*Journal of Advances and Scholarly Researches in Allied Education*
*Vol. 15, Issue No. 12, December-2018, ISSN 2230-7540*

# Improving Software Reliability Prediction Using Various Soft Computing Techniques

**Neeta Tewari[1]* Dr. Alok Kumar Verma[2]**

[1] Research Scholar, Mewar University, Chittorgarh, Rajasthan

[2] Associate Professor, Mewar University, Chittorgarh, Rajasthan

***Abstract – The reliability of software models is measured using the estimation of program errors. Trustworthiness is a real-world phenomenon with many related problems in real-time. A large number of soft computing techniques have been developed to find solutions to problems quickly, accurately, and acceptably but it is very hard to find the one most suitable and that is the one that can be used worldwide. In this paper, we have presented a summary of the software technologies that are possible, and then we have objectively examined the information reliability work performed by the different researchers. Furthermore, we have compared software reliability modeling techniques in software models.***

***Key Words: Software, Software Reliability Models, Soft Computing Techniques***

---------------------------------x--------------------------------

## INTRODUCTION

The great growth in the area of software technology, modern software scope ought to turn out to be more extensive, and hence further software functions need to be achieved to fulfill client requirements by considering software reliability as a dynamic factor in the growth procedure of modern software [1]. Researchers have established diverse growth models to figure out the issues found in software reliability. The prototypes are of two types, non-parameter, and parameter-based. Some multiple parameter techniques are like non-homogeneous Poisson procedure, Bayes process, and stochastic differential equation [2].

Parameter techniques are utilized to measure peripheral parameters by exploiting the valuation of the maximum probability or least squares. Different types of estimation methods give different results for external parameters. These methodologies need conventions recognized in advance to the improvement of pertinent parameter models. Several protocols are reliable in the actual testing process, but others are not reliable in an authentic test scenario [3]. Thus, the conventions that are followed in a test case differ from those in other testing cases depending on the complexity of testing software. Moreover, offering the assumptions which comprise of all testing conditions is not probable. The conventions proposed in parameter techniques are limited, subjective, and have partial testing feature selection to abridge and facilitate modeling. Realistic assumptions and proposing objectives in parameter-based techniques remain as difficult tasks. Other software reliability techniques are preferred to determine software reliability, similar to Statistical Time Series prediction approaches, Bayesian Network technology, and function approximation [4].

Nevertheless, these techniques possess their difficulties along with concerning software reliability valuation. For example, BN approaches need more calculation, complex analysis, fault data, and particular fault data to use distinct possibilities. STS prediction methods emphasize the time factor and do not deliberate the impact of other peripheral elements [5]. Consequently, the state of detected faults in a test demonstrates a great variation when the time changes. Statistical Time Series prediction technique improves enactment in the short-term than a long-term prediction [6].

The quality evaluation is an important task in software production and design since quality is much important for software, with the constant growth of soft project discipline, and software design. The valuation process of software quality highlights is a more important function too. Though, it is commencing the application fields different in heterogeneities of a software system [7]. They chose an evaluation index system of the software for a very great prejudice appearance. Because of the difficulty factors in the evaluation of software quality, the factor essentials to be thinning steadily make the quality evaluation process of the software demonstrate several experience trends in coexisting with objectivity, index quantizing, subjectivity, multilevel. As of these reasons, the application of fuzzy and the general technique of judging for the evaluation process of software quality is increasing. Though, the single fuzzy and broad technologies which are used to judge unavoidably present a lack of shortcomings and the standard deviation of the tolerance [8].

## SOFTWARE RELIABILITY ANALYSIS

Software Reliability is interpreted as the prospect of failure-free operation for a quantified period in a specified environment. Due to the increase in the complexity of software, it was required that serious research was done in this area. Currently, the use of computers has increased manifold. The software turns out to be an essential part of commercial operations, many industrial, and the military. Supercomputers and Microcomputers may find programs comprising millions of programming codes. The importance of software has increased in safety-critical systems which led to the fact that it is an essential research area. Software engineering has made a lot of progress during the last century [9]. Still, it lacks a scientific method to measure them. Software reliability measure is a tool used to evaluate software engineering principles. Hardware technologies have made remarkable progress as compared to predicting reliability in software.

