


A Neuro-Fuzzy Hybridized Approach for Software Reliability Prediction

Ajay Kumar

(Department of Computer Science and Engineering, Ajay Kumar Garg Engineering College,
Ghaziabad, Uttar Pradesh, India,

 <https://orcid.org/0000-0003-0126-7172>, ajaygarg100@gmail.com)

Abstract: *Context:* Reliability prediction is critical for software engineers in the current challenging scenario of increased demand for high-quality software. Even though various software reliability prediction models have been established so far, there is always a need for a more accurate model in today's competitive environment for producing high-quality software. *Objective:* This paper proposes a neuro-fuzzy hybridized method by integrating self-organized-map (SOM) and fuzzy time series (FTS) forecasting for the reliability prediction of a software system. *Methodology:* In the proposed approach, a well-known supervised clustering algorithm SOM is incorporated with FTS forecasting for developing a hybrid model for software reliability prediction. To validate the proposed approach, an experimental study is done by applying proposed neuro-fuzzy method on a software failure dataset. In addition, a comparative study was conducted for evaluating the performance of the proposed method by comparing it with some of the existing FTS models. *Results:* Experimental outcomes show that the proposed approach performs better than the existing FTS models. *Conclusion:* The results show that the proposed approach can be used efficiently in the software industry for software reliability prediction.

Key Words: Software Reliability; Self-Organized-Map (SOM); Fuzzy-Time-Series (FTS)

Category: D.2, D.2.4, D.2.9

DOI: 10.3897/jucs.80537

1 Introduction

In day-to-day life, software plays an important role in many applications such as home appliances, industrial controls, hospital health care units, nuclear reactor plants, aircraft, air traffic control, shopping, and many more. To increase their effectiveness and efficiency, many governments and commercial organizations depend on software. In this growing scenario of software technology, software practitioners are concerned about the good quality of the software as it is highly correlated with end-user satisfaction. Software failure may lead to economic loss and customer dissatisfaction for the organizations. So, in this scenario, software reliability is a key factor for measuring the overall quality of any software system.

Software reliability is defined as the likelihood of software execution in a given environment without failure for a certain period of time [Iannino and Musa 1990]. For example, suppose the software has a reliability of 98 percent over ten elapsed processing hours (execution time). In that case, it means that the software is likely to operate 98 times without any failure out of 100 times. Here, software failure means that it is producing an incorrect output corresponding to a particular input received as per

specification [Pham 2006]. Thus, software failure occurs due to the presence of software fault, which can occur for many reasons such as incorrect code, data handling error, computation error, execution flow failure, data input error, and many more.

The demand for good and reliable software is growing every day in today's competitive economy. For developing reliable software, the prediction of reliability is an important step. Software reliability prediction aids the software industry in developing good reliable software systems within budget constraints and given time. As a result, researchers have always been interested in building an efficient and accurate model for software reliability prediction. Over the last few decades, various software reliability growth models (SRGMs) have been developed and reported in the literature.

In the earlier research work done by various researchers to develop software reliability prediction models, the emphasis was to represent the failure phenomenon during the testing process as a non-homogeneous poisson process (NHPP) [Pham 2006]. This resulted in extreme difficulty to validate underlying assumptions about software failure phenomenon and led to the introduction of non-parametric non-linear tools such as artificial neural network (ANN) [Bisi and Goyal 2016], which could generate a software reliability model from historical data of observed failures. However, ANNs have the problem of getting stuck in a local minimum and slow convergence, which led software reliability researchers to explore ensemble and hybrid techniques for estimating software reliability.

Current initiatives in software reliability research are directed towards more accurate hybrid techniques based on integrating ANNs with other computational intelligence techniques [Bisi and Goyal 2016, Bisi and Goyal 2015]. Based on self-organized maps (SOM) and fuzzy time series (FTS), this paper proposes a neuro-fuzzy hybridized method referred to as SOMFTS to predict a software system's reliability based on its failure history.

This paper's remainder is as follows: The research on software reliability prediction is discussed in Section 2. Section 3 describes the dataset used in this paper. The proposed hybrid SOMFTS technique for software reliability prediction is described in Section 4. Section 5 describes the performance measures that were used to evaluate the proposed method SOMFTS. Finally, section 6 summarizes the experimental findings, section 7 presents discussion of results while section 8 concludes the paper.

2 Related Work

The work involved in establishing software reliability prediction models is described in this section. Various researchers have been working on constructing software reliability prediction models for the past two decades. However, in recent years most researchers emphasized deep neural-based techniques for software reliability prediction.

[Karunanithi et al. 1992] proposed the first software reliability growth model (SRGM) based on an artificial neural network (ANN) for predicting the cumulative failure counts. [Cai et al. 2001] developed a model based on a backpropagation neural network for software reliability prediction. They predicted the next step failure time by using successive failure times as an input. [Tian and Noore 2005] developed an

evolutionary model based on multiple-delayed-input-single-output architecture for predicting cumulative software failure time using ANN.

[Kumar and Singh 2012] conducted an empirical study of selected machine learning techniques, including ANN, SVM, Fuzzy Inference System (FIS), Decision Trees, and Cascade Correlation Neural Network (CCNN). They concluded that SVR might be a better choice for software reliability prediction.

[Ramakrishna et al. 2012] proposed software reliability prediction models based on Support Vector Regression (SVR) and Multilayer Perceptron (MLP). Based on an evaluation study, they concluded that MLP performs better than SVR for software reliability prediction. [Benala et al. 2013] proposed a functional link artificial neural network (FLANN) model based on PSO for software effort prediction.

[Amin et al. 2013] proposed a time series forecasting approach using Autoregressive Integrated moving average (ARIMA) for the software reliability prediction. They compared their proposed model with other traditional software reliability growth models. They claimed that the time series-based model predicts the reliability of a software system better than that of conventional models.

[Kewen et al. 2013] proposed an ensemble method by using K-means clustering methods. In their proposed method, authors use the K-means clustering algorithm to select one combination of various generated neural networks and select the best output using the entropy weight method. In their study, a comparison was carried out between optimal individual networks and ensembled networks. Based on results analysis, authors suggest that ensemble networks give better results than those produced by individual networks.

[Jin and Jin 2014] proposed a hybrid model (IEDA-SVR) by integrating SVR with the improved estimation of distributed algorithms. Based on a comparative study with the other four models, the authors conclude that the proposed IEDA-SVR model can be used for software reliability prediction with better efficiency. In their study, [Song and Chang 2014] have used various non-homogeneous poisson process (NHPP) and time series regression curve models. Furthermore, they collected two software failure datasets for the validation of their proposed approach. The first dataset was collected from the test of system T at AT & T Bell laboratory, and the second dataset was collected from NTDS having information about the failure of the real-time multi-computer complex system. Based on result analysis, the authors find that for dataset1 logarithmic approach is best and for dataset2 S-model is best for the prediction of software reliability.

For software reliability prediction, [Roy et al. 2014] proposed a feed-forward and recurrent neural network-based dynamic weighted combination model. Based on experimental results authors concluded that the proposed model could be used as an efficient model for the software reliability prediction.

[Bisi and Goyal 2015] proposed a hybrid model PSO-ANN by incorporating particle swarm optimization (PSO) technique in an artificial neural network to determine the optimal weights of the network. The proposed model PSO-ANN was compared with the other four traditional software reliability growth models, namely the Jelinski Moranda model, Geometric, Musa Okumoto, and Musa Basic. Based on experimental results, the authors concluded that the proposed hybrid performed better than the singular ANN approach on normalized root means square metric. However, it is difficult to determine the optimal architecture of ANNs in terms of the number of layers in the network and the number of neurons in each layer.

[Lou et al. 2016] applied relevance vector machine (RVM) for software reliability prediction. Although the RVM shares functional similarities with SVM, it is a probabilistic model. RVM requires more training time than SVM because of optimization of a non-convex function, but it does not require the use of free parameters. Regardless of the advantages of RVM over SVM, it requires the selection of a number of kernel parameters that vary according to the type of kernel used.

[Bisi and Goyal 2016] have used four versions of hybrid models by integrating ANN and PSO for software reliability prediction. After comparison with Genetic Algorithm (GA) based software reliability models such as GA-based SVR, GA-based MLP, and GA-based M5P, they came to the conclusion that their proposed model produces more accurate software reliability predictions.

[Jabeen et al. 2017] proposed a hybrid model by combining Jelinski Moranda (JM) model with Grey Model (GM). They have used the advantages of both models JM and GM to produce better results than the results produced by individual models. In another study, [Roy et al. 2017] have used the integration of Particle Swarm Optimization (PSO) and artificial neural network (ANN). Based on experimental results taking Average Error (AE), Mean Square Error (MSE) as evaluation criteria, the authors concluded that their proposed model is better than existing neural network-based models to predict software reliability.

A hybrid model was proposed by [Mallikharajuna and Kodali 2017] based on modified artificial bee colony (MABC) optimization techniques and modified cuckoo search (MCS). In a study, [Jaiswal and Malhotra 2018] evaluate several machine learning techniques used for software reliability prediction such as Feed forward backpropagation neural network, Artificial neural fuzzy inference system, SVR, general regression neural network, MLP, cascading forward BPNN, Bagging, Instance-based learning, MLR, M5P, M5Rules and reduced error pruning tree. They concluded that ANFIS outperformed all other models used in their study. Further, they suggest combining various machine learning techniques to get better performance.

[Ma et al. 2018] proposed a software reliability prediction model based on support vector regression (SVR). They compared their proposed model with other conventional models and found that the SVR model performed better than the conventional models for software reliability prediction.

