Received: March 1, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Jaya Algorithm and Artificial Neural Network Based Approach for Object-Oriented Software Quality Analysis

Mukesh Bansal[1]* **Chaitanya Purushottam Agrawal[2]**

[1]*Makhanlal Chaturvedi National University of Journalism and Communication, Bhopal, India*
[2]*Department of Computer Science and Applications,*
Makhanlal Chaturvedi National University of Journalism and Communication, Bhopal, India
* Corresponding author's Email: mukeshbansal76@gmail.com

Abstract: This paper develops a technique by using Jaya algorithm and feed-forward neural network to determine the quality of object-oriented software by using Chidamber & Kemerer (CK) along with Li & Henry metrics. The technique basically focuses on the maintainability factor of software quality which in turn depends upon the software complexity. The software complexity is directly proportional to the number of changes done per class which is determined by the technique. The analysis has been done on UIMS (User Interface Management System) and QUES (Quality Evaluation System) datasets by using the mean absolute error as the analysis parameter. The reduction in the mean absolute error as compared to the existing state of art techniques along with the individual component of proposed technique proves the significance of the technique.

Keywords: Jaya algorithm, Neural network, Maintainability, Software, Complexity, UIMS, QUES.

1. Introduction

The object-oriented methodology has covered the software development market due to several advantages over the conventional approaches. The development of the software engineering techniques makes it necessary to control the quality and cost of software. The quality attributes for software includes maintainability, reliability, usability, portability, functionality and the efficiency [1, 2]. The maintainability is the major concerns discussed in this paper. The complexity is one of the major factors to control the maintainability of software, as more complex software is difficult to edit and test resulting low software quality [3, 4]. Moreover, the complex software takes a long time to enhance and maintain the software resulting increased cost and effort, so software complexity needs to be controlled to increase the software quality [4].

Various metrics have been given by different authors to compute the complexity of object-oriented methodology based software. Different metrics are of Chidamber & Kemerer (CK) [5], Li & Henry metrics [6], and metrics for object-oriented design (MOOD) [7] to measure the complexity of software [8]. This paper focuses on the CK and Li & Henry metrics to determine software quality. The metrics are used to compute the number of lines changed per class in the software which in turn determines the maintainability of the software [9]. This work takes some changes directly proportional to the complexity of software, i.e., higher software complexity will lead to a large number of changes per class and vice-versa.

Different authors have worked to determine the complexity or maintainability of the software. The author of [10] uses linear regression to determine the maintainability of the software. The author of [11] uses the artificial neural network using CK and Li & Henry metrics to determine the software maintainability. The author of [12] uses the feed forward back propagation neural network and support vector machine to predict the maintainability. The techniques are given by the authors of [10 - 12] work on the original given set of attributes which includes relevant as well as noisy attributes resulting low performance. While the author of [9], uses various combinations of optimization technique along with neural network to determine the software maintainability. The technique is FLANN-CSA, i.e., functional line artificial neural network with clone selection algorithm. The author also uses the principal component analysis (PCA) and rough set theory (RST) to reduce the features that define the CSA (PCA) and CSA (RST) techniques. The PCA and RST used for feature reduction to reduces the noisy feature doesn't explore the search space exhaustively which is improved in the proposed technique by using Jaya algorithm. Further, the paper has been divided into three sections. Section two gives the Jaya algorithm which is used for feature reduction followed by proposed methodology in section three. Then the result along with the implementation and discussion has been given in section four.

2. Jaya algorithm

Jaya algorithm is a simple but powerful optimization algorithm [13]. Jaya algorithm is one of the parameter fewer algorithms, i.e., the performance of the algorithm do not depend upon the parameter setting (constants) unlike the various existing algorithms. The algorithm basically determines the best and the worst solution among the candidate solutions. Then the algorithm updates each solution by moving towards the best solution and moving away from the worst solution. The updation phenomenon is given by Eq. (1):

$$\begin{aligned} S'_{p,q,r} &= S_{p,q,r} + rand_{p,q,r} \mathrm{X}(S_{p,best,r} - S_{p,q,r}) \\ &- rand_{p,q,r} \mathrm{X}(S_{p,worst,r} - S_{p,q,r}) \end{aligned} \quad (1)$$

Here, $S'_{p,q,r}, S_{p,q,r}$ showsthe updated and the current solution for the rth iteration of qth candidate of pth variable. The best and worst solution for the same candidate is given by $S_{p,best,r}$, $S_{p,worst,r}$, $rand_{p,q,r}$ is the random value between 0 and 1. The updated value i.e. $S'_{p,q,r}$is accepted only if it is better than the current solution. This process is repeated to take the solution towards success. The process stops when the stopping criteria are achieved [13]. This algorithm can be applied to optimize any process. The next section describes how Jaya algorithm is used in this work. Moreover, the proposed methodology defines a method to analyze object-oriented software quality.

