ADV MATH SCI JOURNAL
Advances in Mathematics: Scientific Journal **10** (2021), no.1, 543–555
ISSN: 1857-8365 (printed); 1857-8438 (electronic)
https://doi.org/10.37418/amsj.10.1.54 Spec. Iss. on ICIRPS-2020

PREDICTING SOFTWARE BUGS OF NEWLY AND LARGE DATASETS THROUGH A UNIFIED NEURO-FUZZY APPROACH: RELIABILITY PERSPECTIVE

Kavita Sahu[1] and R.K. Srivastava

ABSTRACT. Reliability of software is an essential concern for users for a long time. Software reliability is mainly obtained through modeling and estimating. There are numerous methods for reducing the failure rate. However, the existing methods are nonlinear. Hence the parameter estimation of these methods is difficult. This paper concerns on estimation and prediction of software reliability through different soft computing methods for improving the reliability of software. For estimation and prediction, the authors of this paper take two soft computing methodologies, including fuzzy logic and neural network. The outcomes seem to give satisfactory results on large datasets. For experiments, this paper is using two different large datasets of Apache server and MyLyn application software for showing the effectiveness of the results. The proposed methods of prediction would be useful for practitioners to simplify the procedures during software development in large datasets for reducing failures of software.

1. INTRODUCTION

Software reliability is the area of research in software engineering for more than forty years, but still, some questions are unanswered [1]. Such as "Are we

[1]*corresponding author*

2020 *Mathematics Subject Classification.* 68M15, 68T07.

Key words and phrases. Software Reliability, Reliability Prediction, Soft Computing, Fuzzy Logic, Neural Network.

Submitted: 18.11.2020; *Accepted:* 25.12.2020; *Published:* 22.01.2021.

getting a fully reliable software?", "Are we managing software reliability more efficiently or have been working on the same methods for years" ? The disastrous failure occurred on January 25, 2017, of GPS satellite led to a software error that only occurred for a mere 13 microseconds. However, the result had a significant impact on global positioning systems (GPS), the U.S. Air Force, and communications networks, although the fault was momentary [2]. Such type of failures makes reliability factor an essential and significant area of research which further assures more of software issues such as security, efficiency, and usability as well. Software reliability is defined as the probability of failure-free operation of the software over a specified period in a specified environment [3].

Software reliability cannot be directly measured because failure-free operations occurred during a specific period in a specified environment. Failures and faults of the software are found in its different software development and reliability factors. This is why measuring software reliability is hard because the measurement of its specific attributes is necessary to measure [2]. Failure counts, mean time to failure, mean time to recover, etc. are the attributes on which reliability of software is measured. Until now, there is no right way to conquer the complexity problem of software. In the last few years, many research studies have been carried out in this area of software reliability modeling and forecasting. The issues of software reliability are focused on soft computing techniques for developing and maintaining software systems to enhance it with proper measurement. In this paper, we have used a neural network as an estimation model and compared its results with the hybrid method of Neuro-fuzzy. These results are obtained using MATLAB. Rizvi et al. [4]and Yazdani et al. [5] used soft computing methods and statistical methods to quantify and estimate software reliability. However, the frameworks approached were not practically applicable. Also, soft computing methods provide more crisp results than any other approach. Hence this paper is proposing a hybrid approach of Neuro-fuzzy with application to two real-time databases and compared the results with simple neural network methods.

In this paper,the authors tried to propose a model of the hybrid methodology of neuro-fuzzy to predict the software reliability for the two different datasets. Application software and server dataset, which have different failure patterns than older datasets. This assessment would also be of great help to designers and developers in recognizing the goals of the attributes and making correct decisions while assessing software reliability. Practical assessment of reliability is not only beneficial for reliability assessment but also improves the overall software quality. The critical aspects of this study are:

(i) Analyze software reliability from an industrial perspective with the intent to include guidance for the production of reliable applications.
(ii) The validity of the approach is tested using the Neuro-fuzzy and neural network methodology. All methods in the soft computing problem-solving domain are well known and accessible. The technique proposed in the study delivers accurate and efficient results.
(iii) Two different datasets were taken as alternatives for this case study to test software reliability.
(iv) The analytical initiative of this study seeks to provide insights into how structured and well-proven reliability design techniques are practiced during the life cycle of software development.

