 while applying the proposed models on the dataset as discussed in section 5 using the tool Neuroshell2.

TABLE 5
VALUES OF PARAMETERS BEFORE EXPERIMENTAL SETUP

S.No.	Parameter	Value
1	Scale Function	[0-1]
2	GMDH type	Advanced
3	Optimization	Full
4	Maximum Variable	X1, X2, X3
5	Selection Criteria	Regularly
6	Missing value to be	Error Condition

We received following values of various parameters after we finished the process of machine learning using GMDH technique for the given data and followed by comparing the actual values with that of predicted values.

Best Formula



Where X_1, X_2,……., X_{11} are the parameters estimated by GMDH in terms of OO metrics selected in section-5 and described in Table 2 and their values are given as under:-

X_1= 2*Class-1
X_2= 2* DIT/4 -1
X_3= 2*NOC -1
X_4= 2* (MPC-2)/40 -1
X_5= 2* (RFC-17)/139 -1
X_6= 2* (LCOM -3)/30-1
X_7=2*DAC/25-1
X_8= 2* (WMC-1)/82-1
X_9= 2* (NOM-4)/53-1
X_{10}= 2* (size2-4)/78-1
X_{11}= 2* (size1-115)/894-1
Y = min (max (Change-6)/211……),1..)

The values of various parameters and their description are given in Table 6.

TABLE 6
VALUES OF PARAMETERS CALCULATED WHEN GMDH MODEL IS APPLIED ON DATA SET

S.No	Parameter	Value	Description
1	MSE (Mean Squared Error)	0.003	It is a statistical measure of the differences between the values of outputs in the training set and the output values the network is predicting. This is mean over all patterns in file of the square of the actual value minus the predicted value, i.e., the mean of (actual - predicted)^2. The errors are squared to penalize the larger errors and to cancel the effect of the positive and negative values of the differences.
2	R-Squared	0.913	It compares accuracy of the model to the accuracy of a trivial benchmark model wherein the prediction is just the mean of all of the samples. A perfect fit would result in an R squared value of 1, a very good fit near 1, and a very poor fit less than 0.
3	Correlation Coefficient (Pearson's Linear Correlation Coefficient)	0.955	This is a statistical measure of the strength of the relationship between the actual versus predicted outputs. The r coefficient can range from -1 to +1. The closer r is to 1, the stronger the positive linear relationship, and the closer r is to -1, the stronger the negative linear relationship. When r is near 0, there is no linear relationship.
4	Normalized Mean Square Error	0.032	The Normalized Mean Square Error or Root Mean Square Error (RMSE) is a frequently used measure of the differences between values predicted by a model or an estimator and the values actually observed. It is a good measure of accuracy. The individual differences are called residuals, and it serves to aggregate differences into a single measure of predictive power.

6.2 Prediction Accuracy Measures

An important question that needs to be asked of any prediction model is "How accurate are its predictions". Based on the two values namely actual value and predicted values, researchers have stated various methods to evaluate the quality of predictions [3, 10, 20, 34, 35]. In our proposed study we evaluated and compared the OO software maintainability prediction models quantitatively with other proposed models. We used following measures:

(i). MRE (Magnitude of Relative Error): It is a normalized measure of the discrepancy between actual values and predicted values as proposed by Kitchenham in 1991 [34]. Ever since it is proposed by the author, it has become the de facto standard to measure the accuracy of software maintainability prediction. It is given as :

$$MRE = \frac{|\,Actual\ Value - Predicted\ Value\,|}{Actual\ Value} \quad (7)$$

(ii). MMRE (Mean Magnitude of Relative Error) : It is the mean of MRE and calculated as follows :

$$MMRE = \sum_{i=1}^{N} MRE_i \quad (8)$$

MMRE measures the average relative discrepancy. It is equivalent to the average error relative to the actual effort in the prediction. In our study we have expressed MMRE as actual values however in some studies it is expressed in %. MMRE has been regarded as a versatile assessment criterion and has number of advantages such as it can be used to make comparisons across datasets and all kinds of prediction model types and it is independent of measuring unit and scale independent [20].