[Jabeen et al. 2019] developed a software reliability prediction model based on the Grey-Markov chain concept. The proposed model was compared with two well-known models M-J-M and GM (1,1). Based on results obtained of performance measures used in their study, authors conclude that the proposed model can be used as an efficient model for the software reliability prediction.

[Roy et al. 2019] proposed a software reliability prediction model by incorporating neighborhood particle swarm optimization (PSO) in an artificial neural network (ANN) to determine the optimal weights of the network. They concluded that the neighborhood PSO-ANN approach performed better than the standard PSO-ANN approach and singular ANN approach on the average error performance metric.

For enhancing the prediction accuracy of existing software reliability growth models (SRGMs), [Jabeen et al. 2019] proposed a high precision error iterative analysis method (HPEIAM). They combined the residual errors obtained from the estimated results of SRGMs with the artificial neural network sign estimator for enhancing the prediction accuracy of SRGMs. After applying HPEIAM on various SRGMs (GO, J-M,

Littlewood, Jinyong-GO, and Musa), they concluded that HPEIAM enhanced the performance of compared traditional SRGMs models.

[Li et al. 2019] proposed a hybrid model based on artificial bee colony (ABC) and particle swarm optimization (PSO). Their proposed model constructs a new fitness function for the parameter estimation of existing software reliability growth models such as the GO model. They have used five classic sets of software failure data for parameter estimation of the GO model and concluded that the ABC-PSO model was more capable for parameter estimation and reliability prediction than the traditional SRGMs.

[Behera et al. 2019] proposed a hybrid technique based on chemical reaction optimization (CRO) and FLANN. They concluded that their proposed hybrid model was performed better than conventional models for software reliability prediction. In another study, [Pandey et al. 2020] proposed a hybrid model by combining ensemble learning (EL) and deep representation (DR) for software reliability prediction.

[Juneja 2020] proposed a neuro-fuzzy-based model for enhancing the reliability prediction of software systems. A Pareto-distribution ant colony optimization (PD-ACO) hybrid technique was proposed by [Sudharson 2020] for software reliability prediction during the testing phase. [Kumaresan and Ganeshkumar 2020] proposed a software reliability prediction model based using a time series forecasting approach ARIMA. The authors compared the proposed model with other existing time series-based models to show the better efficiency of their proposed model for software reliability prediction.

[Zhen et al. 2020] proposed a hybrid model (PSO-WPA) by considering the advantages of particle swarm optimization and wolf pack algorithm. Based on experimental results, the authors concluded that the hybrid approach (WPA-PSO) performs better than the individual method WPA and PSO to predict software reliability. In a recent study, [Behra et al. 2021] proposed a model based on a functional link artificial neural network where the chemical reaction optimization trained the model's parameters.

[Kassaymeh et al. 2021] proposed a hybrid model for software reliability prediction by combining the Salp swarm algorithm (SSA) with backpropagation neural network (BPNN) to determine the optimal network weights. After a comparative study based on various performance measures, they concluded that the hybrid model SSA-BPNN outperformed the BPNN after a comparative study based on various performance measures.

From the related work as described above, it can be observed that most of the researchers emphasized the use of artificial neural networks for predicting software reliability. It is noted that neural network-based models are complex due to the involvement of a large number of layers and neurons. Further, it is also noticeable that there is no silver bullet intelligent technique for predicting the reliability of software systems, and hence it is desirable to explore competitive and alternative techniques. Along with the high prediction capability, robustness, efficiency, and ease of interpretation of intelligent techniques for software reliability prediction are also desired. This motivates us to develop a software reliability prediction method based on hybridizations of different categories of intelligent techniques.

In contrast to neural networks, which are regarded as black boxes and difficult to interpret, fuzzy systems are easily interpretable, which motivates us to apply the fuzzy time series approach. Based on self-organized maps (SOM) and fuzzy time series

(FTS), this paper proposes a neuro-fuzzy hybridized method referred to as SOMFTS to predict a software system's reliability based on its failure history. To the best of the author's knowledge, no previous research has attempted to predict the software reliability of a software system using the hybrid neuro-fuzzy technique SOMFTS.

3 Description of Software Failure Dataset

This study chooses a well-known software failure dataset for the military software system originally collected by John D. Musa in 1979 presented in Table 1.

<i>t</i>	<i>F_t</i>	<i>t</i>	<i>F_t</i>	<i>t</i>	<i>F_t</i>
<i>t</i> ₀	5.768	<i>t</i> ₃₄	10.630	<i>t</i> ₆₈	12.598
<i>t</i> ₁	9.574	<i>t</i> ₃₅	8.333	<i>t</i> ₆₉	12.086
<i>t</i> ₂	9.105	<i>t</i> ₃₆	11.315	<i>t</i> ₇₀	12.277
<i>t</i> ₃	7.966	<i>t</i> ₃₇	9.487	<i>t</i> ₇₁	11.960
<i>t</i> ₄	8.648	<i>t</i> ₃₈	8.139	<i>t</i> ₇₂	12.025
<i>t</i> ₅	9.989	<i>t</i> ₃₉	8.671	<i>t</i> ₇₃	9.287
<i>t</i> ₆	10.196	<i>t</i> ₄₀	6.462	<i>t</i> ₇₄	12.495
<i>t</i> ₇	11.640	<i>t</i> ₄₁	6.462	<i>t</i> ₇₅	14.557
<i>t</i> ₈	11.628	<i>t</i> ₄₂	7.696	<i>t</i> ₇₆	13.328
<i>t</i> ₉	6.492	<i>t</i> ₄₃	4.701	<i>t</i> ₇₇	8.946
<i>t</i> ₁₀	7.901	<i>t</i> ₄₄	10.002	<i>t</i> ₇₈	14.782
<i>t</i> ₁₁	10.268	<i>t</i> ₄₅	11.013	<i>t</i> ₇₉	14.897
<i>t</i> ₁₂	7.684	<i>t</i> ₄₆	10.862	<i>t</i> ₈₀	12.140
<i>t</i> ₁₃	8.891	<i>t</i> ₄₇	9.437	<i>t</i> ₈₁	9.798
<i>t</i> ₁₄	9.293	<i>t</i> ₄₈	6.664	<i>t</i> ₈₂	12.091
<i>t</i> ₁₅	8.350	<i>t</i> ₄₉	9.229	<i>t</i> ₈₃	13.098
<i>t</i> ₁₆	9.043	<i>t</i> ₅₀	8.967	<i>t</i> ₈₄	13.368
<i>t</i> ₁₇	9.603	<i>t</i> ₅₁	10.353	<i>t</i> ₈₅	12.721
<i>t</i> ₁₈	9.374	<i>t</i> ₅₂	10.100	<i>t</i> ₈₆	14.192
<i>t</i> ₁₉	8.587	<i>t</i> ₅₃	12.608	<i>t</i> ₈₇	11.370
<i>t</i> ₂₀	8.788	<i>t</i> ₅₄	7.155	<i>t</i> ₈₈	12.202
<i>t</i> ₂₁	8.779	<i>t</i> ₅₅	10.003	<i>t</i> ₈₉	12.279
<i>t</i> ₂₂	8.047	<i>t</i> ₅₆	9.860	<i>t</i> ₉₀	11.367
<i>t</i> ₂₃	10.846	<i>t</i> ₅₇	7.868	<i>t</i> ₉₁	11.392
<i>t</i> ₂₄	8.742	<i>t</i> ₅₈	10.576	<i>t</i> ₉₂	14.411
<i>t</i> ₂₅	7.544	<i>t</i> ₅₉	10.929	<i>t</i> ₉₃	8.333
<i>t</i> ₂₆	8.594	<i>t</i> ₆₀	10.660	<i>t</i> ₉₄	8.071
<i>t</i> ₂₇	11.040	<i>t</i> ₆₁	12.497	<i>t</i> ₉₅	12.202
<i>t</i> ₂₈	10.120	<i>t</i> ₆₂	11.375	<i>t</i> ₉₆	12.783
<i>t</i> ₂₉	10.179	<i>t</i> ₆₃	11.916	<i>t</i> ₉₇	13.159
<i>t</i> ₃₀	5.894	<i>t</i> ₆₄	9.575	<i>t</i> ₉₈	12.753
<i>t</i> ₃₁	9.546	<i>t</i> ₆₅	10.450	<i>t</i> ₉₉	10.353
<i>t</i> ₃₂	9.620	<i>t</i> ₆₆	10.587	<i>t</i> ₁₀₀	12.490
<i>t</i> ₃₃	10.385	<i>t</i> ₆₇	12.720		

Table 1: Software Failure Dataset

The dataset consists of 101 failures that occurred during the development phase with approximately 1,80,000 lines of code at the execution time of 1035 (CPU Time) [recorded in Amin et al. 2013 and Mohanty et al. 2013]. Each failure is represented in the form of the pair (t, Ft) , where Ft represents time (in seconds) to software failure after t^{th} modification has been done. Despite being old, this dataset has been used in this study because it is very popular in the field of software reliability and has been widely used in previous studies [Li and Malaiya 1993, Kiran and Ravi 2007, Zemouri et al. 2010, Mohanty et al. 2013, Jin and Jin 2014, Bisi and Goyal 2015, Behera et al. 2021].

4 Proposed Hybrid Technique for Software Reliability Prediction

This section describes the proposed hybrid technique for software reliability prediction. For the purpose of simple and better understanding, the proposed method is divided into four subsections. Subsection 4.1 presents an overview of the proposed method with graphical abstract of the proposed neuro-fuzzy hybrid model for software reliability prediction. As the proposed method is based on the fuzzy time series and self-organized-map, first these two concepts are discussed before presenting the detailed description of proposed software reliability prediction model. Subsection 4.2 presents the brief overview of fuzzy time series (FTS) and basic concepts of fuzzy set theory such as fuzzy sets, membership function, fuzzification and defuzzification. Subsection 4.3 describes the self-organized-map and finally detailed description of the proposed method is presented in subsection 4.4.

