Date of publication SEP-30, 2021, date of current version MAY-12, 2021.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.20.3.2280

Hybrid Maintainability Prediction using Soft Computing Techniques

MANJU DUHAN, PRADEEP KUMAR BHATIA
Guru Jambheshwar University of Science &Technology, Hisar
(e-mail: duhan.manju@gmail.com, pkbhatia.gju@gmail.com)

Corresponding author: Manju Duhan (e-mail: duhan.manju@gmail.com).

The authors are grateful for support from the Guru Jambheshwar University of Science and Technology, for providing a lab facility to run various tools used in the current study.

ABSTRACT Effective software maintenance is a crucial factor to measure that can be achieved with the help of software metrics. In this paper, authors derived a new approach for measuring the maintainability of software based on hybrid metrics that takes advantages of both i.e. static metrics and dynamic metrics in an object-oriented environment whereas, dynamic metrics capture the run time features of object-oriented languages i.e. run time polymorphism, dynamic binding etc. which is not covered by static metrics. To achieve this, the authors proposed a model based on static and hybrid metrics to measure maintainability factor by using soft computing techniques and it is found that the proposed neuro-fuzzy model was trained well and predict adequate results with MAE 0.003 and RMSE 0.009 based on hybrid metrics. Additionally, the proposed model was validated on two test datasets and it is concluded that the proposed model performed well, based on hybrid metrics.

KEYWORDS Neuro-fuzzy; Neural Network; Dynamic; Maintainability.

I. INTRODUCTION

SOFTWARE maintenance is the most critical activity in the software development life cycle. It can consume around 70% of the cost of the entire life cycle [1]. Thus, the right maintainability prediction can reduce the cost of the software product. Therefore, evaluating maintainability in the early phases of the software development life cycle is essential and beneficial in terms of cost, productivity, time and efforts required to build different projects. Many Object-Oriented metrics exist to compute the maintainability factor of a software system in an Object-Oriented environment. Static metrics provide early detection of faults and cost benefits, whereas dynamic metrics show the actual software behaviour by capturing features like run time polymorphism and dynamic binding. Besides, Dynamic metrics evaluation is a time-consuming task. Therefore, the hybrid approach saves time and effort and covers all the features that affect software maintainability. Therefore, in this paper, authors mainly have chosen six factors: complexity, coupling, cohesion, inheritance, the response between classes and size, to evaluate maintainability as a function of change using static and hybrid metrics (a combination of static and dynamic metrics). In the case of static metrics, we have chosen six metrics, i.e. WMC, DIT, LCOM, RFC, CBO and CLOC whereas, in the case of hybrid metrics, we took three metrics from the static metrics set, i.e. RFC, CBO and DIT and three dynamic metrics, i.e. DLCOM, DWMC and DLOC, as described in Table 1. We have chosen these metrics based on the correlation of these metrics with external quality factor, i.e. maintainability.

Further, there is no direct relationship between maintainability attributes. Therefore, the neural network (NN) [2] approach provides adaptive learning capabilities to predict software maintainability, whereas fuzzy logic can generalize rules. To take advantage of both, we have proposed a neuro-fuzzy (NF) approach [3] to determine a class's maintainability effectively. Further, the authors also compared the proposed NN and NF model with four existing machine learning algorithms as described below.

Table 1. Static and dynamic metrics used in the current study

Static Metrics	**Description**
WMC	"Sum of the complexity of the methods of a class" [4]
DIT	"The max. length from the node to the root of the tree" [4]
LCOM96a	"Number of attributes invoked by all the methods in a class at compile time" [5]
RFC	"Numbers of methods invoked from a class" [4]
CBO	"Number of other classes to which it is coupled" [4]
CLOC	"The number of all nonempty, non-commented lines of the body of the class"
Dynamic Metrics	**Description**
DLCOM	"It is defined as the combination of RLCOM (Runtime Lack of cohesion of methods), RAAR (Runtime Attribute Access Rate) and RMMC(Runtime method to method call)" [6]
DWMC	"Number of times methods of class executed at runtime"
DLOC	"Number of times lines of a class executed at runtime"

A. RANDOM FOREST

Random Forest is a supervised learning algorithm used in classification and regression but most commonly used in classification. This algorithm builds the decision tree using each data sample and takes the prediction of each decision tree. Then voting is performed on every predicted result and selects that prediction which gets the maximum vote and gives the best result.