## RESEARCH METHODOLOGY

The research work in this paper involves two proposed algorithms for software reliability prediction:

**Algorithm I:**

The proposed Fuzzy Greedy Recurrent Neural Network algorithm is designed for the prediction of software reliability by using soft computing methods. It is mainly focused on analyzing software of all data types. It performs with high accuracy. The study prefers a model of deep learning which is based on the Recurrent Neural Network to forecast the total number of faults of the software besides assessing the software reliability. Investigational outcomes showed that the suggested model has improved the performance of prediction associated with the further parameter and neural network models. The deep neural network model can capture the steady in addition to precise features of software faults by using a Fuzzy Greedy Recurrent Neural Network algorithm. It gives better results when compared with existing methods. JAVA/J2EE tool is considered for the process of implementing the proposed algorithm.

## PROPOSED FUZZY GREEDY RECURRENT NEURAL NETWORK ALGORITHM

The proposal aims to predict software malfunctions using the deep neural network model, using the Fuzzy Greedy Recurrent Neural Network algorithm to capture stable and accurate properties. The Fuzzy Greedy Recurrent Neural Network algorithm offers a world-wide optimum approach. Fuzzy Greedy Recurrent Neural Network is used to accurately analyze the database data and produce improved predictive outcomes for device reliability. Fault identification data and correction data may be done as training data intended for a profound neural network using Fuzzy Greedy Recurrent Neural Network algorithms. In contrast to other conventional approaches, this approach has greater power.

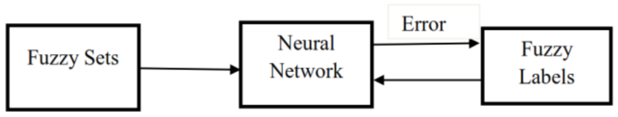

**Figure 1.1 The Neural Network Implementing the Fuzzy Classifier**

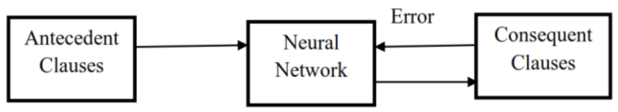

**Figure 1.2 A Neural Network applying Fuzzy Logic**

The proposed Fuzzy Greedy Recurrent Neural Network method uses four types of software metrics like Cyclomatic Complexity, Essential Complexity, line of control, and Design Complexity as prescribed by McCabe.

### Mathematical Modeling of Fuzzy Greedy Recurrent Neural Network

(i) An initial phase of the proposed method is to select the attributes as a pre-processing method. Let attribute A has v different values. Let $s_{ij}$ be some samples of class $C_i$ in a subset $S_j$. It contains those samples in S that have a value $a_j$ of A. The entropy, or expected information based on the segregating into subsets by A, is specified by,

$$E(A) = -\sum_{i=1}^{m} I(S)\frac{s_{1i}+s_{2i}+....+s_{mi}}{s} \qquad (1.1)$$

The above value signifies the information generated by separating the training data set S into v partitions corresponding to v outcomes of a test on attribute A.

(ii) The fuzzy triangular formula is represented as:

$$f(x,a,b,c) = \begin{cases} 0, x \le a \\ \frac{x-a}{b-a}, a \le x \le b \\ \frac{c-x}{c-b}, b \le x \le c \\ 0, c \le x \end{cases} \qquad (1.2)$$

$$f(x,a,b,c) = max\left(min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),0\right) \qquad (1.3)$$

here, a, b, c three corners of the triangle, *X* is assumed to be a universal set, and *x* is an element in *X*. The *A* set is a fuzzy set of *X*, and the value of $\mu_A(x)$ is the membership function concerning each *x* in [0, 1].

For an internal variable $h_i$, the subsequent sigmoid membership function is utilized:

$$M_S(h_i) = \frac{1}{1+exp\{-h_i\}} \qquad (1.4)$$

where, $M_s$ is the sigmoid membership function. The respective internal variable possesses a single equivalent fuzzy set.