3. Proposed methodology

This section describes a technique to determine the number of changes per class by using CK, Li &Henry metrics for object-oriented software's. The technique uses the Jaya algorithm for the feature reduction and the neural network for the prediction. The complete steps of the technique can be easily described in Fig. 1.

The Fig. 1, i.e. flowchart briefs the steps used in the proposed methodology. The input dataset is normalized along with the prediction factor to avoid the noise due to the scalability of data. The scaling has been done by using Eq. (2):

$$NV = \frac{(V - Max\_V)}{(Max\_V - Min\_V)} \quad (2)$$

Here, NV shows the normalized value while the V is the current value with Max_V, Min_V are the maximum and minimum value for present attributes.

Then the normalized data has been reduced by using the Jaya algorithm. In this, Initially, the random contribution is assumed for each attribute towards the prediction. Then the fitness is evaluated by using Eq. (3) which calculates the mean absolute error for this contribution.

$$F = \frac{\sum_{i=1}^{N} \left(Actual\_changes_i - Estimated\_changes_i \right)}{N} \quad (3)$$

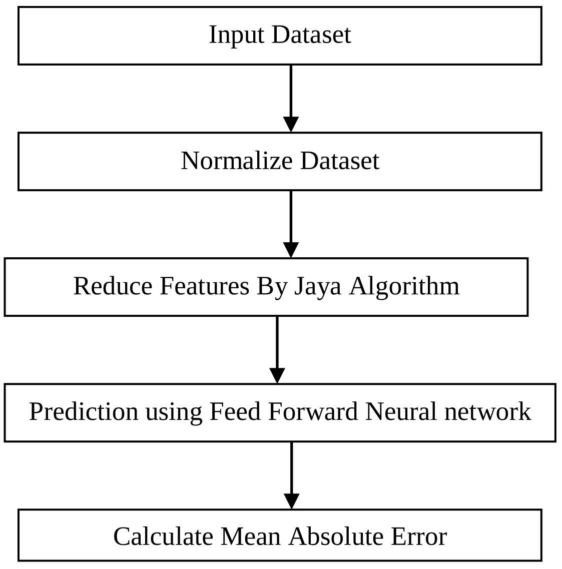
Input Dataset
Normalize Dataset
Reduce Features By Jaya Algorithm
Prediction using Feed Forward Neural network
Calculate Mean Absolute Error

Figure. 1 Proposed flowchart

Where actual changes are given in the dataset itself and estimated changes for any particular instance is given by Eq. (4):

$$Estimated\_changes = \sum_{e=Each\_Attribute} Contribution.Attributes_e \quad (4)$$

This calculates the fitness of each instance of the dataset by using each attribute as e and its contribution as “Contribution”. Then instance corresponding minimum and maximum fitness value are selected as the best and workable solution. Then each solution is updated Eq. (1) already described in the previous section. This process is repeated for the given number of iteration. Here, the maximum iterations are 10 to calculate the best solution. This best solution determines the contribution of each attribute. The new dataset is prepared by using best value contribution as given by Eq. (5):

$$New\_dataset = \Lambda_{e=each\_attribute} Contribution_e.attribute_e \quad (5)$$

The new dataset has been prepared by using the contribution of each attribute identified by using Jaya algorithm given as “Contribution”. This new dataset is classified by using the feed-forward neural network. Five-Fold cross-validation has been used to determine the better mean absolute error in the changes. The following algorithm can understand the complete process:

```
1. Input Dataset
2. For each attribute say V
       $NV = \frac{(V - Max\_V)}{(Max\_V - Min\_V)}$
       Cont =rand;
   End
3. Iteration=1
4. For iteration<max_iteration
       a. $Est\_chn = \sum_{e\varepsilon V} Cont.V_e$
       b. $F = \frac{\sum_{i=1}^{N}(Act\_chn_i - Est\_chn_i)}{N}$
       c. best_cont=cont(Min(F))
       d. worst_cont=cont(Max(F))
       e. $New\_Cont = Cont + rand.(best\_cont - cont) - rand.(worst\_cont - cont)$
       f. $NEst\_chn = \sum_{e\varepsilon V} New\_Cont.V_e$
       g. $NF = \frac{\sum_{i=1}^{N}(Act\_chn_i - NEst\_chn_i)}{N}$
       h. If NF<F
              Cont =NCont
          End if
       i. iteration=iteration+1
   end
5. $New\_dataset = \Lambda_{e\varepsilon V} Cont_e.V_e$
6. [Train Test]=Use 5-Fold Cross-Validation
7. For each fold
       MAE=Use Fed forward Neural Network
   End
8. Return mean(MAE)
```

The above algorithm computes the mean absolute error, which determines the software quality and the process has been implemented using the MATLAB discussed in the next section.