(iii). Pred : It is the measure of what proportion of the predicted values have MRE less than or equal to specified value, given by Fentom. [35]

$$Pred(q) = \frac{K}{N} \quad (9)$$

Where q is the specified value
K is number of cases whose MRE is less than or equal to q
N is total number of cases in the datasets

In current study we have used most commonly values such as pred(0.25) and pred(0.30) in the field of software effort prediction literature so that we can compare our results.

(iv). R-Square - It is a measure of the quality of fit. It is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is perfect correlation between targets and outputs [28]. It is calculated by square of the correlation coefficient. 100% R-square means perfect predictability.

(v). P-values – p-values are used for testing the hypothesis of no correlation. Each p-value is the probability of getting a correlation as large as the observed value by random chance, when the true correlation is zero. If p is small, say less than 0.05, and then the correlation i.e. R is significant [31].

6.3 Comparison with other studies

We also have compared the values of prediction accuracy measures of certain selected parameters with the studies conducted in the last decade. In Table 4 we presented the summarized performance measures of all Models studied on UIMS and QUES data set in the last decade.

TABLE 4
COMPARISON OF VARIOUS MODELS WITH REFERENCE TO
THEIR PREDICTIVE PERFORMANCE FOR QUES AND UIMS SYSTEM

S No	Model Name	Reference	Max MRE	MMRE	MARE	Pred (0.25)	Pred (0.30)	Pred (0.75)	Remarks
1.	GMDH (Group Method of Data Handling) Model	Proposed in Current Study	0.983	0.210	-	0.69	0.722	0.944	Lowest MMRE Recorded in Current Study
2.	Genetic Model		0.794	0.220	-	0.66	0.722	0.972	
3.	PNN (Probabilistic Neural Networks)		0.923	0.230	-	0.68	0.75	0.944	

4.	GRNN (General Regression Neural Network) Model	[8]	4.295	0.765	-	-	-	-	
5.	ANN (Artificial Neural Network) Model	[9]	-	0.265	-	-	-	-	R- Squared 0.582 p-value 0.004
6.	Bayesian Model	[11]	1.592	0.452	-	0.391	0.430	-	
7.	Regression Tree	[11]	2.104	0.493	-	0.352	0.383	-	
8.	Backward Elimination	[11]	1.418	0.403	-	0.396	0.461	-	
9.	Stepwise Selection	[11]	1.471	0.392	-	0.422	0.500	-	
10.	MARS (Multiple adaptive regression splines)	[12]	1.91	0.32	-	0.48	0.59	-	
11.	MLR (Multiple Linear Regression)	[12]	2.03	0.42	-	0.37	0.41	-	
12.	SVM (Support Vector Machine)	[12]	2.07	0.43	-	0.34	0.46	-	
13.	ANN (Artificial Neural Network) Model	[12]	3.07	0.59	-	0.37	0.45	-	
14.	Regression Tree	[12]	4.82	0.58	-	0.41	0.45	-	
15.	TreeNets	[13]	-	0.42	-	0.58	0.65	-	
16.	Generalized Regression	[14]	-	-	0.308	-	-	-	
17.	ANFIS(Adaptive Neuro Fuzzy Inference System) Model	[14]	-	-	0.242	-	-	-	

For analyzing the results of Table 4 we have taken Max MRE (Magnitude of Relative Error) Values of other models as well as proposed models and presented them in Figure 2. It can be observed easily that MRE for GMDH is lowest which implies that it can be used as sound alternative for the prediction of maintainability. In Figure 3 a comparison has been shown between MMRE values of proposed models and other prevalent models

is presented. From Table 4 and Figure 3 it can observed that out of the thirteen algorithms selected and evaluated, the GMDH model, Genetic model and PNN model gives very competitive results and hence show their worth that they can be used in the process of software maintainability prediction. To Analyze the prediction accuracy measure we have taken values at PRED(0.25) and PRED(0.30) of all models and presented them in Figure 4. It is evident from the Table 4 and Figure 4 that prediction accuracy of GMDH network model is much better than other models. It is the only model which is close to the criterion laid by Conte *et al*. [20] and Mac Donell [3].