B. LINEAR REGRESSION

The Linear Regression model mainly based on the concept of best fit by finding the relationship between attributes. This algorithm also comes under the category of supervised learning classification algorithms.

C. SMOreg

Sequential Minimal Optimization Regression algorithm performed best for solving quadratic programming problems to train support vector machines. This algorithm also comes under the category of supervised learning classification algorithms that takes advantage of the regression algorithm.

D. MULTILAYER PERCEPTRON

MLP is a supervised learning classification algorithms that use a back-propagation algorithm for training of attributes feed as input to the artificial neural network. It uses multiple layers and many folds instead of a single layer perceptron that enhance its accuracy.

The rest of the paper divides into mainly four sections. Section 2, presents a literature review and research methodology followed in this paper. Section 3, presents the proposed formula and proposed model to measure maintainability using soft computing techniques. Section 4, explains the experimental study done on HoDoKu software and comparison of various soft computing techniques. Section 5, describes the conclusion and future directions referenced to the current study.

II. RESEARCH WORK

A. RESEARCH BACKGROUND

Software design metrics evaluate the software maintainability in the early phases of software development life cycle (SDLC) that lack in handling run time features of object-oriented languages like dynamic binding and run time polymorphism. Many empirical studies exist in the literature to compute software maintainability using static design metrics [7-16]. However, significantly less emphasis was done to map dynamic metrics to the software's maintainability factor. AI-Jamimi et al., 2012 [16] proposed a fuzzy logic model based on static metrics and applied it on two object-oriented datasets. Alijamaan et al., 2013 [17] proposed an ensemble model using four other machine learning algorithms, i.e. MLP, RBF, SVM and M5P, to find the maintainability of object-oriented software. They applied these models on different datasets, which also took static metrics, i.e. CK, Li and Henry and size, as its base. Baqais et al., 2013 [18] proposed neural network and genetic algorithms implementation to find maintainability estimation by applying it on Andriod, an open-source software, using the same set of static metrics as input to their models used by previous authors. Malhotra et al., 2014 [19] proposed a Group method of data handling (GMDH) model and compared this model with the other two models, i.e. FF3LBPN and GRNN and concluded that the proposed GMDH model was best among all with the least errors. They also used the same set of metrics as the input used by previous author's, i.e. static metrics. Sharma et al., 2015 [20] compared the static and dynamic metrics approach to find maintainability of software by using four machine learning algorithms, i.e. MLP, Linear Regression, SMOreg Gaussian process. The authors concluded that the Linear regression model performed best among all dynamic metrics to find software's maintainability. Kumar et al., 2017 [3] proposed a neuro-fuzzy model to evaluate software maintainability by applying it on two commercial software, i.e. UIMS and QUES, using static metrics. Authors used PCA and RSA theory for feature reduction and concluded that these techniques found the maintainability of software with higher accuracy. Hence, from the existing literature survey, it is observed that all the studies made by various researchers took static metrics as input and very little work done on dynamic metrics. Therefore, in this paper, the authors choose hybrid metrics that are a combination of static and dynamic metrics to predict software maintainability that consumes less time and provides higher accuracy. Further, we observed that machine learning algorithms, i.e. Random Forest and hybrid approach, i.e. neuro-fuzzy, were not explored much. Therefore, the authors proposed a neural network and neuro-fuzzy model based on static and hybrid metrics and compared them with four other existing machine learning

models, i.e. MLP, Random Forest, SMOreg and Linear regression [20].

B. RESEARCH METHODOLOGY

The methodology implemented in the current study is shown in Figure 1. The authors worked on two versions of HoDoKu, open-source software i.e. HoDoKu 2.0.1(2010) and HoDoKu 2.2(2012) that were compared on the same source code. Based on that, 63 classes were evaluated. Static metrics were collected by using the CodeMR tool [21], an Eclipse plugin and dynamic metrics were collected by using the AspectJ [22] tool, an implementation of aspect-oriented programming [23] on the Eclipse platform [24]. Aspects were created and run with Java classes using AspectJ to extract dynamic metrics without affecting the functioning of original Java classes which is the best feature of aspect-oriented programming. The authors also applied four existing machine learning algorithms (MLA) by using WEKA 3.8 tool [25] on static and hybrid metrics. Results were analysed based on Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [20] values obtained by comparing the various machine learning algorithms [26, 27].