4. Result and discussion

The model discussed in the previous section has been analyzed by using metrics values (datasets) given by [6]. The author [6] gives two datasets, i.e., metric value for two software systems known as UIMS and QUES. This software is developed in classic-ADa programming language, an object-oriented language. The two software systems have 39 and 71 instances respectively. The dataset given by [6] comprises of two CK metric suite and the Li & Henry metric suite along with size and the change attribute. The size attribute consists of two types of sizes one is the line of code and other is a sum of the number of attributes and local methods. The change attribute specifies the mean of the changes done in the time span of 3 years. Fig. 2 shows the mean, median and standard deviation of all ten attributes described for the UIMS datasets.

Fig. 2 has two subplots; the first subplot denotes the analysis of mean, median and standard deviation of all attributes while the second subplot represents the analysis of same for a normalized dataset.

The normalization has been done using Eq. (2). A similar analysis for QUES dataset has been shown in Fig. 3. Fig. 3 denotes the analysis of QUES dataset for the given and normalized values by mean, median and standard deviation. The analysis has been done by using five-fold cross-validation.

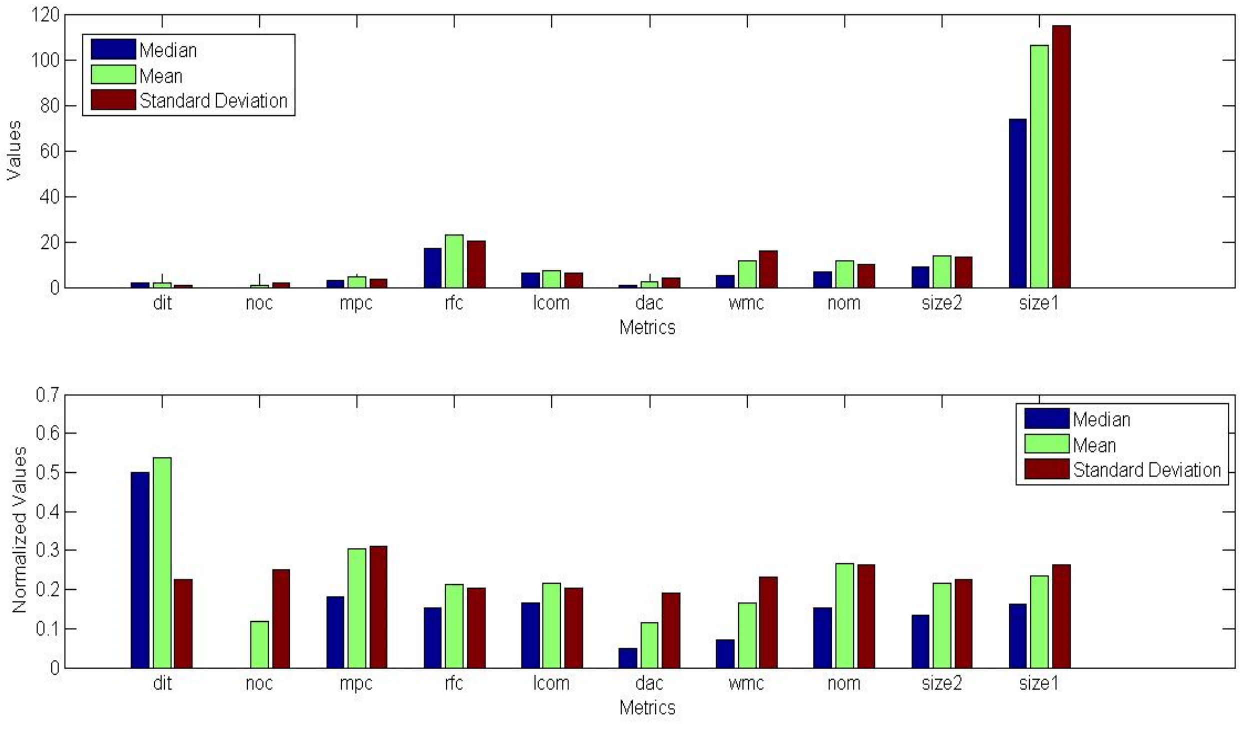

Figure. 2 UIMS dataset analysis