The rest of the paper is as follows: The Second section is describing the hybrid methodology of Neuro-fuzzy, which is used in this research. The third section focuses on the experimental implementation of the methodology. Finally, results, discussion, and conclusion are described in sections forth and five.

2. Methodology Followed

Our work is divided into two parts. The first part is prediction through the neural network, and the second part focuses on prediction through the neuro-fuzzy methodology. Further, the prediction is divided into three phases: Analysis and design phase (Dataset preparation and network creation), Implementation phase (Implementing neural and neuro-fuzzy on the dataset), and Validation phase (Comparison of results with other methods). The description of each method is given in the following section:

2.1. **Data Collection.** To evaluate the proposed model, two datasets are used. One dataset is for server Apache Lucene which contains 600 failure numbers of class data with eight attributes. Another dataset of MyLyn application software with 18 attributes and 1856 classes of failure. In order to reach the software reliability level, two essential attributes, which are several bugs and the number of non-trivial bugs, are taken in this work.

2.2. **Prediction using Neural Network.** Neural Networks are simplified models of the biological nervous system and, therefore, have drawn their motivation from the example of computations performed by human brains [7–9]. A neural network contains characteristics such as mapping capabilities, generalization, robustness, fault tolerance, etc. That is why the neural network is highly used in modeling a reliable system. Neural networks have broadly classified as a single layer, multi-layer, and recurrent neural networks. A Neural Network without the Activation function would simply be a linear regression model. A Multilayer perceptron is a supervised learning algorithm that learns a function by training on a dataset. MLP is perhaps the most popular and most widely used ANN, which consists of two layers, input and output, and one or more hidden layers between these two layers. The hidden layers are introduced to increase the network's ability to model complex functions. Every layer in a network contains enough neurons, depending on the application. The input layer is passive and receives only the data (e.g., data relating to different causative factors). Unlike the input layer, the data are actively processed both by hidden and output layers. The output layer yields the results for the neural network.

Thus, the number of neurons in the layers of input and output is usually fixed by the application type. Typically, the number of hidden layers and their neurons is determined by trial and error for a classification problem. There are three stages involved in ANN data processing: the training stage, the weight determination stage, and the classification stage. The training process is initiated by assigning arbitrary initial weights of connection, which are updated continuously until an acceptable training accuracy is achieved. The weights adjusted obtained from trained neurons in its input layer may be expressed as Eq (2.1),

$$N_j = \Sigma_{i=1}^{n} wt_{ij} d_i, \tag{2.1}$$

where wt represents the weights belonging to input neuron i and hidden neuron j, di is the data at the input neuron, and n is the number of neurons. Further, the transfer function T is expressed as Eq (2.2),

$$d_i = T(net_j) = \frac{1}{1 + e^{-netj}} n. \tag{2.2}$$

The function T is usually a nonlinear sigmoid function that is applied to the weighted sum of the input data before the data are processed to the next layer.

To predict software reliability using a neural network following are the essential steps to be taken.

Step 1. The first step is to import data in the proper format and specify the activation function, which will be used to activate the neuron. In this work, we are using the sigmoid function to activate the inputs. The sigmoid function is of form, as shown in Eq (2.3),

$$f(x) = \frac{1}{1+e^{-x}}. \tag{2.3}$$

Step 2. The next step is to create the neural network according to the datasets. Split the variable according to the dataset and design the neural network according to the inputs.