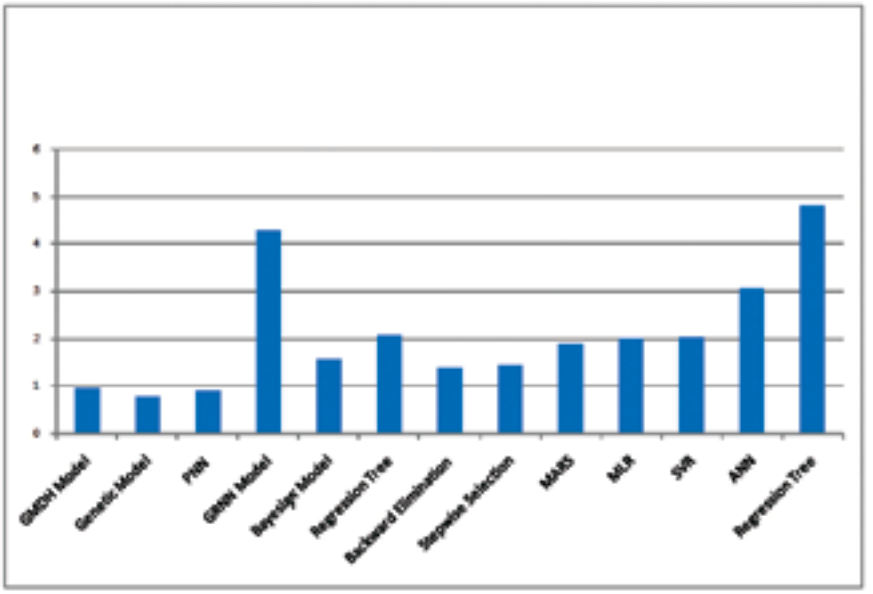


FIGURE 2: MAX MRE OF PROPOSED MODELS VERSUS OTHER MODELS

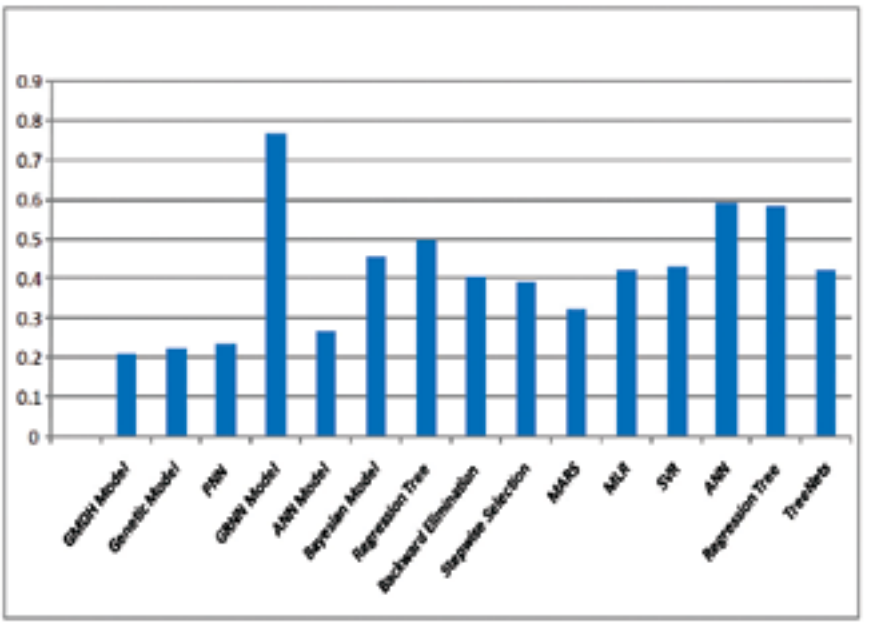


FIGURE 3: MMRE OF PROPOSED MODELS VERSUS OTHER MODELS

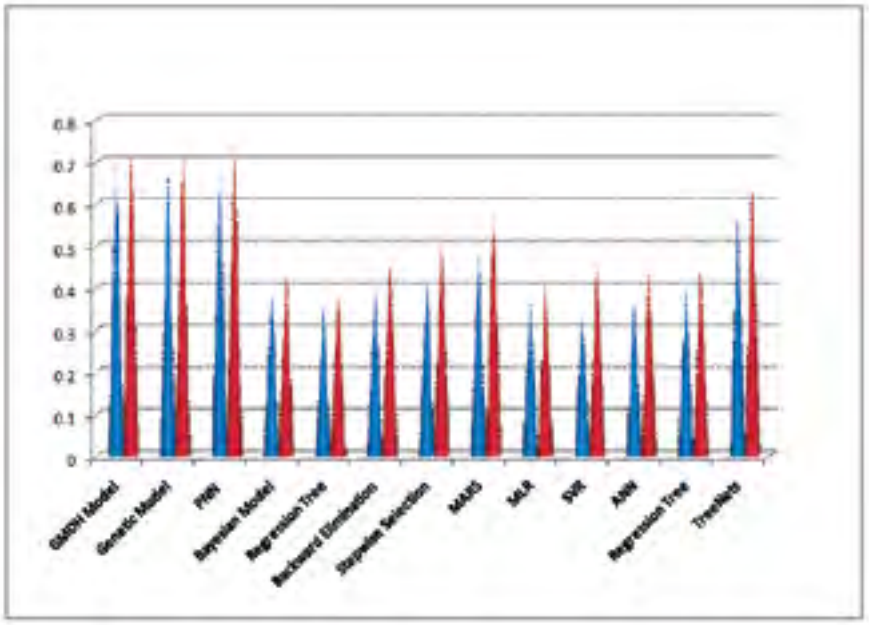


FIGURE 4: PRED(0.25) AND PRED(0.30) OF PROPOSED MODELS VS. OTHER MODELS

6.4 Analysis and Interpretation

In this paper, we have presented an empirical study that sought to build object-oriented software maintainability prediction model using following three machine learning algorithms:

(i) Group Method of Data Handling(GMDH)
(ii) Genetic Algorithm (GA)
(iii) Probabilistic Neural Network (PNN) Using Gaussian Activation function

The GMDH and GA is proposed for the first time for prediction of the software maintainability. Although Artificial Neural Network has been used previously in literature [8, 14, 37] but for the first time the Probabilistic Neural Network (PNN) along with Gaussian activation function has been applied. In this study, to draw most realistic comparison we have also analyzed the same dataset which was originally proposed by Li and Henry and earlier applied by various researchers to predict maintainability as per the details summarized in Table 4.

The criteria for prediction given by Conte *et al*. [20] and MacDonell [3] states that prediction model is considered accurate if value of pred(0.25) and pred(0.30) is greater than pred(0.75) which clearly proposed models in this study satisfies. In the literature it is also suggested that prediction accuracy of software maintenance effort prediction models is often low and thus it is very difficult to satisfy the criteria [10, 13]. It can be noticed from Table 4 that none of the prediction models satisfy the criteria. However, the GMDH model has achieved improved Pred(0.25) and Pred (0.30) over the other models in QUES and UIMS datasets, and its results are quite closer to the criteria set in literature[3, 20].