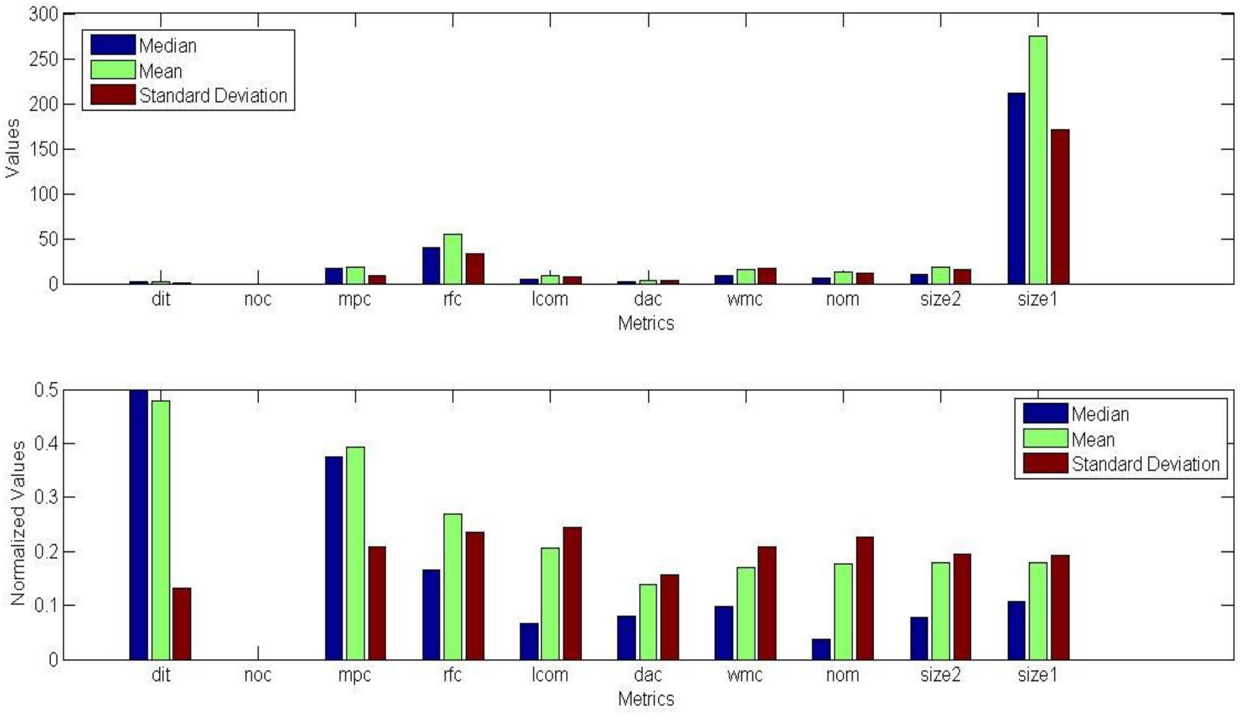

Figure. 3 Analysis of QUES dataset

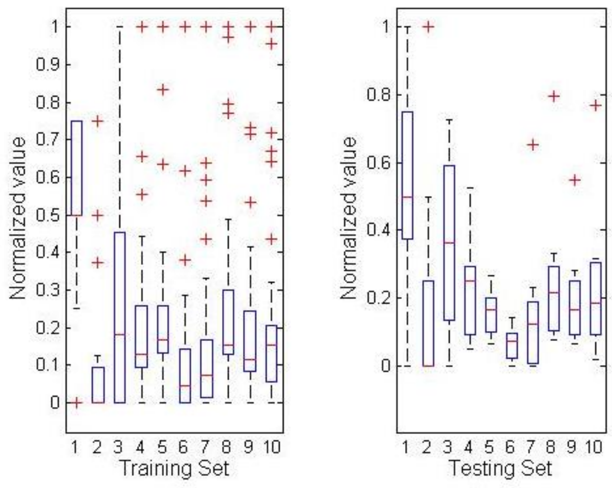

Figure. 4 Box Plot for UIMS dataset

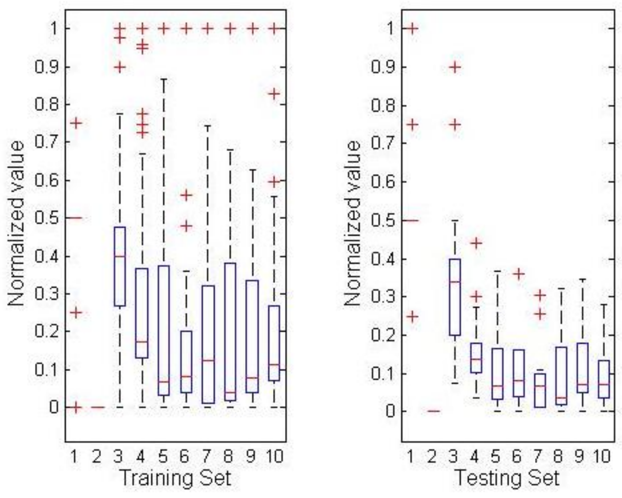

Figure. 5 Box plot for QUES dataset

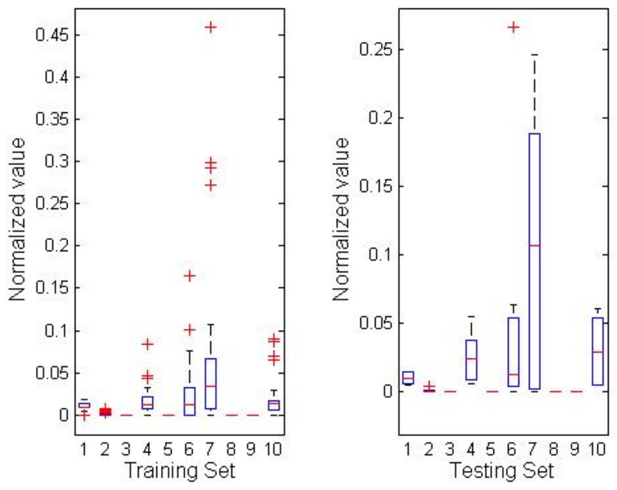

Figure. 6 Box Plot for UIMS reduced data

The box plot for training and testing data of both datasets are shown in Figs. 4 and 5 to analyze the distribution of the data. The datasets have been divided into testing and training part randomly by using the five-fold cross-validation. These figures show only the data for one fold for each dataset respectively.

The data has been reduced by using the Jaya algorithm, the box plot for corresponding reduced data for one fold is given in Figs. 6 and 7. The Figs. 6 and 7 denote that the Jaya algorithm has concussed the variation in the data.

The neural network has been applied to this modified data to analyze the performance. Moreover, the complete datasets have been scaled as shown in Figs. 6 and 7, i.e., changes in the testing as well as training data. The evaluation of objective function for the Jaya algorithm to reduce the data as discussed earlier is shown in Figs. 8 and 9. The Jaya algorithm has been iterated to reduce the data, the cost of the attributes has been calculated in each iteration for each dataset. The Figs. 8 and 9 show the cost for datasets UIMS and QUES respectively for Jaya algorithm.