4.1 An Overview of Proposed Method

For developing reliable software, the prediction of reliability is an important step. Software reliability prediction aids the software industry in developing good reliable software systems within budget constraints and given time. As a result, researchers have always been interested in building an efficient and accurate model for software reliability prediction. In this study a neuro-fuzzy hybridized model SOMFTS is proposed for developing accurate software reliability prediction model that can be used as an efficient tool by the software practitioners in software industries to predict reliability of a software system.

The proposed model is based on self-organized-map and fuzzy-time-series to predict software reliability. In the proposed approach, software reliability prediction of a software system is modelled as fuzzy time series forecasting problem, since software failure dataset has only one dependent variable with no independent variables.

An overview of the proposed method is as follows:

- i. A Software failure dataset having time to failure history of a software system is taken as the input for predicting time to failure values of the software system as fuzzy time series forecasting.
- ii. A Software failure dataset is partitioned into a number of intervals. For partitioning the dataset into intervals, an unsupervised two-layered neural network self-organized-map (SOM) was used.

- iii. Further, these intervals are represented by fuzzy sets by defining linguistic term.
- iv. Each observation of the software failure dataset is fuzzified according to the interval to which time between failure value belongs. For fuzzification of the software failure dataset, we have applied a triangular membership function [see Buckley 1985] on each defined fuzzy set corresponding to each interval.
- v. Construct the fuzzy logical relationships (FLRs) and fuzzy logical relationship groups (FLRGs) used to obtain the fuzzy forecast values of time to failure of software failure dataset.
- vi. Finally, Defuzzification is applied to obtain the predicted values of time to failures of software failure dataset.

The graphical representation of the proposed software reliability prediction model is shown in Figure 1.

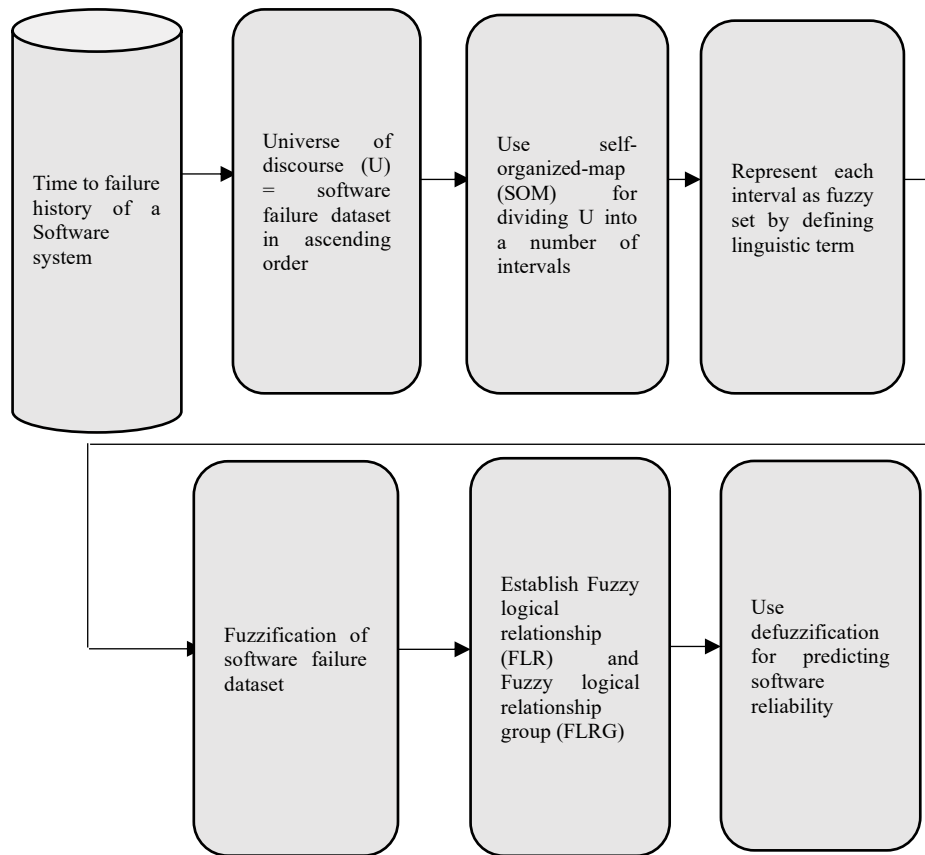


Figure 1: An overview of the proposed software reliability prediction model

4.2 Fuzzy Set and FTS - An overview

This subsection describes the basic concepts of fuzzy set theory and fuzzy time series used in this study.

4.2.1 Fuzzy Sets

Meaning of fuzziness is ‘vagueness’. To deal with the uncertainty arising due to vagueness, fuzzy set theory is an excellent mathematical tool. The concept of fuzzy set was introduced by [Zadeh 1965] for processing data in the presence of uncertainty. Fuzzy set allow the varying degree of membership (between 0 and 1) of the elements to a set. If U is the universe of discourse and particular element of U is represented by u , then a fuzzy set FS defined on U can be written as follows:

$$FS = \{(u, \mu_{FS}(u)), u \in U\} \quad (1)$$

Where $\mu_{FS}(u)$ is a membership function associated with fuzzy set FS . For example, let $U = \{s1, s2, s3, s4, s5\}$ be the universe of discourse of students. Let $FS = \{(s1, 0.5), (s2, 0.4), (s3, 1), (s4, 0.8), (s5, 0.9)\}$ be the fuzzy set of smart students, where ‘smart’ is a fuzzy linguistic term. Here FS represents the smartness of $s1$ is 0.5, $s2$ is 0.4 and so on.

4.2.2 Membership Function

Membership function provides a measure of degree of similarity of an element to a fuzzy set. Various types of membership functions are available in the literature such as triangular, trapezoidal, sigmoid and gaussian. In this study triangular membership function is used because of its simplicity and widely used in previous studies.

4.2.3 Fuzzy Time Series (FTS)

Fuzzy Time Series (FTS) is a modelling approach to predict the future values from the previous or historical values of time series by employing fuzzy set concept. Detailed description of FTS can be found in [Lu et al. 2015, Chen 2014].

4.2.4 Fuzzification and Defuzzification

Fuzzification is the process of converting the dataset of crisp values into a number of fuzzy sets using a membership function. On the other hand, defuzzification is just the reverse process of fuzzification means converting fuzzified values into the crisp values. In this study, fuzzification procedure is used to obtain the fuzzified values of time to failure values of software failure dataset for employing the concept of fuzzy time series forecasting. Defuzzification procedure is used to obtain the predicted values of time to failure values of software failure dataset used in this study. Detailed procedure of

fuzzification and defuzzification for software failure dataset used in this study is described in the section 4.3 namely detailed description of the proposed method.

4.3 Self-Organized-Map (SOM)

SOM is an unsupervised two-layered neural network based upon the principle of competitive learning [Kohonen 1990]. The first layer is designated as the input layer, and the second layer is designated as the output layer. The output layer is also called a feature map. Usually, the output layer of SOM is a one-dimensional or two-dimensional lattice of neurons. The crux of the SOM neural technique is to map each training vector onto a feature space. SOM tries to visualize the similarity between data vectors within a low-dimensional feature space. In the context of this paper, the SOM technique is integrated with the Fuzzy time Series to predict software reliability. SOM technique is applied to effectively fuzzify the time between failures datasets of software systems.

4.4 Detailed Description of Proposed Method

For software reliability prediction, the proposed method is divided into the following seven steps.

Step1: Represent software failure dataset as universe of discourse U .

Step2: Use SOM for partitioning U into the intervals.

Step3: Assign linguistic terms to each interval.

Step4: Fuzzification of software failure dataset.

Step5: Establishment of the FLRs (Fuzzy logical relationship) for software failure dataset.

Step6: Establishment of FLRGs (Fuzzy logical relationship groups) for software failure dataset.

Step7: Defuzzify and predict the software reliability from the fuzzified values of the software failure dataset.

A detailed description of each step of the proposed method with application on the software failure dataset as described in section 3 is presented as follows:

Step1: Represent software failure dataset as $U = [U_{\min}, U_{\max}]$ where U_{\min} and U_{\max} are the minimum and maximum values of time to failure, respectively. In this study $U = [4.701, 14.897]$.

Step2: Use SOM for partitioning U into the intervals.

[Description] Divide U into n number of partitions by applying the well-known clustering algorithm SOM (Self-Organized-Map) [see Kohonen 1990]. Take these partitions as the intervals of different lengths, namely fs_0, fs_1, fs_2, \dots , and fs_{n-1} . Further, each time to failure value of software failure dataset can be assigned to one of the intervals fs_0, fs_1, fs_2, \dots , and fs_{n-1} . Then, compute the centroid value of each interval by calculating the mean of all values in the corresponding interval. Intervals obtained for the software failure dataset as discussed in section 3 by applying SOM with their corresponding centroid value are listed in Table 2.