It is evident from the Table 4, the prediction accuracy of GMDH network model is much better than all the other models. At pred(0.25) its values are 0.69 which means that almost 69% predictions are less than the error of 0.25 prediction accuracy. At pred(0.30) its value is 0.722 which means that almost 72% predictions are less than the error of 0.30 prediction accuracy as compared to other models as shown in Figure 4. Following comparative analysis, it is safe to conclude that GMDH has clearly outperformed than other models. The GMDH models can predict maintainability of the OO software systems with least MMRE when compared with others models such as GRNN, ANN, Bayesians, MARS, TreeNets and SVM for QUES dataset. Hence it is clear inference that GMDH is the most accurate and best model for the predictions of software maintainability.

The SVM (Support Vector Machine) model was proposed recently by Cong *et al.* [19] for predicting maintainability using OO metrics, however it is not comparable to the current study because of the fact that their study was merely conducted on the code which was written for "Temper proof HTML web page" in C++ whereas our

study is conducted on commercially available QUES dataset written in ADA with much higher scope. Not only the sizes of the software differ to large degree but also both systems varied in to great extent with respect to their paradigm and complexity. Secondly, Max MRE and Pred(q) were not provided despite being de facto prediction standards. MARE in their model recorded as 0.218. When it is compared with current study, MMRE has been recorded better at 0.210 with GMDH model that clearly confirms higher competence even in complex environment.