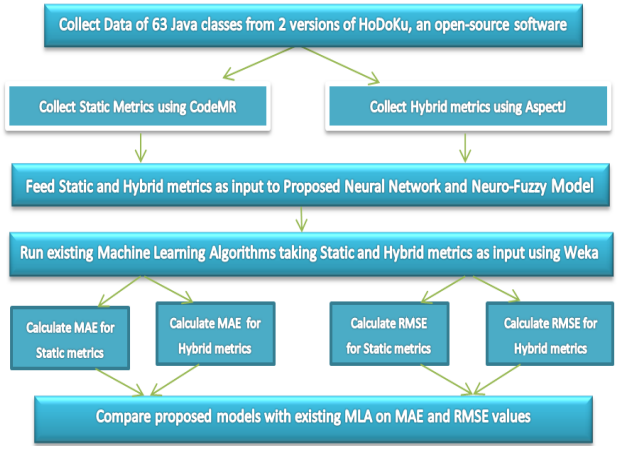

Figure 1. Research Methodology followed in the current study

III. PROPOSED WORK

A. PROPOSED FORMULA

A formula is proposed based on metrics described in section 1. Static and Hybrid metrics were considered as independent variables whereas maintainability i.e. a function of change was considered as a dependent variable. The Change was counted by comparing two consecutive versions of software whereas addition and deletion of code were counted as one changFe while updating of code was counted as two changes [3]. The proposed formula is defined in equation (1) and (2) whereas in equation (2) combination of static and dynamic metrics were used i.e. three metrics DIT, RFC and CBO were static metrics and DWMC, DLCOM and DLOC were dynamic metrics.

$$\textit{Maintainability for static metrics} = f(WMC, DIT, RFC, CBO, LCOM, LOC). \quad (1)$$

$$\textit{Maintainability for Hybrid metrics} = f(DWMC, DIT, RFC, CBO, DLCOM, DLOC). \quad (2)$$

B. PROPOSED NEURAL NETWORK MODEL

The proposed neural network [2] was trained with six inputs and one output i.e. software maintainability. Trainbr was used as a training function, the number of neurons at the hidden layer was set to 30 and transit function was used as a transfer function as described in Table 2.

The proposed ANN as shown in Figure 2 was trained on raw data sets by the standard error back-propagation algorithm at a learning rate of 0.006, having the mean squared error as the training stopping criterion. The network divides the 63 samples into three parts i.e. 45 samples (70%) for training, 9 samples (15%) for testing and 9 samples (15%) for validating the neural network.

Table 2. Proposed neural network Description

Input units	6
Output units	1
No. of neurons at hidden layer	30
Algorithm	Back propagation
Training function	Trainbr
Network ratio	3:1:1

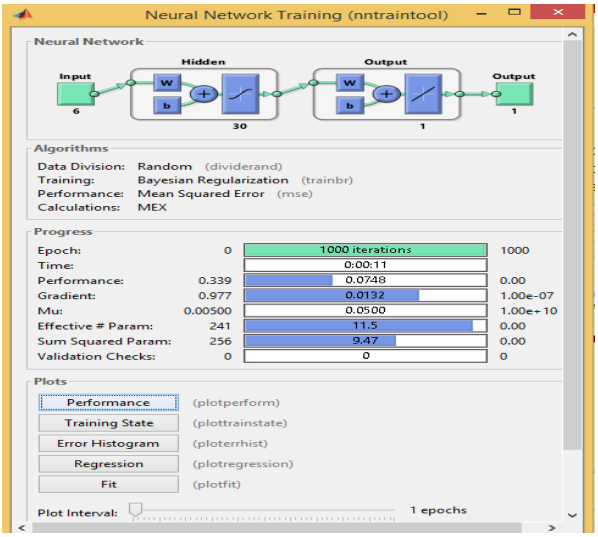

Figure 2. Proposed Neural Network Model

C. PROPOSED NEURO-FUZZY MODEL

Neuro-fuzzy is a hybrid system [3] that takes the benefit of a fuzzy logic approach that provides flexibility to a system rather than crisp logic and a neural network that can learn by itself. Therefore, we can say that the neuro-fuzzy approach is a mixture of implicit and explicit knowledge. In this paper, the neuro-fuzzy model was applied with the help of the MATLAB tool using the ANFIS editor. Firstly, raw data was

loaded into the ANFIS editor by using the load data option and set the type of the data as training data in the load data part of the ANFIS editor. After successful loading of data, a message appeared on the ANFIS editor screen that train data loaded. In this paper, 63 Java classes were loaded successfully in the ANFIS editor for training having 6 numbers of inputs and one output as shown in Figure 3.