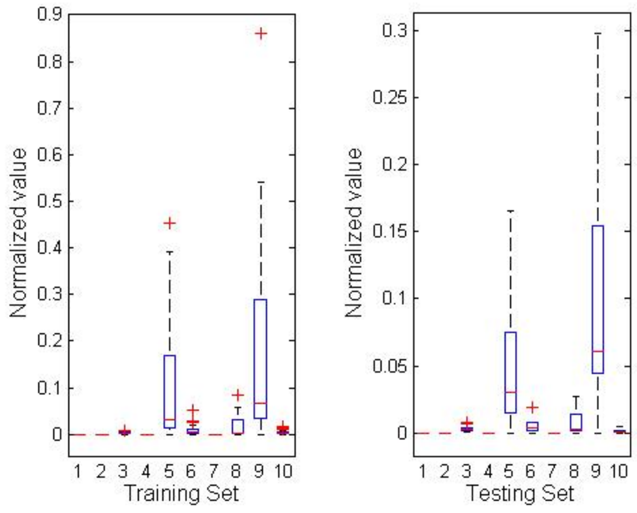

Figure. 7 Box plot for QUES reduced data

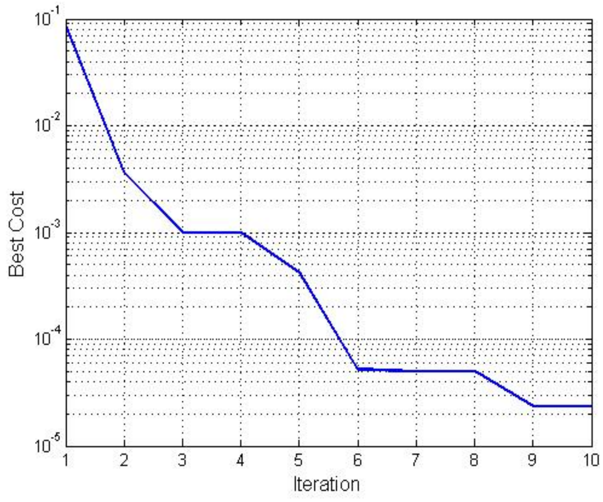

Figure. 8 Cost by objective function using UIMS dataset

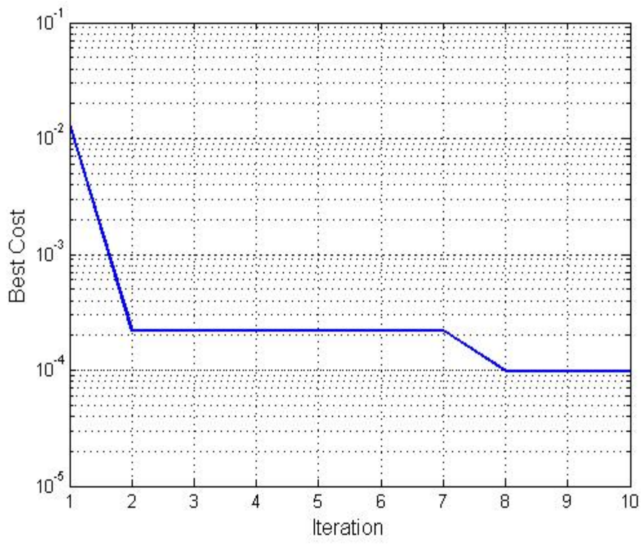

Figure. 9 Cost by objective function using QUES dataset

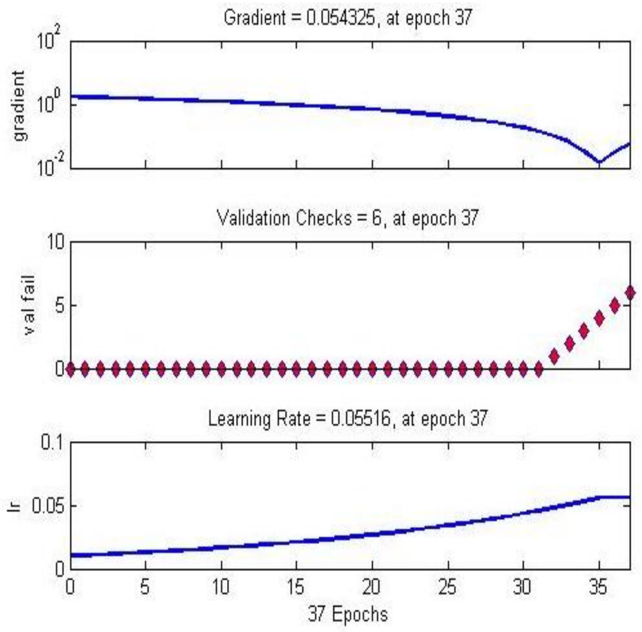

Figure. 10 Neural network training state for UIMS reduced data

The neural network is applied to the reduced data to make the prediction. The training pattern displaying complete details of training for UIMS and QUES datasets is given in Figs. 10 and 11.

The Figs. 10 and 11 demonstrate that the 37 and 29 epoch is used to train the reduced data of UIMS and QUES dataset