(iii) The greedy evaluation function denotes a degree of priority for putting the corresponding element 'x' into the solution under construction. An alternative approach is the set X as a fuzzy set with a well-defined membership function $\mu(x)$, the form of which is given by,

$$\mu(x) = \frac{1}{1+\lambda^2\left[\left(\frac{1-\lambda}{\lambda}\right)x-\theta\right]^2} \qquad (1.5)$$

$$greedy_{dsize} = \sum_{j=0}^{dsize}(dset(j) > dset(j+1)) \qquad (1.6)$$

The evaluation function in Equation (1.6) has the following properties: μ (λθ/ (1 – λ)) = 1 and 0 < μ(x) < 1 for all x ≠ λθ / (1 – λ). This implies that x closest numerically to λθ/ (1 – λ) should be given higher priority. Let $x_{min}$ and $x_{max}$ be the smallest and the largest values of x, respectively. We define a small enough value $\lambda_{min}$=$x_{min}$ / ($x_{min}$+ θ) for θ > 0 and any λ ≤ $\lambda_{min}$ for which inequality x ≠ $\lambda_{min}$θ/ (1 – $\lambda_{min}$) holds. It also defines a big enough value $\lambda_{max}$ = $x_{max}$ /($x_{max}$+ θ) for θ > 0 and any λ ≥ $\lambda_{max}$ for which inequality x ≤ $\lambda_{max}$θ/(1 – $\lambda_{max}$) holds.

(iv) The recurrent neural network model has two external input variables $x_1$, $x_2$ with a single output y. Consequently, a recurrent neural network has two nodes in layer 1 and one node in layer 5. This method is described layer by layer to understand the mathematical function of each node clearly.

a) In layer 1, a particular node agrees to one input variable besides directly communicates input values to the next layer, thus needs no computation.

b) In layer 2, a particular node relates to a fuzzy set and computes a membership value.

$$S_{testsetsize} = \sum_{i=0}^{testsetsize}(testset(i) < i) \qquad (1.7)$$

$$\Phi(z) = \frac{1}{1+e^{-1}} \qquad (1.8)$$

When using the Fuzzy Greedy Recurrent Neural Network, a particular internal variable possesses a single corresponding fuzzy set. Links in layer two are all set to unity.

c) In layer 3, every node denotes a fuzzy logic rule then achieves antecedent matching of this rule through the subsequent operation of AND,

$$\mu_i = M_S(h_i).\prod_{j=1}^{n} M_G^i(x_j) = \frac{1}{1+exp\{-h_i\}}\prod_{j=1}^{n} exp\left\{-\left(\frac{x_j-m_{ij}}{\sigma_{ij}^2}\right)^2\right\} \qquad (1.9)$$

Where n is the number of external inputs. The link weights are all set to unity.

d) In layer 4, the particular node corresponds to one context node and achieves a de-fuzzification process for internal variables h. The simple weighted sum is calculated in each node,

$$h_i = \sum_{k=1}^{r} \mu_k w_{ik} \qquad (1.10)$$

e) In layer 5, each node corresponds to one output variable and performs weighted average operations for output y. The mathematical function is,

$$y = \frac{\sum_{i=1}^{r}\mu_i b_i}{\sum_{i=1}^{r}\mu_i} \qquad (1.11)$$

where, $b_i$ is a fuzzy singleton value working as the subsequent part of output variable y.

**Algorithm II:**

Naïve Bayes Classifier is a classifier based on Bayes theorem utilized in the prediction of Software Fault. It resolves many complications similar to spam classification in the case of electronic mail by helping in the forecasting of spams, medical diagnosis (specified a list of symptoms, forecast whether the patient has cancer or not), etc. This technique can be utilized to forecast faulty and non-faulty units. It is considered to provide better accuracy in comparison with other classifiers. It provides computational efficiency and is simple to build, as no learning stage is necessary. An advantage of the Naive Bayes classifier is that it only needs a small amount of training information to evaluate the parameters required for classification. For the reason that the independent variable is supposed, merely the variances of the variables for the respective class required to be defined and not the whole covariance matrix.