Interval	Centroid	Interval	Centroid
fs ₀ = [4.701, 4.701]	4.701	fs ₃₃ = [10.100, 10.120]	10.110
fs ₁ = [5.768, 5.894]	5.831	fs ₃₄ = [10.179, 10.196]	10.187
fs ₂ = [6.462, 6.492]	6.472	fs ₃₅ = [10.268, 10.268]	10.268
fs ₃ = [6.664, 6.664]	6.664	fs ₃₆ = [10.353, 10.353]	10.353
fs ₄ = [7.155, 7.155]	7.155	fs ₃₇ = [10.385, 10.385]	10.385
fs ₅ = [7.544, 7.544]	7.544	fs ₃₈ = [10.450, 10.450]	10.450
fs ₆ = [7.684, 7.696]	7.690	fs ₃₉ = [10.576, 10.587]	10.581
fs ₇ = [7.868, 7.901]	7.884	fs ₄₀ = [10.630, 10.660]	10.645
fs ₈ = [7.966, 7.966]	7.966	fs ₄₁ = [10.846, 10.846]	10.846
fs ₉ = [8.047, 8.071]	8.059	fs ₄₂ = [10.862, 10.862]	10.862
fs ₁₀ = [8.139, 8.139]	8.139	fs ₄₃ = [10.929, 10.929]	10.929
fs ₁₁ = [8.333, 8.333]	8.333	fs ₄₄ = [11.013, 11.040]	11.026
fs ₁₂ = [8.350, 8.350]	8.350	fs ₄₅ = [11.315, 11.315]	11.315
fs ₁₃ = [8.587, 8.594]	8.591	fs ₄₆ = [11.367, 11.375]	11.371
fs ₁₄ = [8.648, 8.671]	8.660	fs ₄₇ = [11.392, 11.392]	11.392
fs ₁₅ = [8.742, 8.742]	8.742	fs ₄₈ = [11.628, 11.640]	11.634
fs ₁₆ = [8.779, 8.788]	8.784	fs ₄₉ = [11.916, 11.916]	11.916
fs ₁₇ = [8.891, 8.891]	8.891	fs ₅₀ = [11.960, 11.960]	11.960
fs ₁₈ = [8.946, 8.967]	8.957	fs ₅₁ = [12.025, 12.025]	12.025
fs ₁₉ = [9.043, 9.043]	9.043	fs ₅₂ = [12.086, 12.091]	12.088
fs ₂₀ = [9.105, 9.105]	9.105	fs ₅₃ = [12.140, 12.140]	12.140
fs ₂₁ = [9.229, 9.229]	9.229	fs ₅₄ = [12.202, 12.202]	12.202
fs ₂₂ = [9.287, 9.287]	9.287	fs ₅₅ = [12.277, 12.279]	12.278
fs ₂₃ = [9.293, 9.293]	9.293	fs ₅₆ = [12.490, 12.497]	12.494
fs ₂₄ = [9.374, 9.374]	9.374	fs ₅₇ = [12.598, 12.598]	12.598
fs ₂₅ = [9.437, 9.437]	9.437	fs ₅₈ = [12.608, 12.608]	12.608
fs ₂₆ = [9.487, 9.487]	9.487	fs ₅₉ = [12.720, 12.721]	12.720
fs ₂₇ = [9.546, 9.546]	9.546	fs ₆₀ = [12.753, 12.783]	12.768
fs ₂₈ = [9.574, 9.575]	9.575	fs ₆₁ = [13.098, 13.368]	13.128
fs ₂₉ = [9.603, 9.620]	9.611	fs ₆₂ = [13.328, 13.368]	13.348
fs ₃₀ = [9.798, 9.798]	9.798	fs ₆₃ = [14.192, 14.192]	14.192
fs ₃₁ = [9.860, 9.860]	9.860	fs ₆₄ = [14.411, 14.557]	14.484
fs ₃₂ = [9.989, 10.003]	9.998	fs ₆₅ = [14.782, 14.897]	14.840

Table 2: Intervals with their centroid values for software failure dataset

Step3: Assign linguistic terms to each interval.

[Description] Assign n linguistic terms $FS_0, FS_1, FS_2, \dots,$ and FS_{n-1} for n number of intervals $fs_0, fs_1, fs_2, \dots,$ and fs_{n-1} respectively. Fuzzy sets for representing these linguistic terms can be described as follows:

$$\begin{aligned}
 FS_0 &= 1/fs_0 + 0.5/fs_1 + 0/fs_2 + \dots + 0/fs_{n-3} + 0/fs_{n-2} + 0/fs_{n-1}, \\
 FS_1 &= 0.5/fs_0 + 1/fs_1 + 0.5/fs_2 + \dots + 0/fs_{n-3} + 0/fs_{n-2} + 0/fs_{n-1}, \\
 FS_2 &= 0/fs_0 + 0.5/fs_1 + 1/fs_2 + \dots + 0/fs_{n-3} + 0/fs_{n-2} + 0/fs_{n-1}, \\
 &\vdots \\
 FS_{n-1} &= 0/fs_0 + 0/fs_1 + 0/fs_2 + \dots + 0/fs_{n-3} + 0.5/fs_{n-2} + 1/fs_{n-1} \quad (2)
 \end{aligned}$$

Here, interval fs_i has maximum value of membership degree of fuzzy set FS_i and $0 \leq i \leq (n-1)$. In this study 66 linguistic terms are defined as $FS_0, FS_1, FS_2, \dots, FS_{65}$ for 66 intervals generated by SOM. All linguistic terms are given as follows:

$$\begin{aligned}
 FS_0 &= 1/fs_0 + 0.5/fs_1 + 0/fs_2 + \dots + 0/fs_{63} + 0/fs_{64} + 0/fs_{65}, \\
 FS_1 &= 0.5/fs_0 + 1/fs_1 + 0.5/fs_2 + \dots + 0/fs_{63} + 0/fs_{64} + 0/fs_{65}, \\
 FS_2 &= 0/fs_0 + 0.5/fs_1 + 1/fs_2 + \dots + 0/fs_{63} + 0/fs_{64} + 0/fs_{65}, \\
 &\vdots \\
 FS_{65} &= 0/fs_0 + 0/fs_1 + 0/fs_2 + \dots + 0/fs_{63} + 0.5/fs_{64} + 1/fs_{65} \quad (3)
 \end{aligned}$$

For example, in Eq. 3 linguistic term FS_0 can be represented by fuzzy set = $\{fs_0, fs_1, fs_2, \dots, fs_{65}\}$ having 66 members with membership degree values = $\{1, 0.5, 0, \dots, 0\}$. In the similar manner FS_1 can be represented by fuzzy set = $\{fs_0, fs_1, fs_2, \dots, fs_{65}\}$ having 66 members with membership degree values = $\{0.5, 1, 0, \dots, 0\}$ and so on. Each interval has been assigned a weight. The weight of each interval is the number of elements that belong to that interval. The weight of the fs_2 interval, for example, will be 3 if it contains three elements.

Step4: Fuzzification of software failure dataset.

[Description] Each observation of the software failure dataset is fuzzified according to the interval to which time between failure value belongs. For fuzzification of the software failure dataset, we have applied a triangular membership function [see Buckley 1985] on each defined fuzzy set corresponding to each interval. Assume that maximum membership degree of time to failure after a software modification occurs at interval fs_i and $0 \leq i \leq (n-1)$, and FS_i denotes fuzzified value for that particular modification. For example, the time to failure after t_5 modification is 9.989, which belongs to the interval $fs_{32} = [9.989, 10.003]$ with the maximum membership value one, so it can be fuzzified as FS_{32} . In Eq.3, the membership degree of interval fs_0 in FS_0 and FS_1 are 1 and 0.5, respectively, with 0 membership values for the remaining fuzzy sets.

Similarly, the membership degree of interval fs_1 in $FS_0, FS_1,$ and FS_2 are 0.5, 1, and 0.5, respectively, with 0 membership values for the remaining fuzzy sets and so on. For the purpose of easy calculation membership degree for each fuzzy set is taken as 0, 0.5, or 1. The fuzzified time to failure values for the software failure dataset used in this study is listed in Table 3.