respectively. Moreover, the learning rate along with its gradient has been shown in Figs. 10 and 11, respectively.

The training state for each dataset by a neural network is given in Figs. 12 and 13 for the original data, i.e. without Jaya algorithm.

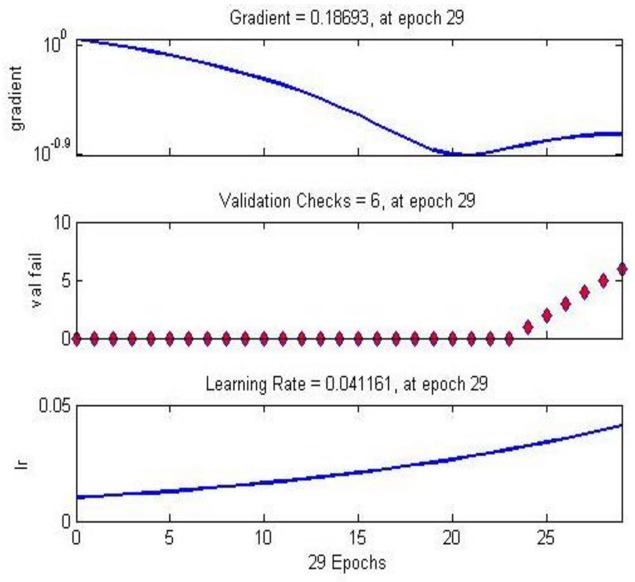

Figure. 11 Neural network training state for QUES reduced data

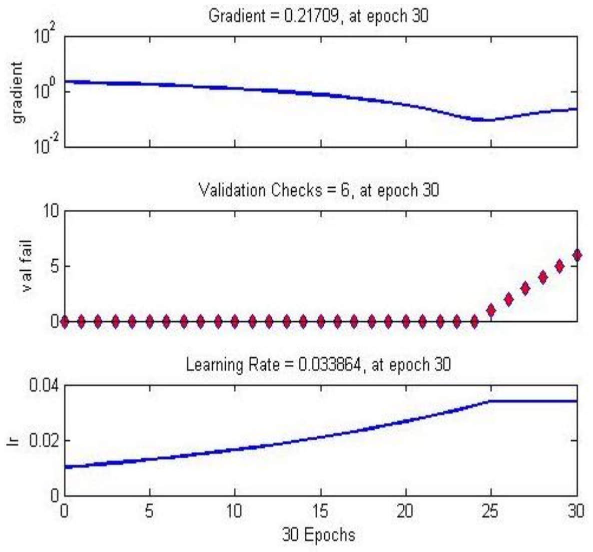

Figure. 12 Neural network training state for UIMS dataset

The change in the number of epochs, learning rate as well as in the gradient can be easily recognized for UIMS and QUES datasets in Figs. 12 and 13 as compared to the corresponding values for reduced datasets in Figs. 10 and 11.