Step 3. Further step is to decide the number of hidden layers be used for learning and learning rate. The hidden layer can range from 1 to maximum accuracy the user wants. Several hidden layers decide accuracy. In this work, we are using Multi-Layer Perceptron Network (MLP) neural network. It is a feed-forward artificial neural network and supervised algorithm. The usage of MLP neural networks is because of its capability to learn with non-linear models and models with real-time data. It contains further steps

- Calculate the Error : How far is the model output from the actual output.
- Minimum Error : Check whether the error is minimized or not.
- Update the Parameters : If the error is enormous,then update the parameters (weights and biases). After that, again, check the error. Repeat the process until the error becomes minimum.
- Model is ready to make a Prediction : Once the error becomes minimum, one can feed some inputs to one$'s$ model, and it will produce the output.

2.3. **Prediction using Neuro-Fuzzy.** Fuzzy logic is capable of handling predictions with incomplete or imprecise data. It can also model non-linear functions of arbitrary functions. Fuzzy logic starts with the concept of a fuzzy set. A fuzzy set is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership. A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the universe of discourse, a fancy name for a simple concept. The only condition a membership function must satisfy is that it must vary between 0 and 1. The function itself can be an arbitrary curve whose shape we can define as a function that suits us from simplicity, convenience, speed, and efficiency. A classical set might be expressed as Eq (2.4),

$$A = \{x|x > 6\}. \tag{2.4}$$

A fuzzy set is an extension of a standard set. If X is the universe of discourse and its elements are denoted by x, then a fuzzy set A in X is defined as a set of ordered pairs, which is shown in Eq (2.5),

$$A = \{x, \mu_A x | x \varepsilon X\}. \tag{2.5}$$

Then, $\mu_A x$ is called the membership function (or MF) of x in A. The membership function maps each element of X to a membership value between 0 and 1. The most straightforward membership functions are formed using straight lines. Of these, the simplest is the triangular membership function, and it has the function name trimf. Triangular membership functions defined by a lower limit a, an upper limit b, and a value m, where a < m < b. The formulas for trimf is given below in Eq (2.6),

$$\mu_A x = \begin{cases} 0, & \text{x} \leq a \\ \frac{x-a}{m-a}, & \text{a} < x \leq m \\ \frac{b-x}{b-m}, & \text{m} < x < b \\ 0, & \text{x} \geq b \end{cases} \tag{2.6}$$

TFN is nothing more than a collection of three points forming a triangle. The trapezoidal membership function, trapmf, has a flat top and is just a truncated triangle curve. These straight-line membership functions have the advantage of simplicity. The methodology for the hybridization of Neural and Fuzzy techniques is described in the next section.

In a wide variety of real-world problems, hybrid systems incorporating fuzzy logic and neural networks prove their effectiveness. That smart technique has unique computational properties (e.g., learning capacity, decision explanation) that make them ideal for specific problems and not others. While neural networks, for example, are good at recognizing patterns, they are not good at explaining how to reach their decisions. Fuzzy logic systems are good at explaining their decisions, which can reason with imprecise knowledge, but they cannot immediately learn the rules they use to make those decisions. When considering the varied nature of application domains, hybrid systems are also essential. Many complex domains have many different problems with components, each of which may require different processing types.

A neuro-fuzzy framework is a fuzzy framework that uses attacking in calculation got from or propelled by a neural system hypothesis to focus its limitations (fuzzy sets and fuzzy rules) by preparing information tests. Neuro-fuzzy was proposed by J. S. R. Jang [10]. Neuro-Fuzzy System synergizes these two procedures by consolidating the human-like thinking style of fuzzy frameworks with the learning and connection of the structure of neural systems. The types of Neuro-fuzzy Systems are

- Mamdani based Neuro-Fuzzy inference system;
- Takagi Sugeno fuzzy inference system.