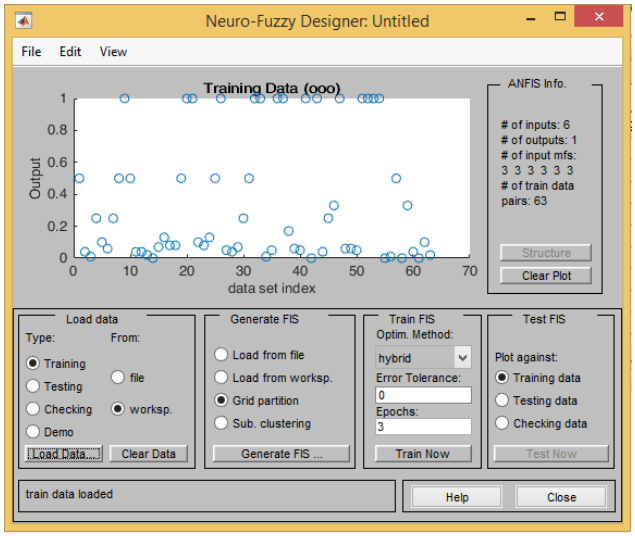

Figure 3. Loaded data in the neuro-fuzzy model

After the successful loading of data, we can see the structure of the proposed neuro-fuzzy model having 6 inputs and one output i.e. maintainability. After that, the authors generated a Sugeno fuzzy inference system having three membership function (gaussmf) i.e. low, medium and high using grid partitioning method and then train the generated FIS using a hybrid optimization method by setting the number of epochs to 60. Further, trained ANFIS was tested on 2 validation datasets by setting the load data type to testing and again authors generated the FIS on testing data. The detailed description of the proposed neuro-fuzzy model is shown in Table 3.

Table 3. Proposed Neuro-Fuzzy Model Description

Input units	6
Output units	1
No of train data pairs	63
No. of fuzzy rules generated	749
FIS model	Sugeno
FIS training optimization method	Hybrid
FIS input membership function	Gaussmf (Low, Medium, High)
FIS output membership function	Linear
FIS generation method	Grid Partitioning
No. of epochs	60

D. MACHINE LEARNING ALGORITHMS USED IN CURRENT STUDY

Authors have also used four machine learning algorithms i.e. Linear Regression, Multilayer Perceptron, Random Forest and SMOreg in the current study to find the maintainability factor of software. The authors applied these algorithms by using WEKA 3.8 tool [25]. Firstly, static and hybrid metrics raw data were normalized using the min-max normalization [12] technique to reduce the complexity of attributes taken as input to machine learning models and processing speed got faster to measure the desired output. After that, the classification of processed data was done by setting the cross-validation folds value to 30 as shown in Figure 4.