The mean absolute error has been calculated by using the Eq. (6) for each dataset.

$$MAE = \frac{\sum_{i=1}^{N}\left(Actual\_ch_i - Estimated\_ch_i\right)}{N} \tag{6}$$

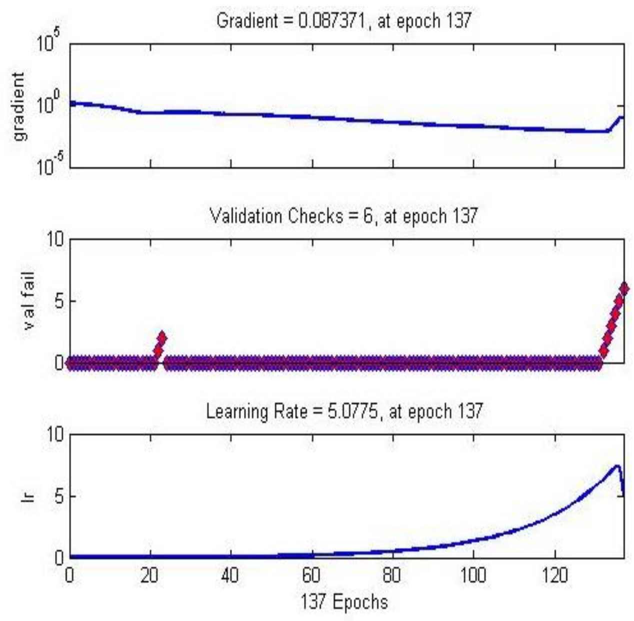

Figure. 13 Neural network training state for QUES dataset

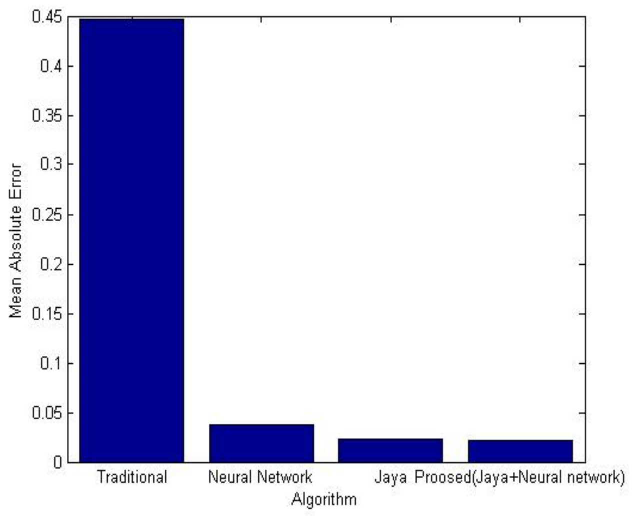

Figure. 14 MAE comparison of UIMS dataset

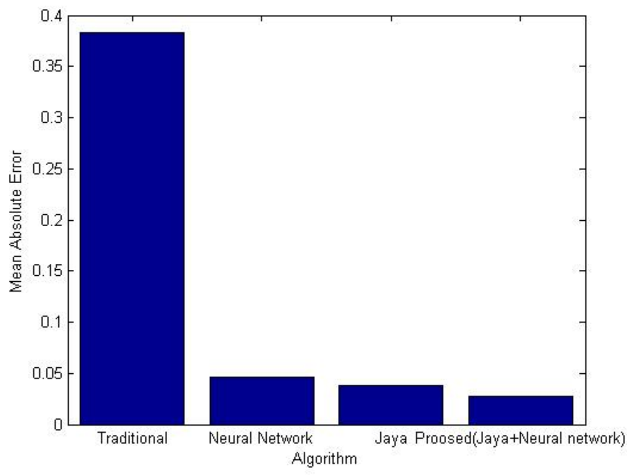

Figure. 15 MAE comparison of QUES dataset

Table 1. MAE comparison of proposed technique with existing state of the art technique

Techniques	MAE in UIMS dataset	MAE in QUES dataset
Traditional[4]	0.4464	0.3833
CSA[9]	0.0937	0.1587
CSA(PCA)[9]	0.0998	0.1181
CSA(RST)[9]	0.0537	0.1374
Jaya Algorithm [13]	0.0235	0.0381
Proposed	0.0211	0.0272

The mean absolute error is calculated by comparing the actual and the estimated changes. Here N is the number of instances in the corresponding dataset.