A neuro-fuzzy framework makes utilization of neural systems to give fuzzy frameworks a sort of programmed tuning strategy by not influencing their functionalities. There is a preparation process, where a neural system modifies its weights to minimize the mean square error between the yield of the system and the desired output. Mamdani fuzzy inference framework varies from the other variation Takagi-Sugeno fuzzy inference framework and has the advantage of having a likeness to the human subjective framework.The fuzzy neuro technique is a backbone of soft computing. Soft computing techniques defined as the combination of neuro-fuzzy computing and derivative-free optimization. The characteristics of soft computing are human expertise, biologically inspired computing models, new optimization techniques, and numerical computation.

3. Data Analysis and Results

The data are usually partitioned into at least two subsets when developing an artificial neural network, such as training and testing data. The training data should be selected before executing the artificial neural network program. In the past few years,several software reliability models have been analyzed, designed, and evaluated. Soft computing plays an essential role in the recent advancements in software reliability growth models. In this paper, we are going to implement hybrid neuro-fuzzy techniques on two different datasets obtained from the eclipse bug prediction dataset [11]. The bug prediction dataset is a collection of software system models and metrics and their histories. One dataset is for server Apache Lucene which contains 600 failure numbers of class data with eight attributes. Another dataset of MyLyn application software with 18 attributes and 1856 classes of failure. We have taken our case study with two attributes, which are several bugs and several non-trivial bugs. Furthermore, several bugs mean less reliability of software, and less number of bugs means higher reliability of software. Hence several bugs decide the reliability of software. The structure of ANFIS for predicting reliability over this dataset is as shown in Table 1.

TABLE 1. Structure of ANFIS

No. Of Nodes	No. Of Linear Parameters	No. Of Non-Linear Parameters	No. Of Training Data Pairs
16	3	9	600

The number of hidden layers is two, and the number of neurons is 16. Further, with the help of Eqs (2.1) to 4.2, we calculate the MSE and VAF using the approaches neural with MLP algorithm, Neuro-fuzzy in MATLAB, and statistical measurement using a model of Song, Chang, and Pham [12]. Fig 1 shows the training error graph and comparison of predicted and previous data. Fig 1 shows the variances between the Number of bugs and predicted bugs for reliability prediction of two datasets MyLyn and Apache Lucene. Fig 2 shows the training error with the MyLyn dataset in ANFIS structure. The results of the proposed methods are compared with the method proposed by Song, Chang, and Pham statistical model. The combined results with different models are shown in Table2. The graphical representation of the results is shown in Fig3.

Variance Accounted For (VAF): VAF is used to determine the correctness of the model. To check the validity of the approach and performance of model VAF is calculated as follows in Table3.

There have been numerous works done in the field of software reliability prediction using hybrid soft computing techniques. In this work, authors have used the neuro-fuzzy technique for software reliability prediction in the MATLAB environment. The results of the mean squared error of different modelsmentioned