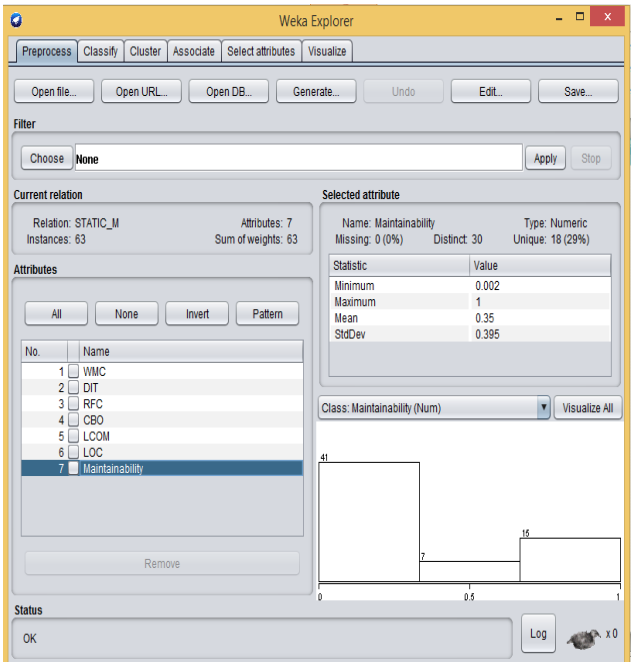

Figure 4. Loaded data of static metrics in WEKA 3.8 tool

IV. ANALYSIS RESULTS

An experimental study was conducted on HoDoKu, open-source software by comparing 2 consecutive versions of it. Static and hybrid metrics were extracted from classes having the same source code in both versions. Static metrics were extracted using the CodeMR tool and dynamic metrics were extracted using AspectJ, an implementation of aspect-oriented programming on the eclipse platform. Aspect-oriented programming is used as the process of building a tracing or profiling framework using the aspect-oriented approach that is relatively simpler than any other approach. Further, statistics were applied to calculated values of metrics using the MATLAB tool [2]. Functions named min( ), max( ), mean( ) and std( ) used in MATLAB to find a minimum, maximum, mean and standard deviation of measured metrics values, respectively. The statistical result on metrics values are shown in Table 4 and Table 5, respectively.

Table 4. Statistical data of Static Metrics

Metric Name	MIN	MAX	MEAN	STD. DEV.
WMC	0	749	64	124.29
DIT	1	6	3.41	2.15
RFC	0	444	72.65	80.04
CBO	0	21	3.47	4.26
LCOM(96a)	0	1	0.71	0.31
CLOC	3	2336	290.57	394.25
CHANGE	0	607	44.60	111.06

Table 5. Statistical data of Hybrid Metrics

Metric Name	MIN	MAX	MEAN	STD. DEV.
DWMC	0	450	63.45	98.34
DIT	1	6	3.36	2.18
RFC	0	444	76.07	91.56
CBO	0	21	3.65	4.79
DLCOM	0	0.99	0.65	0.31
DLOC	2	1040	225.63	253.39
CHANGE	0	607	44.60	111.06

From these descriptive statistics some observations were recorded as mentioned below:

- WMC and DWMC showed a great difference in their values that indicates all the methods of a class were not executed at run time. Hence, the complexity of methods at a run time decreases when compared with the compile-time complexity of methods that affects the maintainability of software and makes the DWMC an essential metric to predict the maintainability of software.
- LCOM and DLCOM values were nearly the same in most of the cases and showed high variation in some cases that shows cohesion of classes is not predicted well by static metrics. Therefore, DLCOM has its importance to find the maintainability of software.
- Size of code is decreased by almost half at run time that is computed well by DLOC metric as compared to size computed at compile time by CLOC metric.

A. COMPARISON RESULTS

Static and Dynamic Metrics were analysed on six soft computing techniques including the proposed neural network and neuro-fuzzy model and results indicate that the proposed neuro-fuzzy model performed well and better predicts the maintainability of software in the case of hybrid metrics as shown in Table 6. From Table 6, it is observed that hybrid metrics provides better results irrespective of any machine learning prediction model as compared to static metrics.

Table 6. Comparison of various classification algorithms

	Static metrics			Hybrid metrics		
Classification Models	MAE	RMSE	Build Time	MAE	RMSE	Build Time
Linear Regression	0.31	0.37	0.12	0.30	0.38	0.11
Multilayer Perceptron	0.30	0.45	0.04	0.27	0.33	0.04
SMOreg	0.31	0.39	0.34	0.30	0.38	0.33
Random Forest	0.22	0.30	0.16	0.20	0.28	0.12
Proposed Neural Network Model	0.27	0.33	0.11	0.19	0.27	0.10
Proposed Neuro-Fuzzy Model	0.01	0.05	0.45	0.003	0.009	0.44

From Table 6, the authors observed that the highest value of MAE is 0.31 for static metrics and 0.30 for hybrid metrics using Linear Regression and SMOreg models that indicate that these models have the highest difference between predicted and actual value. The lowest value of MAE is 0.01 for static metrics and 0.003 for hybrid metrics using the neuro-fuzzy model that indicates that the proposed neuro-fuzzy model was trained well and has minimized the differences between predicted and actual values more in the case of hybrid metrics. In contrast, time taken to build the neuro-fuzzy model is highest as compared to other classification models and least in the case of MLP model that shows there is a trade-off between time and accuracy factor of various models.