## PROPOSED FUZZY ATTRIBUTE CLUSTER NET BAYES ALGORITHM

Fuzzy Naive Bayes method computes conditional class probabilities and then predict the most probable class of a vector of training data $X = \{X_1, X_2, \ldots, X_n\}$, according to sample data D. The functions in Naïve Bayes are computed as follows:

$$P(\frac{w_i}{X}) = [P(\frac{X}{w_i})P(w_i)\mu_i(X)]/P(X) \qquad (1.12)$$

Where P is the probability distribution, X be the no. of input training data sets and w be the weightage of the classifiers. This Naive Bayes method assumes conditionally independent among the events in X. It modifies the equation (4.3) to:

$$P(\frac{w_i}{X}) = (1/S)P(w_i)\prod_{k=1}^{n}[P(\frac{X_k}{w_i})\mu_i(X)] \qquad (1.13)$$

Then, the classification rule for Fuzzy Naive Bayes is made by:

$$X \in w_i \; if: P\left(\frac{w_i}{X}\right) > P\left(\frac{w_j}{X}\right), \; for \quad all\; i \neq j \quad and\; i, j \in \Omega \qquad (1.14)$$

where $P(\frac{w_i}{X}) = P(\frac{w_i}{X_l} = A_{li}, \ldots, X_n = A_{ni})$ by equation (4.4) and A be the attributes and i and j be the instances.

The proposed algorithm consists of a combination classifier which combines Fuzzy Set Attributes and Cluster Theory and a Naive Bayes classifier, termed as Fuzzy Attribute Cluster Naive Bayes classifier,

$$FNBclassify(a) = \underset{c \in C}{argmax}\, P(c) \textstyle\sum_{x_1 \in X_1} P(x_1|c)\mu_{x_1} \ldots \sum_{x_n \in X_n} P(x_n|c)\mu_{x_n} \qquad (1.15)$$

where, $\mu_{x_i} \in [0,1]$ represents a membership function or degree of truth of attribute, $x_i \in X_i$ in a new example a. To be conservative, all degrees of truth must be standardized in the existing variable assignation, in this case $\sum_{x_i \in X_i} \mu_{x_i} = 1$. The probabilities for equation (4.6) can be computed as below,

$$P(C = c) = \frac{(\sum_{e \in L} \mu_c^e) + 1}{|L| + |D(C)|} \qquad (1.16)$$

$$P(X_i \in x_i) = \frac{(\sum_{e \in L} \mu_{x_i}^e) + 1}{|L| + |D(X_i)|} \qquad (1.17)$$

$$P(X_i = x_i | C = c) = \frac{(\sum_{e \in L} \mu_{x_i}^e \mu_c^e) + 1}{(\sum_{e \in L} \mu_c^e) + |D(X_i)|} \qquad (1.18)$$

where L is the training set involving of all illustrationse $= \{X_1 = x_1, \ldots, X_n = x_n, C = c\}$, $\mu_c^e \in [0,1]$ signifies the degree of truth of c ∈ C in an example e ∈ L, and $\mu_{x_i}^e \in [0,1]$ is the membership of attribute $x_i \in X_i$ in such an example. As mentioned before, all degrees of truth must be normalized such that $\sum_{c \in C} \mu_c^e = 1$ and $\sum_{x_i \in X_i} \mu_{x_i}^e = 1$ Note that Laplace-correction is applied to compute the probabilities.

## RESULT AND DISCUSSION

### Algorithm I:

The accuracy performance of the proposed Fuzzy Greedy Recurrent Neural Network on datasets PC4, CM1, KC1, and PC1 are compared with the various existing algorithms which are shown in Table 1.1. These values indicate better performance of the proposed Fuzzy Greedy Recurrent Neural Network in terms of accuracy over others.