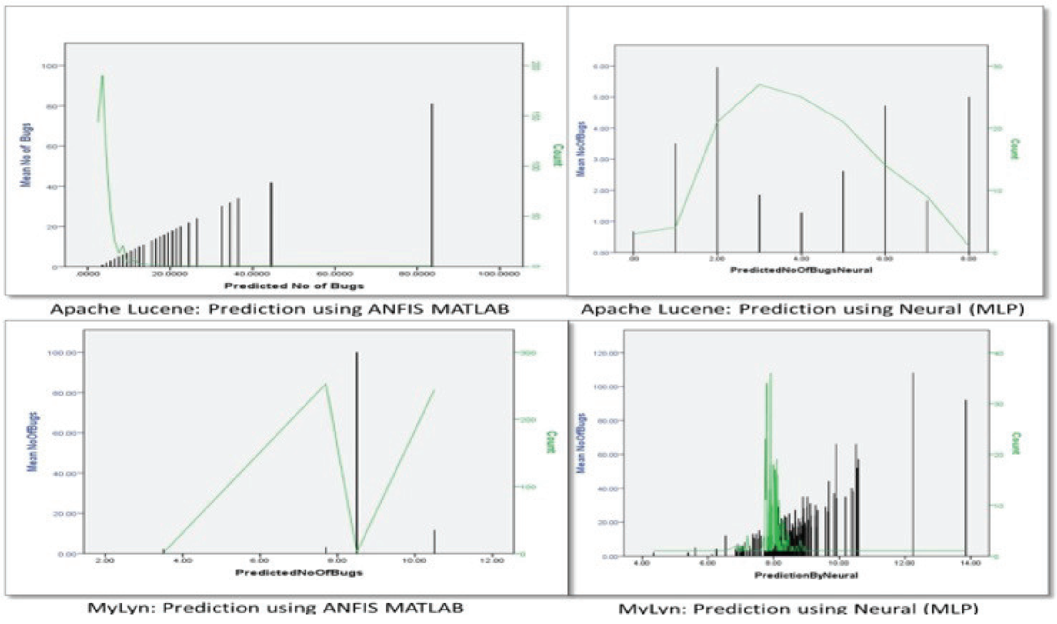
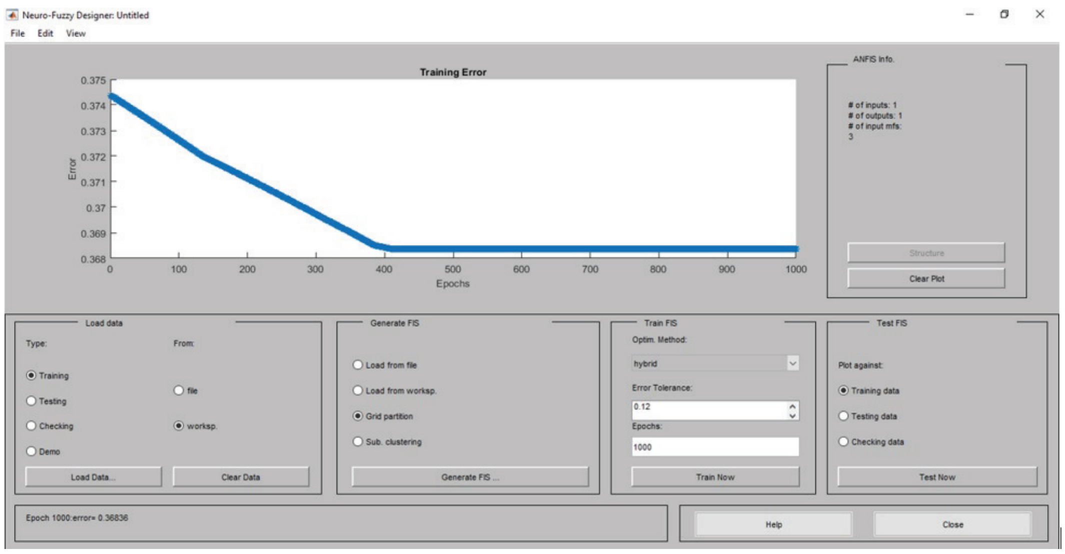


FIGURE 1. Graphs Between Mean No of Bugs and Predicted No of Bugs Using Different Models



ANFIS Info.
# of inputs: 1
# of outputs: 1
# of input mfs:
3

Load data
Type: [✓ Training] Testing Checking Demo
From: file [✓ worksp.]

Generate FIS
Load from file
Load from worksp.
[✓ Grid partition]
Sub. clustering

Train FIS
Optim. Method: hybrid
Error Tolerance: 0.12
Epochs: 1000

Test FIS
Plot against: [✓ Training data] Testing data Checking data

Epoch 1000:error= 0.36836

FIGURE 2. Training Error in ANFIS Structure

in Table 1, which shows the comparison between proposed models with the neural network technique and neuro-fuzzy technique with the results of Song, Chang, and Pham model is proved to be better than the later one. It is clear from this comparison that our proposed model gives better results than Song, Chang, and Pham model while implementing on two real-time datasets. It is also clear from the results of our proposed model that hybrid technique (neuro-fuzzy) provides us very less error rate in comparison to another neural network model.