Comparison of MAE and RMSE values for static and hybrid metrics is shown in Figure 5 and Figure 6 respectively that clearly shows that hybrid metrics are better predictors of maintainability as compared to static metrics irrespective of any classification algorithm.

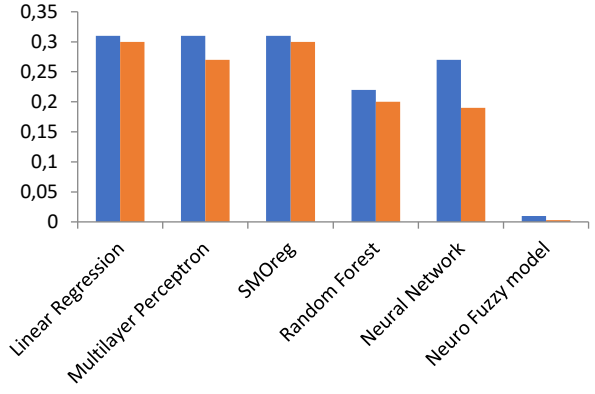

Figure 5. Comparison of MAE values of various classification algorithms.

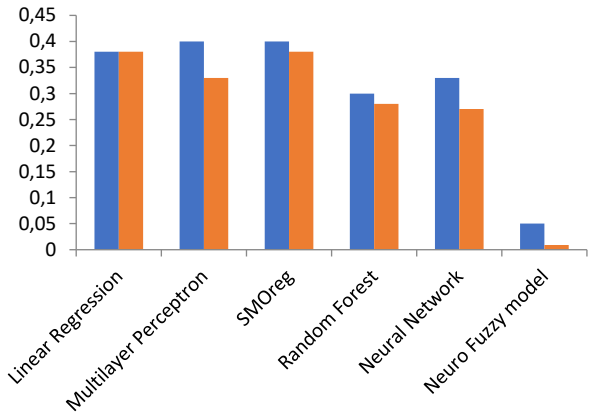

Figure 6. Comparison of RMSE values of various classification algorithms

B. VALIDATION OF PROPOSED MODEL

The six machine learning models were validated on 2 datasets and the result indicates that the neuro-fuzzy model was trained well that gave MAE 0.01 for test data set 2 in case of static metrics as shown in Table 7 and 0.001 for test data set 1 in case of hybrid metrics as shown in Table 8 that indicates hybrid metrics better predicts maintainability of software and gives higher accuracy than static metrics.

Table 7. Validation of various classification algorithms based on Static Metrics

	Static metrics			
	Validation Test 1		Validation Test 2	
Classification Models	MAE	RMSE	MAE	RMSE
Linear Regression	0.22	0.28	0.33	0.39
Multilayer Perceptron	0.23	0.31	0.25	0.34
SMOreg	0.24	0.32	0.29	0.438
Random Forest	0.19	0.27	0.15	0.23
Proposed Neural Network Model	0.27	0.31	0.30	0.40
Proposed Neuro-Fuzzy Model	0.03	0.09	0.01	0.03

Table 8. Validation of various classification algorithms based on Hybrid metrics

	Hybrid metrics			
	Validation Test 1		Validation Test 2	
Classification Models	MAE	RMSE	MAE	RMSE
Linear Regression	0.21	0.27	0.32	0.39
Multilayer Perceptron	0.19	0.30	0.13	0.16
SMOreg	0.20	0.28	0.28	0.40
Random Forest	0.07	0.11	0.09	0.13
Proposed Neural Network Model	0.13	0.20	0.20	0.31
Proposed Neuro-fuzzy Model	0.001	0.003	0.004	0.007

The authors compared the MAE values of dataset 1 and dataset 2 as shown in Figure 7 and 8, respectively based on static and hybrid metrics. It is found that the value of MAE was lower in the case of hybrid metrics irrespective of any machine learning algorithm and gave the best results in the case of hybrid approach i.e. neuro-fuzzy.