7. THREATS TO VALIDITY

Like other empirical studies, limitations confronted during the current study are given as under:

(i) UIMS and QUES datasets which are considered in the study undertaken are written in ADA language. The models which have been derived in this study are likely to be valid for the code written in other object-oriented programming languages, for example C++ or Java, however, further research can only establish their usefulness in predicting the maintainability of other development paradigms.

(ii) During the process of selecting independent variables while constructing the proposed model, although utmost care has been taken and only those eleven variables are being chosen which we consider to have the strong impact on maintainability, nevertheless few other prevailing independent variables and their effect on maintainability of software also needs to be determined. Some of the widely used variables which warrant further consideration for fair comparability are Class Method Complexity (CMC), Number of Ancestor Classes (NAC), and Number of Descendent Classes (NDC), coupling through Abstract Data Type (CTA), Class Complexity (CC), Exception Handling Factor EHF, Number of Object Memory Allocation (NOMA), Average Number of Live Variables, and Average Live Variable Span etc.

(iii) Similarly, in our study although three machine learning algorithms GMDH, GA and PNN have been applied, however few others which have also gained popularity in recent times due to their effectiveness, like Random Forest, Decision Tree and Naïve Bays Network are also required to be studied on the way to empirically ascertain their merit over the proposed models.

(iv) Measuring the effectiveness of machine learning algorithms for predictions of the procedural languages is also a limitation of the proposed models.

8. CONCLUSION

Three different machine learning algorithms are used for the purpose of prediction of software maintainability. Even though many studies reported wide application of GMDH model and GA model in diverse fields for the purpose of prediction of high order input output relationship which is complex, non linear and unstructured but for the first time they are used for prediction of software maintainability. The goal of our study is to construct suitable model using machine learning algorithm for the prediction of object oriented software maintainability which are not only easy to apply but could reduce the prediction errors to minimum. The prediction performance of the machine learning algorithms based models like GMDH, GA and PNN were assessed and compared with prevailing models in terms of MMRE, Pred(0.25) and Pred(0.30). It was found that GMDH model outperformed the prevailing models as the least MMRE value is recorded. As far as Pred(0.25) and Pred(0.30) values are concerned, all the three proposed models are significantly better over others. Thus, it is concluded that GMDH network model is indeed a very useful modeling technique and it could be used as a sound alternative for the prediction of software maintainability.

As the proposed model was found suitable for estimating the software reliability in an earlier work [24], therefore with current findings it can be safely presumed that both the reliability and maintainability, which remain indispensable components of software quality, can possibly be predicted with application of same model i.e. GMDH. This will certainly reduce the challenges involved with prediction of maintainability and assists software developers to strategically utilize their resources, enhance process efficiency and optimize the associated maintenance costs.

As the current study was based on two commercial software datasets UIMS and QUES developed in ADA, therefore the authors are actively involved in carrying out further work that applies the proposed GMDH network model to more objective datasets to ascertain its authenticity for wider software paradigms. Such studies would allow us to investigate the capability of GMDH network and finally establishing a generalized model in the field of Software Quality. Further research is planned in an attempt to combine GMDH model with other data mining techniques so as to develop prediction models which can estimate the maintainability of software more accurately with least precision errors.

REFERENCES

[1]. IEEE Standard. 1219-1993 IEEE Standard for Software Maintenance. INSPEC Accession Number: 4493167 DOI: 10.1109/IEEESTD.1993.11557 Journal .1993

[2]. W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," *Journal of Systems and Software*, vol. 23, no 2, pp. 111-122, 1993.

[3]. S.G. Mac Donell, "Establishing relationships between specification size and software process effort in case environment*," Information and Software Technology*, vol. 39, no 1, pp. 35-45, 1997.

[4]. L Briand, C Bunse, J Daly, "A controlled experiment for evaluating quality guidelines on the maintainability of object oriented design", *IEEE Transaction on software Engineering,* vol:27, no: 6, pp 513-530, 2001, DOI: 10.1109/32.926174.

[5]. F. Fioravanti and P. Nesi, "Estimation and prediction metrics for adaptive maintenance effort of object oriented systems", *IEEE Transactions on Software Engineering*, vol. 27, no. 12, pp. 1062-1084, 2001.