TABLE 2. Mean Squared Error of Different Models

Case Number	Neural Network	Neuro-fuzzy	Song, Chang and Pham Model
MyLyn	0.8583	0.0929	2.58
Apache	2.596	0.1895	1.25

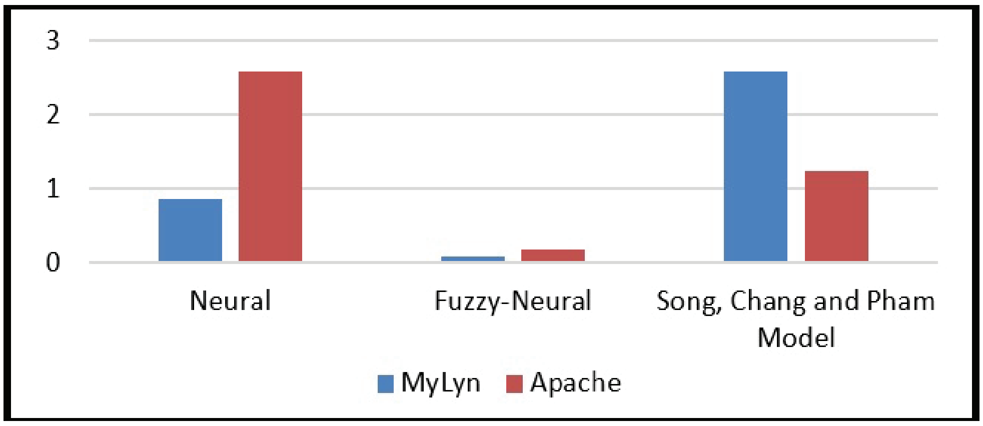


FIGURE 3. Graphical Representation of MSE by different methods

Table 3 shows the different evaluation criteria for bothmodels (Neuro-Fuzzy and Neural Network)in both cases(MyLyn and Apache Lucene). The table shows that the obtained VAF for the Neuro-Fuzzy model is pointing at the correctness on the Apache Lucene dataset. Other evaluation criteria such as NRMSE, RMSE, and MSE also show that the error rate is lower for the Apache Lucene dataset. This proves that our proposed neuro-fuzzy model is more appropriate for big and variant datasets such as Apache Lucene.

4. DISCUSSION

Predicting software reliability using soft computing techniques is a challenging task and also a growing research area. Hybrid methods of software reliability

TABLE 3. Different evaluation criteria for the different model on cases

Case Number	Testing VAF	RMSE	NRMSE	MSE
MyLyn(Neuro-fuzzy)	67.12	0.046925	0.004469	0.0929
Apache Lucene (Neuro-Fuzzy)	85.1	0.103001	0.001272	0.1895
Apache Lucene(Neural)	15.19	3.1801	2.9057	2.596
MyLyn(Neural)	45.74	0.274109	0.013949	0.8583

prediction have been given by several authors such as bio-inspired hybrid soft computing technique was proposed by Chander Diwaker et al. in 2018 [6]. The study conducted in this paper shows that the previously proposed approaches have some shortcomings, such as old database based predictions, which is not applicable in the current scenario of sophisticated software.

The main challenges that are faced by various researchers in estimating and predicting software reliability are associated with the new datasets and identifying relevant attributes from it. In this research, a dataset of server and application software has been taken, and two attributes of this data have been identified. Based on these attributes, the reliability of these datasets has been predicted for the future. Our proposed study has emphasized hybrid technique approaches of soft computing. In this paper, we have proved the betterment of our study by proposed work and Table 1, which is a mean squared error rate between the different models. After analyzing the results,the hybrid method of neuro-fuzzy seems to be a better method than the neural method by the MLP algorithm.In the future, this work can be extended as follows:

- Further predicting failure for software to improve the software reliability can extend this work to the other soft computing techniques to give a better estimation of the software reliability at different stages of the Software Development process.
- Prediction and sharpness of the Fuzzy Rule Generation for the Fuzzy Inference System can further be improved with decision-making systems.
- Decision-making methods, such as the Analytic hierarchy process with a hybrid method of soft computing approaches, may give better results.