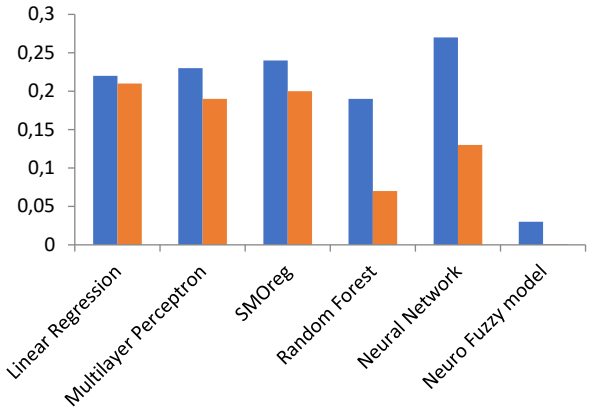

Figure 7. Comparison of MAE values of Classification models on data set 1

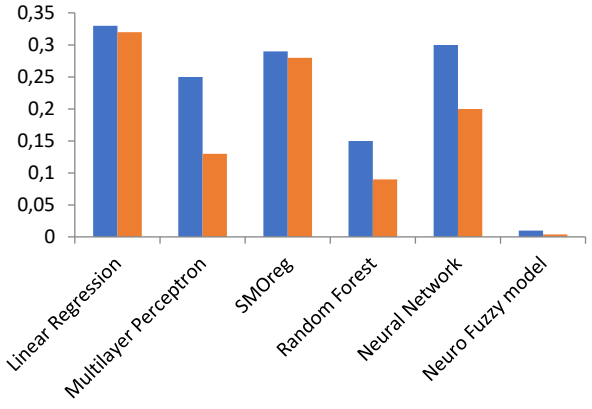

Figure 8. Comparison of MAE values of Classification models on data set 2

V. CONCLUSIONS

The main purpose of the current study was to analyze the usefulness of hybrid metrics for maintainability estimation in an object-oriented environment that takes the advantages of both i.e. static metrics and dynamic metrics that takes less time and effort to compute maintainability of software with higher accuracy. Comparison of various soft computing techniques was done with the proposed neural network and neuro-fuzzy approach based on static and hybrid metrics by collecting metrics from HoDoKu, open-source software. The proposed model has taken static and hybrid metrics as input and maintainability as output i.e. function of change from two consecutive versions of HoDoKu on the same source code. Based on that, 63 Java classes were compared based on mean absolute error (MAE) and root mean square error (RMSE) values and it is found that hybrid metrics performed utterly well for all the machine learning algorithms as compared to static metrics to estimate the software maintainability. Further, it is concluded that the proposed

neuro-fuzzy model was trained well and gave satisfactory results in the case of hybrid metrics with 0.001 mean absolute error (MAE) on validation test data set 1. Hence, this study would help the software industry to predict the maintainability of software in advance with less time and higher accuracy with the help of hybrid metrics. In future, the model will be more sophisticated by taking a large set of metrics on large projects in Object Oriented environment.

References

[1] R. S. Pressman, *Software Engineering – A Practitioner's Approach*, 7th ed., McGraw Hill, 2005.

[2] MATLAB Neural Network Tool Box 2016 Product Help.

[3] L. Kumar, S.K. Rath, "Software maintainability prediction using hybrid neural network and fuzzy logic approach with parallel computing concept," *International Journal of System Assurance Engineering and Management*, Springer, vol. 8, pp. 1487–1502, 2017. https://doi.org/10.1007/s13198-017-0618-4.

[4] S. R. Chidamber, C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol.20, issue 6, pp. 476-493, 1994. https://doi.org/10.1109/32.295895.

[5] tusharma.in/technical/revisiting-lcom/

[6] Manju, P. K. Bhatia, "Measurement of dynamic cohesion using aspect oriented approach," *International Journal of Research and Analytical Reviews (IJRAR)*, vol. 6, issue 2, pp. 438-432, 2019.

[7] W. Li, S. Henry, "Maintenance metrics for the object-oriented paradigm," *Proceedings of the First IEEE International Software Metrics Symposium*, 1993, pp. 52–60.

[8] P. Oman, J. Hagemeister, "Construction and testing of polynomials predicting software maintainability," *Journal of Systems and Software*, Elsevier, vol. 24, issue 3, pp. 251–266, 1994. https://doi.org/10.1016/0164-1212(94)90067-1.

[9] S. L. Schneberger, "Distributed computing environments: effects on software maintenance difficulty," *Journal of Systems and Software*, Elsevier, vol. 37, issue 2, pp. 101–116, 1997. https://doi.org/10.1016/S0164-1212(96)00107-0.