[6]. KK Aggarwal, Y Singh, JK Chhabra, "An Integrated Measure of Software Maintainability", Annual *proceedings: Reliability and Maintainability Ssymposium*, pp 235-241, 2002.

[7]. S. Misra, "Modeling design/coding factors that drive maintainability of software systems" *, Software Quality Journal*, vol. 13, no. 3, pp. 297-320, 2005.

[8]. M. Thwin and T. Quah, "Application of neural networks for software quality prediction using object oriented metrics", *Journal of Systems and Software*, vol. 76, no. 2, pp. 147-156, 2005.

[9]. K.K. Aggarwal, Y. Singh, P. Chandra and M. Puri, " Measurement of Software Maintainability Using a Fuzzy Model", *Journal of Computer Sciences*, vol. 1, no.4, pp. 538-542, 2005 ISSN 1549-3636 © 2005 Science Publications.

[10]. A.D. Lucia, E. Pompella, and S. Stefanucci, "Assessing effort estimation models for corrective maintenance through empirical studies" , *Information and Software Technology*, vol. 47, no. 1, pp. 3-5, 2005.

[11]. C.V. Koten, A.R. Gray, "An application of Bayesian network for predicting object-oriented software maintainability", *Information and Software Technology Journal* , vol: 48, no : 1, pp 59-67, Jan 2006.

[12]. Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines" , *Journal of Systems and Software,* vol. 80, no. 8, pp. 1349-1361, 2007.

[13]. MO. Elish and KO. Elish , "Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study", *European Conference on Software Maintenance and Reengineering,* pp 1534-5351, March 2009, DOI 10.1109/CSMR.2009.57.

[14]. A Kaur, K Kaur, R Malhotra, "Soft Computing Approaches for Prediction of Software Maintenance Effort*", International Journal of Computer Applications*), vol 1, no. 16, *pp :* 0975 – 8887, 2010.

[15]. L Ping, "A Quantitative Approach to Software Maintainability Prediction", *International Forum on Information Technology and Applications, vol: 1, no : 1, pp : 105-108, July 2010.*

[16]. D.F. Specht , "Probabilistic Neural Networks", *Journal of Neural Networks, Elsevier, v*ol. 3, no 1, pp 109-118, 1990, DOI : .org/10.1016/0893-6080(90)90049-Q.

[17]. T.D. Morco, "Controlling Software Projects: Management, Measurement and Estimation," ISBN 10: 0131717111 / 0-13-171711-1; ISBN 13: 9780131717114; Yourdon ; 1982.

[18]. M. Dagpinar and J.H. Jahnke , "Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison", *Proceedings of the 10th Working Conference on Reverse Engineering, pp 155-164, Nov* 2003.

[19]. C Jin , A.L.Jin , "Applications of Support Vector Machine and Unsupervised Learning for Predicting Maintainability using Object-Oriented Metrics", *Second International Conference on Multi Media and Information Technology ,vol 1 ,no : 1, pp 24-27, April 2010.*

[20]. S. Conte, H. Dunsmore, and V. Shen," Software Engineering Metrics and Models". Book, *Menlo Park, CA: Publisher: Benjamin-Cummings publishing co., ISBN:0-8053-2162-4*, 1986.

[21]. A. G. Ivakhnenko, "Group Method of Data Handling- A Rival of the method of Stochastic Approximation," *Soviet Automatic Control,*, vol 13, no. 3, 43-71, 1966.

[22]. A. G. Ivakhnenko and Y. U. Koppa, "Regularization of decision functions in the group method of data handling", *Soviet Automatic Control*, vol 15 no. 2, 28-37, 1970.

[23]. S.J. Farlow, "The American Statistician", vol 35, no. 4 , Nov 1981, 210-215, {The American Statistician is currently published by American Statistical Association}.

[24]. R. Mohanty, V. Ravi, M.R. Patra, "Software Reliability Prediction using Group Method of Data Handling", Proceeding of 12th International conference on *RSFDGrC, Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, LNAI 5908, pp. 344-351, 2009.

[25]. W. Li, "Another Metric Suite for Object-oriented Programming", *The Journal of System and Software*, vol 44, no : 2, pp 155–162, December 1998.