5. Conclusion

Software reliability is not a hundred percent achievable in the real world. However, we could estimate it through the quality of the software. In this paper, we have analyzed the various systematic existing approaches to prove software reliability using soft computing models. We emphasize on some soft computing model to prove the software reliability. The computing task would serve as a reference to both old and new includes neural network, Fuzzy logic, and neural fuzzy and genetic algorithm. From our result analysis, we observed that the handling of software datasets in a variety of application models was proved that the computation results using a hybrid method of neuro-fuzzy are much better than the other computing models. Finally this model supports our understanding of current trends and guides the research flows. This study provides the best indication of the prediction strength of developed neuro-fuzzy model for accessing the software reliability.

References

[1] K. Sahu, R.K. Srivastava: *Revisiting Software Reliability, Data Management, Analytics and Innovation*, Springer, **254** (2019), 221-235.

[2] A. Verma: *Worst of the Worst⁻ The Biggest Software Fails in Recent Memory*, (2018) Available online at: https://www.functionize.com/blog/worst-of-the-worst-the-biggest-software-fails-in-recent-memory/.

[3] K. Sahu, Rajshree: *Stability: abstract roadmap of security*, American International Journal of Research in Science, Engineering & Mathematics, (2015), 183-186.

[4] S.W.A. Rizvi, V.K. Singh, R.A. Khan: *Fuzzy logic-based software reliability quantification framework: early-stage perspective (FLSRQF)* Procedia Computer Science, **89** (2016), 359-368.

[5] A. Kukkar, R. Mohana, A. Nayyar, J. Kim, B.G. Kang, N. Chilamkurti: *A Novel Deep-Learning-Based Bug Severity Classification Technique Using Convolutional Neural Networks and Random Forest with Boosting*, Sensors, **19**(13) (2019), 2964.

[6] K. Kaswan, S. Choudhary, K. Sharma: *Software Reliability Modeling using Soft Computing Techniques: Critical Review*, International Journal of Information Technology and Computer Science, **7** (2015), 90-101.

[7] A. Nayyar: *Instant Approach to Software Testing: Principles, Applications, Techniques, and Practices*, BPB Publications, 2019.

[8] K. Sahu, R.K. Srivastava: *Needs and Importance of Reliability Prediction: An Industrial Perspective*, Information Sciences Letters, **9**(1) (2020), 33-37.

[9] N. KARUNANITHI, D. WHITLEY, Y.K. MALAIYA:*Using neural networks in reliability prediction*, IEEE Software, **9**(4) (1992), 53-59.

[10] J.S.R. JANG, C.T. SUN, E. MIZUTANI: *Neuro-fuzzy and soft computing a computational approach to learning and machine intelligence* [Book Review], IEEE Transactions on automatic control, **42**(10) (1997), 1482-1484.

[11] *Bug Prediction Dataset- Evaluate Your Bug Prediction Approach on Our Benchmark*, 2015, Available online at http://bug.inf.usi.ch/index.php

[12] K.Y. SONG, I.H. CHANG, H. PHAM: *A software reliability model with a Weibull fault detection rate function subject to operating environments*, Applied Sciences, **7**(10) (2017), art.id.983.

DEPARTMENT OF COMPUTER SCIENCE
DR. SHAKUNTALA MISRA NATIONAL REHABILITATION UNIVERSITY
LUCKNOW, U.P., INDIA-226017
Email address: * kavi9839@gmail.com

DEPARTMENT OF COMPUTER SCIENCE
DR. SHAKUNTALA MISRA NATIONAL REHABILITATION UNIVERSITY
LUCKNOW, U.P., INDIA-226017