t	F_t	Fuzzified F_t	Centroid	Weight	t	F_t	Fuzzified F_t	Centroid	Weight
t_0	5.768	FS ₁	5.831	2	t_{51}	10.353	FS ₃₆	10.353	2
t_1	9.574	FS ₂₈	9.575	2	t_{52}	10.100	FS ₃₃	10.110	2
t_2	9.105	FS ₂₀	9.105	1	t_{53}	12.608	FS ₅₈	12.608	1
t_3	7.966	FS ₈	7.966	1	t_{54}	7.155	FS ₄	7.155	1
t_4	8.648	FS ₁₄	8.660	2	t_{55}	10.003	FS ₃₂	9.998	3
t_5	9.989	FS ₃₂	9.998	3	t_{56}	9.860	FS ₃₁	9.860	1
t_6	10.196	FS ₃₄	10.187	2	t_{57}	7.868	FS ₇	7.884	2
t_7	11.640	FS ₄₈	11.634	2	t_{58}	10.576	FS ₃₉	10.581	2
t_8	11.628	FS ₄₈	11.634	2	t_{59}	10.929	FS ₄₃	10.929	1
t_9	6.492	FS ₂	6.472	3	t_{60}	10.660	FS ₄₀	10.645	2
t_{10}	7.901	FS ₇	7.884	2	t_{61}	12.497	FS ₅₆	12.494	3
t_{11}	10.268	FS ₃₅	10.268	1	t_{62}	11.375	FS ₄₆	11.371	3
t_{12}	7.684	FS ₆	7.690	2	t_{63}	11.916	FS ₄₉	11.916	1
t_{13}	8.891	FS ₁₇	8.891	1	t_{64}	9.575	FS ₂₈	9.575	2
t_{14}	9.293	FS ₂₃	9.293	1	t_{65}	10.450	FS ₃₈	10.450	1
t_{15}	8.350	FS ₁₂	8.350	1	t_{66}	10.587	FS ₃₉	10.581	2
t_{16}	9.043	FS ₁₉	9.043	1	t_{67}	12.720	FS ₅₉	12.720	2
t_{17}	9.603	FS ₂₉	9.611	2	t_{68}	12.598	FS ₅₇	12.598	1
t_{18}	9.374	FS ₂₄	9.374	1	t_{69}	12.086	FS ₅₂	12.088	2
t_{19}	8.587	FS ₁₃	8.591	2	t_{70}	12.277	FS ₅₅	12.278	2
t_{20}	8.788	FS ₁₆	8.784	2	t_{71}	11.960	FS ₅₀	11.960	1
t_{21}	8.779	FS ₁₆	8.784	2	t_{72}	12.025	FS ₅₁	12.025	1
t_{22}	8.047	FS ₉	8.059	2	t_{73}	9.287	FS ₂₂	9.287	1
t_{23}	10.846	FS ₄₁	10.846	1	t_{74}	12.495	FS ₅₆	12.494	3
t_{24}	8.742	FS ₁₅	8.742	1	t_{75}	14.557	FS ₆₄	14.484	2
t_{25}	7.544	FS ₅	7.544	1	t_{76}	13.328	FS ₆₂	13.348	2
t_{26}	8.594	FS ₁₃	8.591	2	t_{77}	8.946	FS ₁₈	8.957	2
t_{27}	11.040	FS ₄₄	11.026	2	t_{78}	14.782	FS ₆₅	14.840	2
t_{28}	10.120	FS ₃₃	10.110	2	t_{79}	14.897	FS ₆₅	14.840	2
t_{29}	10.179	FS ₃₄	10.187	2	t_{80}	12.140	FS ₅₃	12.140	1
t_{30}	5.894	FS ₁	5.831	2	t_{81}	9.798	FS ₃₀	9.798	1
t_{31}	9.546	FS ₂₇	9.546	1	t_{82}	12.091	FS ₅₂	12.088	2
t_{32}	9.620	FS ₂₉	9.611	2	t_{83}	13.098	FS ₆₁	13.128	2
t_{33}	10.385	FS ₃₇	10.385	1	t_{84}	13.368	FS ₆₂	13.348	2
t_{34}	10.630	FS ₄₀	10.645	2	t_{85}	12.721	FS ₅₉	12.720	2
t_{35}	8.333	FS ₁₁	8.333	2	t_{86}	14.192	FS ₆₃	14.192	1
t_{36}	11.315	FS ₄₅	11.315	1	t_{87}	11.370	FS ₄₆	11.371	3
t_{37}	9.487	FS ₂₆	9.487	1	t_{88}	12.202	FS ₅₄	12.202	2
t_{38}	8.139	FS ₁₀	8.139	1	t_{89}	12.279	FS ₅₅	12.278	2
t_{39}	8.671	FS ₁₄	8.660	2	t_{90}	11.367	FS ₄₆	11.371	3
t_{40}	6.462	FS ₂	6.472	3	t_{91}	11.392	FS ₄₇	11.392	1
t_{41}	6.462	FS ₂	6.472	3	t_{92}	14.411	FS ₆₄	14.484	2
t_{42}	7.696	FS ₆	7.690	2	t_{93}	8.333	FS ₁₁	8.333	2
t_{43}	4.701	FS ₀	4.701	1	t_{94}	8.071	FS ₀₉	8.059	2
t_{44}	10.002	FS ₃₂	9.998	3	t_{95}	12.202	FS ₅₄	12.202	2
t_{45}	11.013	FS ₄₄	11.026	2	t_{96}	12.783	FS ₆₀	12.768	2
t_{46}	10.862	FS ₄₂	10.862	1	t_{97}	13.159	FS ₆₁	13.128	2
t_{47}	9.437	FS ₂₅	9.437	1	t_{98}	12.753	FS ₆₀	12.768	2
t_{48}	6.664	FS ₃	6.664	1	t_{99}	10.353	FS ₃₆	10.353	2
t_{49}	9.229	FS ₂₁	9.229	1	t_{100}	12.490	FS ₅₆	12.494	3
t_{50}	8.967	FS ₁₈	8.957	2					

Table 3: Fuzzified values of time to failures for the software failure dataset with their centroid and weight

Step5: Establishment of the FLRs (Fuzzy logical relationship) for software failure dataset.

[Description] According to the definition given in [Song and Chissom 1993], the establishment of the FLRs between two successive fuzzified values of time to failure for the given software failure dataset can be done as follows:

Consider $FS(t-1) = FS_i$ and $FS(t) = FS_j$, are two successive fuzzy values. The FLR between these two successive fuzzy values can be represented as follows:

$$FS_i \rightarrow FS_j, \tag{4}$$

Where FS_i and FS_j represent the preceding state and present state of FLR.

For example, from Table 3, it can be noted that fuzzified values of time to failure for t_3 and t_4 are FS_8 and FS_{14} , respectively. So, an FLR as $FS_8 \rightarrow FS_{14}$ can be established between FS_8 and FS_{14} . Similarly, FLRs for the software failure dataset can be generated and are listed in Table 4.

FLRs	FLRs	FLRs	FLRs	FLRs	FLRs
$FS_1 \rightarrow FS_{28}$	$FS_{29} \rightarrow FS_{24}$	$FS_{40} \rightarrow FS_{11}$	$FS_{36} \rightarrow FS_{33}$	$FS_{57} \rightarrow FS_{52}$	$FS_{59} \rightarrow FS_{63}$
$FS_{28} \rightarrow FS_{20}$	$FS_{24} \rightarrow FS_{13}$	$FS_{11} \rightarrow FS_{45}$	$FS_{33} \rightarrow FS_{58}$	$FS_{52} \rightarrow FS_{55}$	$FS_{63} \rightarrow FS_{46}$
$FS_{20} \rightarrow FS_8$	$FS_{13} \rightarrow FS_{16}$	$FS_{45} \rightarrow FS_{26}$	$FS_{58} \rightarrow FS_4$	$FS_{55} \rightarrow FS_{50}$	$FS_{46} \rightarrow FS_{54}$
$FS_8 \rightarrow FS_{14}$	$FS_{16} \rightarrow FS_{16}$	$FS_{26} \rightarrow FS_{10}$	$FS_4 \rightarrow FS_{32}$	$FS_{50} \rightarrow FS_{51}$	$FS_{54} \rightarrow FS_{55}$
$FS_{14} \rightarrow FS_{32}$	$FS_{16} \rightarrow FS_9$	$FS_{10} \rightarrow FS_{14}$	$FS_{32} \rightarrow FS_{31}$	$FS_{51} \rightarrow FS_{22}$	$FS_{55} \rightarrow FS_{46}$
$FS_{32} \rightarrow FS_{34}$	$FS_9 \rightarrow FS_{41}$	$FS_{14} \rightarrow FS_2$	$FS_{31} \rightarrow FS_7$	$FS_{22} \rightarrow FS_{56}$	$FS_{46} \rightarrow FS_{47}$
$FS_{34} \rightarrow FS_{48}$	$FS_{41} \rightarrow FS_{15}$	$FS_2 \rightarrow FS_2$	$FS_7 \rightarrow FS_{39}$	$FS_{56} \rightarrow FS_{64}$	$FS_{47} \rightarrow FS_{64}$
$FS_{48} \rightarrow FS_{48}$	$FS_{15} \rightarrow FS_5$	$FS_2 \rightarrow FS_6$	$FS_{39} \rightarrow FS_{43}$	$FS_{64} \rightarrow FS_{62}$	$FS_{64} \rightarrow FS_{11}$
$FS_{48} \rightarrow FS_2$	$FS_5 \rightarrow FS_{13}$	$FS_6 \rightarrow FS_0$	$FS_{43} \rightarrow FS_{40}$	$FS_{62} \rightarrow FS_{18}$	$FS_{11} \rightarrow FS_{09}$
$FS_2 \rightarrow FS_7$	$FS_{13} \rightarrow FS_{44}$	$FS_0 \rightarrow FS_{32}$	$FS_{40} \rightarrow FS_{56}$	$FS_{18} \rightarrow FS_{65}$	$FS_{09} \rightarrow FS_{54}$
$FS_7 \rightarrow FS_{35}$	$FS_{44} \rightarrow FS_{33}$	$FS_{32} \rightarrow FS_{44}$	$FS_{56} \rightarrow FS_{46}$	$FS_{65} \rightarrow FS_{65}$	$FS_{54} \rightarrow FS_{60}$
$FS_{35} \rightarrow FS_6$	$FS_{33} \rightarrow FS_{34}$	$FS_{44} \rightarrow FS_{42}$	$FS_{46} \rightarrow FS_{49}$	$FS_{65} \rightarrow FS_{53}$	$FS_{60} \rightarrow FS_{61}$
$FS_6 \rightarrow FS_{17}$	$FS_{34} \rightarrow FS_1$	$FS_{42} \rightarrow FS_{25}$	$FS_{49} \rightarrow FS_{28}$	$FS_{53} \rightarrow FS_{30}$	$FS_{61} \rightarrow FS_{60}$
$FS_{17} \rightarrow FS_{23}$	$FS_1 \rightarrow FS_{27}$	$FS_{25} \rightarrow FS_3$	$FS_{28} \rightarrow FS_{38}$	$FS_{30} \rightarrow FS_{52}$	$FS_{60} \rightarrow FS_{36}$
$FS_{23} \rightarrow FS_{12}$	$FS_{27} \rightarrow FS_{29}$	$FS_3 \rightarrow FS_{21}$	$FS_{38} \rightarrow FS_{39}$	$FS_{52} \rightarrow FS_{61}$	$FS_{36} \rightarrow FS_{56}$
$FS_{12} \rightarrow FS_{19}$	$FS_{29} \rightarrow FS_{37}$	$FS_{21} \rightarrow FS_{18}$	$FS_{39} \rightarrow FS_{59}$	$FS_{61} \rightarrow FS_{62}$	
$FS_{19} \rightarrow FS_{29}$	$FS_{37} \rightarrow FS_{40}$	$FS_{18} \rightarrow FS_{36}$	$FS_{59} \rightarrow FS_{57}$	$FS_{62} \rightarrow FS_{59}$	

Table 4: FLRs for the software failure dataset

Step6: Establishment of FLRGs (Fuzzy logical relationship groups) for software failure dataset.