[26]. S. Muthanna, K. Kontogiannis, K. Ponnambalam, B. Stacey, " A Maintainability Model for Industrial Software Systems Using Design Level Metrics", Proceeding of Seventh Conference on Reverse Engineering, IEEE Computer Society, pp. 248, 2000.

[27]. K. Fujimoto and S. Nakabayashi, "Applying GMDH algorithm to extract rules from examples", *Systems Analysis Modeling Simulation,* vol. 43, no. 10, October 2003, pp. 1311-1319.

[28]. M.M. Ibrahiem, E.l. Emary and S. Ramakrishnan, "On the Application of Various Probabilistic Neural Networks in Solving Different Pattern Classification Problems", *World Applied Sciences Journal*, vol. 4, no. 6, pp. 772-780, 2008, ISSN 1818-4952.

[29]. S. Chidamber, R. Shyam and C. Kamerer, "Towards a metrics Suite for Object-Oriented Design Proceedings", *Proceeding of Conference on object – oriented programming systems, languages and applications, OOPSLA'91*, pp.197-211, November, 1991.

[30]. S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. pp. 476-493, June 1994.

[31]. http://www.mathworks.com.

[32]. http://www.inf.kiev.ua/GMDH-home/GMDH_abo.htm.

[33]. http://www.wardsystems.com.

[34]. B.A. Kitchenham, L.M. Pickard, S.G. MacDonell, M.J. Shepperd," What accuracy statistics really measure", *IEEE Proceedings-Software* vol. 148, no. 3, pp 81–85, 2001.

[35]. N.E. Fentom, S.L. Pfleeger, "Software Metrics: A Rigorous and Practical Approach, Second Edition", *PSW publishing Company*, 1997.

[36]. K. K. Aggarwal, Y. Singh,A. Kaur and R. Malhotra, "Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics", *World Academy of Science*, pp. 140-144, 2006

[37]. KK Aggarwal, Y Singh, A Kaur, R Malhotra, "Application of artificial neural network for predicting fault proneness models" , *International Conference on Information Systems, Technology and Management (ICISTM 2007*), pp.12-13, March 2007.

[38]. KK Aggarwal, Y Singh, A Kaur, R Malhotra, "Analysis of object-oriented metrics", International Workshop on Software Measurement *(IWSM)*, 2005.

ABOUT THE AUTHORS

Ruchika Malhotra is an Assistant professor with Department of Software Engineering, Delhi Technological University (formerly Delhi College of Engineering), India. She was an Assistant professor with University School of Information Technology, Guru Gobind Singh Indraprastha University, India. She is the Executive Editor of Software Engineering International Journal, Delhi Technological University. She received her doctorate and masters degree from the University School of Information Technology, Guru Gobind Singh Indraprastha University, India. She is a co-author of book titled "Object Oriented Software Engineering" published by PHI Learning. Her research interests are in improving software quality, statistical and adaptive prediction models for software metrics, neural nets modeling, and the definition and validation of software metrics. She has 60 publications in international journals and conferences. Her paper titled Analysis of object oriented Metrics was published as a chapter in the book Innovations in Software Measurement (Shaker -Verlag, Aachen 2005). She can be contacted by e-mail at ruchikamalhotra2004@yahoo.com.

Anuradha Chug is Assistant Professor in the Department of University School of Information and Communication Technology (USICT), Guru Gobind Singh Indraprastha University, New Delhi. She has long teaching experience of almost 19 years to her credit as faculty and in administration at various educational institutions in India. She has worked as guest faculty in Netaji Subhash Institute of Information and Technology, Dwarka, New Delhi and Regular Faculty at Government Engineering College, Bikaner. She has also worked as Academic Head, Aptech, Meerut and Program Coordinator at Regional Centre, IGNOU, Meerut. Her areas of research interest are Software Engineering, Neural Networks, Analysis of Algorithms, Data Structures and Computer Networks. She is currently pursuing her Doctorate degree from Delhi Technological University. She has earlier achieved top rank in her M.Tech (IT) degree and conferred the University Gold Medal in 2006 from Guru Gobind Singh Indraprastha University. Previously she has acquired her Master's degree in Computer Science from Banasthali Vidyapith, Rajasthan in the year 1993. She has also presented number of research papers in national/ international seminars, conferences and research journals. She can be contacted by E-mail at a_chug@yahoo.co.in