[10] R. Kohavi, "Relation between software metrics and maintainability," *Proceedings of the FESMA99 International Conference, Federation of European Software Measurement Associations*, Amsterdam, The Netherlands, vol. 1, pp. 465–476, 1999.

[11] M. Dagpinar, J.H. Jahnke, "Predicting maintainability with object oriented metrics – An empirical comparison," *Proceedings of the 20th Working Conference on Reverse Engineering (WCRE)*, 2003, pp. 155–164.

[12] K. K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, "Application of artificial neural network for predicting maintainability using object-oriented metrics," *Proceedings of the World Academy of Science, Engineering and Technology*, vol. 15, pp. 140–144, 2006.

[13] Y. Zhou, H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression spline," *Journal of Systems and Software*, Elsevier, vol. 80, issue 8, pp. 1349–1361, 2007. https://doi.org/10.1016/j.jss.2006.10.049.

[14] M. O. Elish, K. O. Elish, "Application of TreeNet in predicting object-oriented software maintainability: a comparative study," *Proceedings of the 13th European Conference on Software Maintenance and Reengineering, CSMR'09*, 2009, pp. 69–78. https://doi.org/10.1109/CSMR.2009.57.

[15] C. Jin, J.-A. Liu, "Applications of support vector machine and unsupervised learning for predicting maintainability using object oriented metrics," *Proceedings of the IEEE Second International Conference on Multimedia and Information Technology*, Kaifeng, 2010, pp. 24-27. https://doi.org/10.1109/MMIT.2010.10.

[16] H. Al-Jamimi, M. Ahmed, "Prediction of software maintainability using fuzzy logic," *Proceedings of the 3rd International Conference on Software Engineering and Service Science (ICSESS)*, 2012, pp. 702–705. https://doi.org/10.1109/ICSESS.2012.6269563.

[17] H. Aljamaan, M. O. Elish, I. Ahmad, "An ensemble of computational intelligence models for software maintenance effort prediction," *Proceedings of the Advances in Computational Intelligence*, 2013, pp. 592–603. https://doi.org/10.1007/978-3-642-38679-4_60.

[18] A. A. B. Baqais, M. Alshayeb, Z. A. Baig, "Hybrid intelligent model for software maintenance prediction," *Proceedings of the World Congress on Engineering*, 2013, vol. 1, pp. 358-362.

[19] R. Malhotra, A. Chug, "Application of group method of data handling model for software maintainability prediction using object oriented systems," *Int. J. Syst. Assur. Eng. Manag.*, Springer, vol. 5, issue 2, pp. 165–173, 2014. https://doi.org/10.1007/s13198-014-0227-4.

[20] H. Sharma, A. Chug, "Dynamic metrics are superior than static metrics in maintainability prediction: An empirical case study," *Proceedings of the 4th IEEE International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)*, pp. 1-6, 2015. https://doi.org/10.1109/ICRITO.2015.7359354.

[21] CodeMR guide (2020), https://www.codemr.co.uk/docs/codemr-intellij-userguide.pdf

[22] AspectJ, 2020, http://www.eclipse.org/aspectj

[23] AspectJ tutorial, 2020. https://o7planning. org/en/10257/Java-aspect-oriented-programming-tutorial-with-aspectj

[24] Eclipse guide, 2020. https://www.eclipse.org/aspectj/doc/next/progguide/printable.html

[25] S. S. Aksenova, "Machine Learning with WEKA", WEKA Explorer Tutorial, 2004.

[26] J. Han, M. Kamber, *Data Mining Concepts and Techniques*, 2nd ed., Elsevier, 2006.

[27] J. Ulrich, *Supervised Machine Learning for Email Thread Summarization*, Master's Thesis, University of British Columbia, Vancouver, Austin, 2006.

MANJU DUHAN *is pursuing a PhD degree at Guru Jambheshwar University of Science and Technology, Hisar, India. Her research interests are Neural networks, Machine learning, Neuro-fuzzy, Software Quality Metrics in Object Oriented environment.*

PRADEEP KUMAR BHATIA, *Doctor of Sciences, a Professor, Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, Hisar, India. His research areas are Software Engineering, Computer graphics, Fuzzy and Soft Computing.*