[Description] As per the definition of FLRGs given in [Cheng 1996], FLRs having the same preceding state can be collected together in the same FLRG. Consider the following FLRs:

$$\begin{aligned} &FS_i \rightarrow FS_{k1}, \\ &FS_i \rightarrow FS_{k2}, \\ &FS_i \rightarrow FS_{k3}, \\ &\cdot \\ &\cdot \\ &\cdot \\ &FS_i \rightarrow FS_{km} \end{aligned} \quad (5)$$

Now all the FLRs on the right-hand side of Eq. 5 can be collected together in the same FLRG as follows:

$$FS_i \rightarrow FS_{k1}, FS_{k2}, FS_{k3} \dots, FS_{km}. \quad (6)$$

For example, from Table 4, it can be observed that three FLRs $FS_2 \rightarrow FS_7$, $FS_2 \rightarrow FS_2$, and $FS_2 \rightarrow FS_6$ have the same preceding state and hence can be grouped to form the FLRGs as $FS_2 \rightarrow FS_7, FS_2, FS_6$. Thus, all FLRGs for the software failure dataset can be constructed in a similar way.

After the construction of FLRGs, assign trends as described in [Singh 2018] to each fuzzy set in the present state of FLRGs according to the following three cases:

Case1 Upward trend (\uparrow):

If the index value of the preceding state's fuzzy set is less than the index value of the present state's fuzzy set, the fuzzy set trend of the present state will be upward (\uparrow).

Case2 Downward trend (\downarrow):

If the index value of the preceding state's fuzzy set is greater than the index value of the present state's fuzzy set, the fuzzy set trend of the present state will be downward (\downarrow).

Case3 Unchanged ($=$):

If the index value of the preceding state's fuzzy set is equal to the index value of the present state's fuzzy set, the present state's fuzzy set trend will remain unchanged ($=$).

For example, consider FLRG: $FS_2 \rightarrow FS_7, FS_2, FS_6$. Here FS_7 , FS_2 , and FS_6 can be associated with the trends FS_7 (\uparrow), FS_2 ($=$), and FS_6 (\uparrow) respectively. Therefore, all the FLRGs for the software failure dataset are listed in Table 5.

FLRGs	Preceding state → Present state	FLRGs	Preceding state → Present state
FLRG1	FS ₀ → FS ₃₂ (↑)	FLRG34	FS ₃₃ → FS ₃₄ (↑), FS ₅₈ (↑)
FLRG2	FS ₁ → FS ₂₇ (↑), FS ₂₈ (↑)	FLRG35	FS ₃₄ → FS ₁ (↓), FS ₄₈ (↑)
FLRG3	FS ₂ → FS ₂ (=), FS ₆ (↑), FS ₇ (↑)	FLRG36	FS ₃₅ → FS ₆ (↓)
FLRG4	FS ₃ → FS ₂₁ (↑)	FLRG37	FS ₃₆ → FS ₃₃ (↓), FS ₅₆ (↑)
FLRG5	FS ₄ → FS ₃₂ (↑)	FLRG38	FS ₃₇ → FS ₄₀ (↑)
FLRG6	FS ₅ → FS ₁₃ (↑)	FLRG39	FS ₃₈ → FS ₃₉ (↑)
FLRG7	FS ₆ → FS ₀ (↓), FS ₁₇ (↑)	FLRG40	FS ₃₉ → FS ₄₃ (↑), FS ₅₉ (↑)
FLRG8	FS ₇ → FS ₃₅ (↑), FS ₃₉ (↑)	FLRG41	FS ₄₀ → FS ₁₁ (↓), FS ₅₆ (↑)
FLRG9	FS ₈ → FS ₁₄ (↑)	FLRG42	FS ₄₁ → FS ₁₅ (↓)
FLRG10	FS ₉ → FS ₄₁ (↑), FS ₅₄ (↑)	FLRG43	FS ₄₂ → FS ₂₅ (↓)
FLRG11	FS ₁₀ → FS ₁₄ (↑)	FLRG44	FS ₄₃ → FS ₄₀ (↓)
FLRG12	FS ₁₁ → FS ₉ (↓), FS ₄₅ (↑)	FLRG45	FS ₄₄ → FS ₃₃ (↓), FS ₄₂ (↓)
FLRG13	FS ₁₂ → FS ₁₉ (↑)	FLRG46	FS ₄₅ → FS ₂₆ (↓)
FLRG14	FS ₁₃ → FS ₁₆ (↑), FS ₄₄ (↑)	FLRG47	FS ₄₆ → FS ₄₇ (↑), FS ₄₉ (↑), FS ₅₄ (↑)
FLRG15	FS ₁₄ → FS ₂ (↓), FS ₃₂ (↑)	FLRG48	FS ₄₇ → FS ₆₄ (↑)
FLRG16	FS ₁₅ → FS ₅ (↓)	FLRG49	FS ₄₈ → FS ₂ (↓), FS ₄₈ (=)
FLRG17	FS ₁₆ → FS ₉ (↓), FS ₁₆ (=)	FLRG50	FS ₄₉ → FS ₂₈ (↓)
FLRG18	FS ₁₇ → FS ₂₃ (↑)	FLRG51	FS ₅₀ → FS ₅₁ (↑)
FLRG19	FS ₁₈ → FS ₃₆ (↑), FS ₆₅ (↑)	FLRG52	FS ₅₁ → FS ₂₂ (↓)
FLRG20	FS ₁₉ → FS ₂₉ (↑)	FLRG53	FS ₅₂ → FS ₅₅ (↑), FS ₆₁ (↑)
FLRG21	FS ₂₀ → FS ₈ (↓)	FLRG54	FS ₅₃ → FS ₃₀ (↓)
FLRG22	FS ₂₁ → FS ₁₈ (↓)	FLRG55	FS ₅₄ → FS ₅₅ (↑), FS ₆₀ (↑)
FLRG23	FS ₂₂ → FS ₅₆ (↑)	FLRG56	FS ₅₅ → FS ₄₆ (↓), FS ₅₀ (↓)
FLRG24	FS ₂₃ → FS ₁₂ (↓)	FLRG57	FS ₅₆ → FS ₄₆ (↓), FS ₆₄ (↑)
FLRG25	FS ₂₄ → FS ₁₃ (↓)	FLRG58	FS ₅₇ → FS ₅₂ (↓)
FLRG26	FS ₂₅ → FS ₃ (↓)	FLRG59	FS ₅₈ → FS ₄ (↓)
FLRG27	FS ₂₆ → FS ₁₀ (↓)	FLRG60	FS ₅₉ → FS ₅₇ (↓), FS ₆₃ (↑)
FLRG28	FS ₂₇ → FS ₂₉ (↑)	FLRG61	FS ₆₀ → FS ₃₆ (↓), FS ₆₁ (↑)
FLRG29	FS ₂₈ → FS ₂₀ (↓), FS ₃₈ (↑)	FLRG62	FS ₆₁ → FS ₆₀ (↓), FS ₆₂ (↑)
FLRG30	FS ₂₉ → FS ₂₄ (↓), FS ₃₇ (↑)	FLRG63	FS ₆₂ → FS ₁₈ (↓), FS ₅₉ (↓)
FLRG31	FS ₃₀ → FS ₅₂ (↑)	FLRG64	FS ₆₃ → FS ₄₆ (↓)
FLRG32	FS ₃₁ → FS ₇ (↓)	FLRG65	FS ₆₄ → FS ₁₁ (↓), FS ₆₂ (↓)
FLRG33	FS ₃₂ → FS ₃₁ (↓), FS ₃₄ (↑), FS ₄₄ (↑)	FLRG66	FS ₆₅ → FS ₅₃ (↓), FS ₆₅ (=)

Table 5: FLRGs for the software failure dataset

Step7: Defuzzify and predict the software reliability from the fuzzified values of the software failure dataset.

[Description] The fuzzified value at a time (t-1) is necessary to predict software reliability at time t. The complete procedure can be divided into three cases. The first two cases are adopted from the frequency distribution-based defuzzification technique given in [Singh 2018]. The third case is included to deal with the situation where the

fuzzified value of the preceding state is not present in FLRGs. A detailed explanation of all the cases is given below:

Case1: This case will be applicable when the present state has more than one fuzzified value. Description of this case is given below:

- Step i: Obtain the fuzzified value of time to failure time (t-1) as FS_i.
- Step ii: FLRG of the form “FS_i→FS_{j0}, FS_{j1}, ...FS_{jp}” is obtained from FLRGs.
- Step iii: Obtain the interval fs_i where fuzzy set FS_i (preceding state) has the maximum degree of membership.
- Step iv: Further, variation v_i in the time to failure for fuzzy set FS_i can be computed using the following formula.

$$v_i = \frac{c_i \times w_i}{100} \tag{7}$$

Here, c_i and w_i represents the centroid of corresponding interval fs_i and weightage of fuzzy set FS_i, respectively.

Step v: Find intervals fs_{j0}, fs_{j1}, ..., fs_{jp} for the maximum membership value of fuzzy sets FS_{j0}, FS_{j1}, ..., FS_{jp}, respectively. Calculate centroids c_{j0}, c_{j1}, ..., c_{jp} for fs_{j0}, fs_{j1}, ..., fs_{jp} respectively.