**Table 1.1 Comparison of Accuracy results of Proposed Fuzzy Greedy Recurrent Neural Network with other Methods over various datasets**

| Algorithm \ Dataset | CM1 | KC1 | PC1 | PC4 |
|---|---|---|---|---|
| RIPPER | 84.52 | 82.91 | 92.07 | 93.07 |
| C4.5 | 85.12 | 81.34 | 88.39 | 82.35 |
| CBA2 | 80.36 | 83.71 | 91.78 | 84.72 |
| NB2 | 86.74 | 82.86 | 89.21 | 84.95 |
| BBN | 76.83 | 75.99 | 90.44 | 76.99 |
| RBF | 89.91 | 84.81 | 92.84 | 85.21 |
| KNN | 83.27 | 83.99 | 91.82 | 84.12 |
| CFS+NB | 84.84 | 82.46 | 89 | 83.45 |
| RF | 85.75 | 85.06 | 91.43 | 86.23 |
| NB1 | 82.26 | 82.44 | 88.27 | 84.46 |
| J48 | 85.46 | 84.16 | 90.9 | 88.24 |
| SVM+EA | 90.16 | 85.4 | 93.24 | 87.547 |
| Proposed Model - ***Fuzzy Greedy Recurrent Neural Network*** | 90.1606 | 85.3959 | 93.688 | 89.378 |

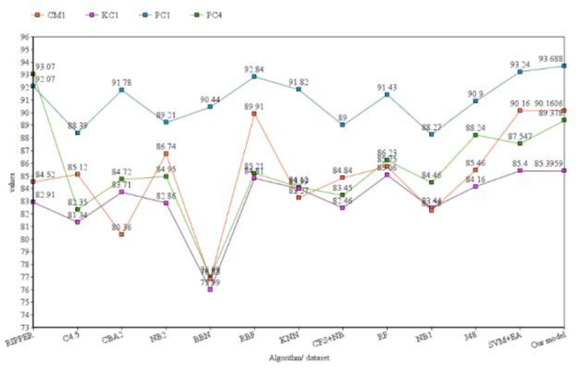

**Figure 1.3 Performance Comparison Graph of Proposed Fuzzy Greedy Recurrent Neural Network with other Methods over various datasets**

Table 1.1 and Figure 1.3 shows that the dataset such as CM1, KC1, and PC1 is used to validate the proposed method. The CM1 accuracy value for the proposed method is 90.1606, KC1 value is measured as 85.3959, and PC1 dataset as 93.688. These values indicate the accuracy of the Fuzzy Greedy Recurrent Neural Network better when compared with the existing method Support Vector Machine and Evolutionary Algorithms (SVM+EA).

**Algorithm II:**

Similarly, another proposed Fuzzy Attribute Cluster Net Bayes algorithm has been made to work on datasets PC4, CM1, KC1, and PC1. The Performance Metrics of Fuzzy Attribute Cluster Net Bayes algorithm as presented by the simulation software is given below:

**Table 1.2 Performance Metrics of proposed Fuzzy Attribute Cluster Net Bayes Algorithm**

| Metrics | TP rate | FP rate | Precision | Recall | F-measure | ROC area | Class |
|---|---|---|---|---|---|---|---|
| Correct | 0.989 | 0.015 | 0.903 | 0.989 | 0.944 | 0.9928 | Cluster 0 |
| Incorrect | 0.985 | 0.011 | 0.998 | 0.985 | 0.992 | 0.9928 | Cluster 1 |
| Weighted Average | 0.986 | 0.012 | 0.987 | 0.986 | 0.986 | 0.9928 | |

The performance comparison of the various clustering algorithms with the proposed Fuzzy Attribute Cluster Net Bayes algorithm is listed in Table 1.3.

**Table 1.3 Comparison of Accuracy results of Proposed Fuzzy Attribute Cluster Net Bayes Algorithm with other Clustering Methods**

| Evaluation Parameter | Fuzzy C-means Clustering | Adaptive Neuro-fuzzy Technique | Proposed Fuzzy Attribute Cluster Net Bayes Algorithm |
|---|---|---|---|
| Accuracy | 75 | 87 | 99.28 |
| Net Reliability | 60.07 | 47.20 | 35.23 |
| Mean Absolute Error (MAE) | 0.25 | 0.13 | 0.16 |
| Root Mean Squared Error (RMSE) | 0.0833 | 0.0194 | 1.089 |