The analysis of the MAE is shown in Figs. 14 and 15 respectively for datasets UIMS and QUES respectively. The performance analysis has been done on four variants, i.e., traditional, i.e. statistical method to compute changes directly, compute the changes using a neural network, compute the changes by using Jaya algorithm and the proposed (changes by using Jaya and Neural Network).

The comparison is shown in Figs. 14 and 15 clearly demonstrate that error has been reduced by using the Jaya with neural network technique as compared to other individual techniques. The reduce error shows the significance of the algorithm. The improvement in the performance is due to the optimized exploration and exploitation search property of the Jaya algorithm. Jaya algorithm enables the proposed technique to determine the relevant feature which is further classified by the neural network. The neural network gives the better classification results due to its learning capability. This comparison can be made with the different techniques available in the literature with their corresponding results given in Table 1.

Table 1 compares the existing state of art techniques, i.e., CSA, CSA (PCA) and CSA (RST) already described in previous sections with the proposed algorithm. The reduction in the absolute error in the proposed technique signifies its performance here. This is due to the enhanced performance of the Jaya algorithm as compared to the rough set theory. This removes the noisy features resulting improved performance as compared to existing techniques.

5. Conclusion

This paper designs a Jaya and neural network based technique to analyze the complexity of software which in turn determines the maintainability cost of the software. The technique has been analyzed by computing the mean absolute error over two datasets, i.e., UIMS and QUES for the number of changes in software over a span of 3 years time. The designed technique has been compared with the existing state of art technique as well as with the traditional (statistical) approach to show its significance. The proposed technique shows the 7.3% reduced error as compared to existing CSA (RST) technique which already shows more than 90% improvement on traditional technique. Moreover, the designed technique has been compared with its components, i.e., Jaya algorithm and the neural network to prove its significance. In future new metrics can be incorporated to improve the performance. The model designed in the paper can verify the new metrics.

References

[1] L. Kumar, A. Krishna, and S. K. Rath, "The impact of feature selection on maintainability prediction of service-oriented applications", *Serv. Oriented Comput. Appl.*, Vol. 11, No. 2, pp. 137–161, 2017.

[2] H. W. Jung, S. G. Kim, and C. S. Chung, "Measuring software product quality: A survey of ISO/IEC 9126", *IEEE Softw.*, Vol. 21, No. 5, pp. 88–92, 2004.

[3] K. Kevrekidis, S. Albers, P. J. M. Sonnemans, and G. M. Stollman, "Software complexity and testing effectiveness: An empirical study", In: *Proc. of the Annual Reliability and Maintainability Symposium*, pp. 539–543, 2009.

[4] R. D. Banker, S. M. Datar, and D. Zweig, "Software complexity and maintainability", In: *Proc. of the tenth international conference on Information Systems - ICIS '89*, pp. 247–255, 1989.

[5] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design", *IEEE Trans. Softw. Eng.*, Vol. 20, No. 6, pp. 476–493, 1994.

[6] W. Li and S. Henry, "Object-oriented metrics that predict maintainability", *J. Syst. Softw.*, Vol. 23, No. 2, pp. 111–122, 1993.

[7] F. B. Abreu and R. Carapuça, "Object-Oriented Software Engineering : Measuring and Controlling the Development Process", In: *Proc. of the 4th. Int. Conf. Softw. Qual.*, pp. 3–5, 1994.

[8] M. Bansal and C. P. Agrawal, "Critical Analysis of Object Oriented Metrics in Software Development", In: *Proc. of the Fourth International Conference on Advanced Computing & Communication Technologies*, pp. 197–201, 2014.

[9] L. Kumar and S. K. Rath, "Hybrid functional link artificial neural network approach for predicting maintainability of object-oriented software", *J. Syst. Softw.*, Vol. 121, pp. 170–190, 2016.

[10] S. C. Misra, "Modeling design/Coding factors that drive maintainability of software systems", *Softw. Qual. J.*, Vol. 13, No. 3, pp. 297–320, 2005.

[11] K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics", In: *Proc. of the World Acad. Sci. Eng. Technol.* Vol 15, vol. 15, no. 10, pp. 285–289, 2006.

[12] R. Malhotra and A. Chug, "Application of Group Method of Data Handling model for software maintainability prediction using object oriented systems", *Int. J. Syst. Assur. Eng. Manag.*, Vol. 5, No. 2, pp. 165–173, 2014.

[13] R. V. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems", *Int. J. Ind. Eng. Comput.*, Vol. 7, No. 1, pp. 19–34, 2016.