Step vi: Compute FS_{centroid}, mean of centroids c_{j0}, c_{j1}, ..., c_{jp} by using the following formula.

$$FS_{centroid} = \frac{c_{j0} + c_{j1} + \dots + c_{jp}}{p} \tag{8}$$

Here p denotes the number of fuzzy sets in a FLRG’s present state.

Step vii: Compute trend value by using the following formula.

$$T = \frac{(c_{j0} * v_i) + (c_{j1} * v_i) + \dots + (c_{jp} * v_i)}{p} \tag{9}$$

Here * represents the operator addition or subtraction depending upon downward trend and upward trend, respectively. For the trend unchanged, the centroid value is taken as it is.

Step viii: Predict software reliability using the following formula.

$$FS_{predict} = \frac{(FS_{centroid} + T)}{2} \tag{10}$$

Case2: This case will be applicable when the present state has only one fuzzified value. Description of this case is given below:

Step i: Obtain the fuzzified value of time to failure at the time (t-1) as FS_i

Step ii: FLRG of the form “FS_i→FS_j” is obtained from FLRGs.

Step iii: Obtain the interval fs_i where fuzzy set FS_i (preceding state) has the maximum degree of membership.

Step iv: Further, variation v_i in the time to failure for fuzzy set FS_i can be computed using the following formula.

$$v_i = \frac{c_i \times w_i}{100} \tag{11}$$

Here, c_i and w_i represents the centroid of corresponding interval fs_i and weightage of fuzzy set FS_i, respectively.

Step v: Compute trend value by using the following formula.

$$T = (c_i * v_i) \tag{12}$$

Here *, c_i, and v_i have the same meaning as in case 1.

Step vi: Predict software reliability using the following formula.

$$FS\ predict = \frac{c_j + T}{2} \tag{13}$$

Case3: This case will be applicable when the fuzzified value of the preceding state is not present in FLRGs. Description of this case is given below:

This case is applicable when the previous state fuzzified value is not present in FLRGs.

A detailed explanation of this case is given below:

Step i: Obtain the fuzzified value of time to failure at the time (t-1) as FS_i

Step ii: Take the centroid of interval fs_i as the predicted value since there is no rule in FLRGs corresponding to FS_i as the preceding state.

5 Performance Measures

For the evaluation of the proposed model, four performance measures have been considered as evaluation criteria. All four measures are defined as follows:

- Normalized Root Mean Square Error (NRMSE) can be defined as:

$$NRMSE = \sqrt{\frac{\sum_{i=1}^n (predict_i - actual_i)^2}{\sum_{i=1}^n actual_i^2}} \tag{14}$$

- Average Prediction Error Rate (APER) can be defined as:

$$APER = \left(\frac{\sum_{i=1}^n (predict_i - actual_i) / actual_i}{n} \right) \times 100 \quad (15)$$

- Mean Absolute Error (MAE) can be defined as:

$$MAE = \frac{1}{n} \sum_{i=1}^n |predict_i - actual_i| \quad (16)$$

- Correlation Coefficient (r) gives the strength of the relationship between two variables (in this study, actual value and predicted value). This study chooses Pearson's correlation coefficient and can be calculated using the following equation.

$$r = \frac{n(\sum actual_i)(\sum predict_i) - (\sum actual_i)(\sum predict_i)}{\sqrt{[n \sum actual_i^2 - (\sum actual_i)^2][n \sum predict_i^2 - (\sum predict_i)^2]}} \quad (17)$$

In this study, $actual_i$ and $predict_i$ are the actual and predicted value of time to failure of the software after i^{th} modification in the software, and n is the total number of predictions for time to failure.

6 Experimental Results

This section is further divided into two subsections. The first subsection presents computed predictions of time to failure for the software failure dataset used in this study, as described in section 3. The second subsection describes the results of four performance measures as defined in section 5 for the proposed SOMFTS method and four existing fuzzy time series models for comparative study. SOMFTS was implemented using MATLAB R2021 version 9.10. Open-source package PyFTS was used to implement four existing fuzzy time series models, namely [Chen 1996], [Yu 2005], [Cheng et al. 2009], and [Efendi et al. 2013].

6.1 Results of Predictions for Time to Failures for Software Failure Dataset

Computed predicted values of time to failures of the software failure dataset as described in section 3 using the proposed SOMFTS approach are listed in Table 6. Here two examples have been explained to compute the predicted values of the time to failure of the software system by applying the proposed SOMFTS approach.

Example 1: For predicting the time to failure value at t_6 , the fuzzified time to failure value at t_5 is required. From Table 3, the fuzzified time to failure value at t_5 can be obtained, which is FS_{32} . Next, obtain the FLRG for which the preceding state is FS_{32} . From Table 5, this FLRG can be obtained as FLRG33 in the form $FS_{32} \rightarrow FS_{31} (\downarrow)$, $FS_{34} (\uparrow)$, $FS_{44} (\uparrow)$. Because there is more than one fuzzified time to failure value (FS_{31} , FS_{34} , and FS_{44}) are available in the present state here, case 1 is applicable. In FLRG $FS_{32} \rightarrow FS_{31} (\downarrow)$, $FS_{34} (\uparrow)$, $FS_{44} (\uparrow)$, the fuzzified time to failure values in the present state exhibit two different trends as an upward trend ($FS_{34} (\uparrow)$, $FS_{44} (\uparrow)$) and downward trend ($FS_{31} (\downarrow)$). Next, obtain the interval fs_{32} from Table 2, for which the fuzzy set FS_{32} has the maximum degree of membership. The centroid value for this interval fs_{32} is 9.998 ($=C_{32}$). Here the weight of fuzzy set FS_{32} is 3 ($=W_{32}$). Compute the variation in time to failure for fuzzy set FS_{32} as $V_{32} = [9.998 * 3 / 100] = 0.299$ using Eq. (7).

Next, from Table 2, find the intervals for which fuzzy sets FS_{31} , FS_{34} , and FS_{44} have the maximum membership degree values, fs_{31} , fs_{34} , and fs_{44} , respectively. Centroid for these intervals fs_{31} , fs_{34} and fs_{44} are 9.860 ($=C_{31}$), 10.187 ($=C_{34}$) and 11.026 ($=C_{44}$) respectively. Average of centroids can be calculated as $F_{\text{centroid}} = (9.860 + 10.187 + 11.026) / 3 = 10.358$ using Eq. (8). Now, using Eq. (9) trend value can be calculated as: $T = (9.860 + 0.299) / 3 + (10.187 - 0.299) / 3 + (11.026 - 0.299) / 3 = 10.258$. Here, the addition and subtraction operation are done for the downward trend and upward trend, respectively. Now, prediction for time to failure value at t_6 can be computed using Eq. (10) as follows: $F_{\text{predict}} = (10.358 + 10.258) / 2 = 10.308$.

Example 2: For predicting the time to failure value at t_{15} , the fuzzified time to failure value at t_{14} is required. From Table 3, the fuzzified time to failure value at t_{14} can be obtained, which is FS_{23} . Next, obtain the FLRG for which the preceding state is FS_{23} . From Table 5, this FLRG can be obtained as FLRG24 in the form $FS_{23} \rightarrow FS_{12} (\downarrow)$. Here case 2 is applicable since only one fuzzified time to failure value (FS_{12}) is available in the present state. In FLRG $FS_{23} \rightarrow FS_{12} (\downarrow)$, the fuzzified time to failure value in the present state exhibits a downward trend. Next, obtain the interval fs_{23} from Table 2, for which the fuzzy set FS_{23} has the maximum degree of membership. The centroid value for this interval fs_{23} is 9.293 ($=C_{23}$). Here the weight of fuzzy set FS_{23} is 1 ($=W_{23}$). Compute the variation in time to failure value for fuzzy set FS_{23} using Eq. (11) as follows: $V_{23} = [9.293 * 1 / 100] = 0.09293$. Next, from Table 2, find the interval for which fuzzy set FS_{12} has the maximum membership degree, fs_{12} . The centroid for this interval fs_{12} is 8.350 ($=C_{12}$). Now, using Eq. (9) trend value can be calculated as $T = (8.350 + 0.09293) = 8.443$. Here, an addition operation is done for the downward trend. Now, prediction for time to failure value at t_{15} can be computed using Eq. (13) as follows: $F_{\text{predict}} = (8.350 + 8.443) / 2 = 8.396$.