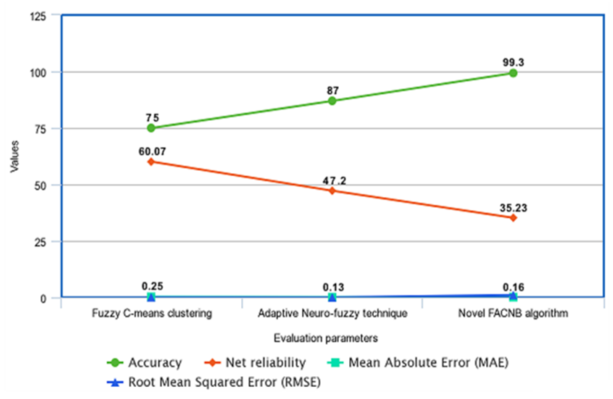

**Figure 1.4 Performance Comparison Graph of Proposed Fuzzy Attribute Cluster Net Bayes Algorithm with other Methods over Evaluated Parameters**

## CONCLUSION

Finally, a Comparative of calculation results of our work on both of the proposed methods, Fuzzy Greedy Recurrent Neural Network and Fuzzy Attribute Cluster Net Bayes Algorithm as furnished in Table 1.4 shows that the Fuzzy Attribute Cluster Net Bayes Algorithm is producing a superior performance to Fuzzy Greedy Recurrent Neural Network, hence superseding the performance of all the existing methods upon whom Fuzzy Greedy Recurrent Neural Network proved to be better.

**Table 1.4 Comparison of Fuzzy Greedy Recurrent Neural Network and Fuzzy Attribute Cluster Net Bayes Algorithm for PC4 Dataset**

| Calculation Results | Fuzzy Attribute Cluster Net Bayes Algorithm (%) | Fuzzy Greedy Recurrent Neural Network (%) |
|---|---|---|
| Correctly Classified Instances | 98.56 | 89.37 |
| Incorrectly Classified Instances | 1.44 | 10.63 |
| Kappa Statistic | 93.55 | 33.09 |
| Mean Absolute Error | 0.16 | 15.08 |
| Root Mean Square Error | 1.089 | 28.68 |
| Relative Absolute Error | 7.53 | 70.68 |
| Root Relative Squared Error | 33.23 | 87.61 |

## REFERENCES

1. Kapur, P. K., Goswami, D. N., & Gupta, A. (2004). *A software reliability growth model with testing effort dependent learning function for distributed systems*. International Journal of Reliability, Quality and Safety Engineering, Vol. 11(04), pp. 365-377.

2. Hossain, S. A., & Dahiya, R. C. (1993). *Estimating the parameters of a non-homogeneous Poisson-process model for software reliability*. IEEE Transactions on Reliability, Vol. 42(4), pp. 604-612.

3. Bertolino, A. (2007, May). *Software testing research: Achievements, challenges, dreams.* In Proceeding of IEEE Future of Software Engineering (FOSE 2007), pp. 85-103. IEEE Computer Society, Washington, DC, USA

4. Zhang. C, Wang. J. (2016). *Software reliability prediction using a deep learning model based on the RNN encoder-decoder.* Reliability Engineering & System Safety, Vol. 170, pp. 73-82

5. Yontay, P. (2016). *A Bayesian Network Approach to Early Reliability Assessment of Complex Systems* (Doctoral dissertation, Arizona State University).

6. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2015). *Statistical and Machine Learning forecasting methods: Concerns and ways forward.* PloS one, 13(3), pp. e0194889.

7. Basili, V. R., Briand, L. C., & Melo, W. L. (1996). *A validation of object-oriented design metrics as quality indicators.* IEEE Transactions on Software Engineering, Vol. 22(10), pp. 751-761.

8. Song, Y. & Yan, J (2009). *A Software Quality Comprehensive Assessment*

*Algorithm with the Fuzzy Reference Pattern.* Second International Symposium on Information Science and Engineering.

9. Nagar, P., & Thankachan, B. (2012). *Application of goel-Okumoto model in software reliability measurement.* International Journal of Computer Applications, Special Issue on Issues and Challenges in Networking, Intelligence, and Computing Technologies ICNICT, Vol. 5, pp. 1-3.

10. Sadhankar, D. & Sasankar, A. (2013). *An Overview and Classification of Software Reliability Models.* International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Vol. 3(1), pp. 1960-1966.

**Corresponding Author**

**Neeta Tewari***

Research Scholar, Mewar University, Chittorgarh, Rajasthan