t	F_t	F_t'	t	F_t	F_t'	t	F_t	F_t'
t_0	5.768	-	t_{34}	10.630	10.593	t_{68}	12.598	13.395
t_1	9.574	9.502	t_{35}	8.333	10.414	t_{69}	12.086	12.151
t_2	9.105	9.778	t_{36}	11.315	9.687	t_{70}	12.277	12.582
t_3	7.966	8.011	t_{37}	9.487	9.544	t_{71}	11.960	11.788
t_4	8.648	8.620	t_{38}	8.139	8.187	t_{72}	12.025	11.965
t_5	9.989	8.235	t_{39}	8.671	8.619	t_{73}	9.287	9.347
t_6	10.196	10.308	t_{40}	6.462	8.235	t_{74}	12.495	12.448
t_7	11.640	8.733	t_{41}	6.462	7.284	t_{75}	14.557	12.927
t_8	11.628	9.111	t_{42}	7.696	7.284	t_{76}	13.328	10.985
t_9	6.492	9.111	t_{43}	4.701	6.796	t_{77}	8.946	10.972
t_{10}	7.901	7.284	t_{44}	10.002	9.975	t_{78}	14.782	12.507
t_{11}	10.268	10.346	t_{45}	11.013	10.308	t_{79}	14.897	13.564
t_{12}	7.684	7.741	t_{46}	10.862	10.596	t_{80}	12.140	13.564
t_{13}	8.891	6.796	t_{47}	9.437	9.492	t_{81}	9.798	9.859
t_{14}	9.293	9.249	t_{48}	6.664	6.519	t_{82}	12.091	12.039
t_{15}	8.350	8.396	t_{49}	9.229	9.196	t_{83}	13.098	12.582
t_{16}	9.043	9.001	t_{50}	8.967	9.003	t_{84}	13.368	13.058
t_{17}	9.603	9.566	t_{51}	10.353	12.507	t_{85}	12.721	10.972
t_{18}	9.374	9.879	t_{52}	10.100	11.302	t_{86}	14.192	13.395
t_{19}	8.587	8.637	t_{53}	12.608	11.297	t_{87}	11.370	11.441
t_{20}	8.788	9.819	t_{54}	7.155	7.218	t_{88}	12.202	11.666
t_{21}	8.779	8.465	t_{55}	10.003	9.962	t_{89}	12.279	12.401
t_{22}	8.047	8.465	t_{56}	9.860	10.308	t_{90}	11.367	11.788
t_{23}	10.846	11.443	t_{57}	7.868	7.934	t_{91}	11.392	11.666
t_{24}	8.742	8.796	t_{58}	10.576	10.346	t_{92}	14.411	14.427
t_{25}	7.544	7.588	t_{59}	10.929	11.685	t_{93}	8.333	10.985
t_{26}	8.594	8.553	t_{60}	10.660	10.700	t_{94}	8.071	9.687
t_{27}	11.040	9.819	t_{61}	12.497	10.414	t_{95}	12.202	11.443
t_{28}	10.120	10.596	t_{62}	11.375	12.927	t_{96}	12.783	12.401
t_{29}	10.179	11.297	t_{63}	11.916	11.666	t_{97}	13.159	11.741
t_{30}	5.894	8.733	t_{64}	9.575	9.634	t_{98}	12.753	13.058
t_{31}	9.546	9.502	t_{65}	10.450	9.778	t_{99}	10.353	11.741
t_{32}	9.620	9.563	t_{66}	10.587	10.529	t_{100}	12.490	11.302
t_{33}	10.385	9.879	t_{67}	12.720	11.685			

Table 6: Predictions for time to failure for software failure dataset using proposed approach SOMFTS

6.2 Comparative Study

Considering the corpus of fuzzy time series models, comparing the proposed approach to all of the fuzzy time series models is impossible. To assess the proposed method's performance, we have compared it with four existing fuzzy time series models, namely [Chen 1996], [Yu 2005], [Cheng et al. 2009], and [Efendi et al. 2013]. The experimental results in terms of four performance measures as described in section 5 for the proposed method SOMFTS and four existing fuzzy time series models are presented in Table 7.

Model/Performance Measures	NRMSE	APER	MAE	Correlation Coefficient (r)
SOMFTS (Proposed Model)	0.1037	1.241	0.7252	0.8540
[Chen 1996]	0.1625	2.7902	1.3048	0.5794
[Yu 2005]	0.1685	5.3125	1.3131	0.5520
[Cheng et al. 2009]	0.1700	5.5065	1.3411	0.5356
[Efendi et al. 2013]	0.1622	3.2484	1.2915	0.5803

Table 7: Experimental Results for software reliability prediction

7 Discussion of Results

From Table 7, it can be observed that the proposed method SOMFTS has the minimum values 0.1037, 1.241, and 0.7252 for NRMSE, APER, and MAE, respectively. On the other hand, SOMFTS has a maximum value of 0.8540 for coefficient correlation (r). Thus, the proposed approach SOMFTS outperforms the four existing fuzzy time series methods used in this study in terms of all the four performance results.

8 Conclusion

This paper proposes a novel approach by integrating a well-known unsupervised clustering technique, self-organized-map, and fuzzy time series (FTS) forecasting for the reliability prediction of a software system based on its failure history. An experimental study was done by applying SOMFTS on a software failure dataset for the military system to validate the proposed approach. A comparison study was also carried out to assess the performance of the proposed approach by comparing SOMFTS to four existing FTS models. Based on experimental study, the proposed method can be employed as an efficient tool in the software industry for software reliability prediction. The proposed software reliability prediction method will aid the software industry in developing good reliable software systems within budget constraints and given time. Furthermore, applying the proposed approach to a large number of datasets and in other fields may be the direction for future work.

References

[Amin et al. 2013] Amin, A., Grunske, L., Colman, A.: "An approach to software reliability prediction based on time series modeling"; *Journal of Systems and Software*, 86, 7 (2013), 1923-1932.

- [Behera et al. 2019] Behera, A. K., Nayak, S. C., Dash, C. S. K., Dehuri, S., Panda, M.: "Improving software reliability prediction accuracy using CRO-based FLANN"; Proc. Innovations in Computer Science and Engineering, Springer, Singapore (2019), 213-220.
- [Behera et al. 2021] Behera, A. K., Panda, M., Dehuri, S.: "Software reliability prediction by recurrent artificial chemical link network"; International Journal of System Assurance Engineering and Management, (2021), 1-14.
- [Benala et al. 2013] Benala, T. R., Chinnababu, K., Mall, R., Dehuri, S.: "A particle swarm optimized functional link artificial neural network (PSO-FLANN) in software cost estimation"; Proc. International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA), Springer, Berlin, Heidelberg (2013), 59-66.
- [Bisi and Goyal 2015] Bisi, M., Goyal, N. K.: "Prediction of software inter-failure times using artificial neural network and particle swarm optimisation models"; International Journal of Software Engineering, Technology and Applications, 1, 2 (2015), 222-244.
- [Bisi and Goyal 2016] Bisi, M., Goyal, N.K.: "Software development efforts prediction using artificial neural network"; IET Software, 10, 3 (2016), 63-71.
- [Buckley 1985] Buckley, J. J.: "Ranking alternatives using fuzzy numbers"; Fuzzy sets and systems, 15, 1 (1985), 21-31.
- [Cai et al. 2001] Cai, K. Y., Cai, L., Wang, W. D., Yu, Z. Y., Zhang, D.: "On the neural network approach in software reliability modeling"; Journal of Systems and Software, 58, 1 (2001), 47-62.
- [Chen 2014] Chen, MY.: "A high-order fuzzy time series forecasting model for internet stock trading"; Future Generation Computer Systems, 37, (2014), 461-467.
- [Chen 1996] Chen, S. M. (1996): "Forecasting enrollments based on fuzzy time series"; Fuzzy sets and systems, 81, 3 (1996), 311-319.
- [Cheng et al. 2009] Cheng, C.-H., Chen, Y.-S., Wu, Y.-L.: "Forecasting innovation diffusion of products using trend-weighted fuzzy time-series model"; Expert Systems with Applications, 36, 2 (2009), 1826-1832.
- [Efendi et al. 2013] Efendi, R., Ismail, Z., Deris, M. M.: "Improved weight Fuzzy Time Series as used in the exchange rates forecasting of US Dollar to Ringgit Malaysia"; International Journal of Computational Intelligence and Applications, 12, 1 (2013), 1350005.
- [Iannino and Musa 1997] Iannino, A., Musa, J. D.: "Software Reliability"; Advances in Computers, 30 (1997), 85-170.
- [Jabeen et al. 2017] Jabeen, G., Yang, X., Ping, L., Rahim, S., Sahar, G., Shah, A. A.: "Hybrid software reliability prediction model based on residual errors"; Proc. 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), IEEE, (2017), 479-482.
- [Jabeen et al. 2019a] Jabeen, G., Yang, X., Luo, P., Rahim, S.: "Application of Grey-Markov Chain Model in Software Reliability Prediction"; Journal of Computers, 30, 3 (2019), 14-27.
- [Jabeen et al. 2019b] Jabeen, G., Luo, P., Afzal, W.: "An improved software reliability prediction model by using high precision error iterative analysis method"; Software Testing, Verification & Reliability, 29, 6-7 (2019).
- [Jaiswal and Malhotra 2018] Jaiswal, A., Malhotra, R.: "Software reliability prediction using machine learning techniques"; International Journal of System Assurance Engineering and Management, 9, 1 (2018), 230-244.

- [Jin and Jin 2014] Jin, C., Jin, S. W.: "Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms"; *Applied Soft Computing*, 15 (2014), 113-120.
- [Juneja 2019] Juneja, K.: "A fuzzy-filtered neuro-fuzzy framework for software fault prediction for inter-version and inter-project evaluation"; *Applied Soft Computing*, 77 (2019), 696-713.
- [Karunanithi et al. 1992] Karunanithi, N., Whitley, D., Malaiya, Y. K.: "Using neural networks in reliability prediction"; *IEEE Software*, 9, 4 (1992), 53-59.
- [Kassaymeh et al. 2021] Kassaymeh, S., Abdullah, S., Al-Laham, M., Alweshah, M., Al-Betar, M. A., Othman, Z.: "Salp Swarm Optimizer for Modeling Software Reliability Prediction Problems"; *Neural Processing Letters*, (2021), 1-37.
- [Kiran and Ravi 2007] Kiran, N. R., Ravi, V.: "Software Reliability Prediction by Soft Computing Techniques"; *The Journal of Systems and Software*, 81, 4 (2007), 576-583.
- [Kohonen 1990] Kohonen, T.: "The self-organizing map"; *Proc. IEEE*, (1990), 1464-1480.
- [Kumar and Singh 2012] Kumar, P., Singh, Y.: "An empirical study of software reliability prediction using machine learning techniques"; *International Journal of System Assurance Engineering and Management*, 3, 3 (2012), 194-208.
- [Kumaresan and Ganeshkumar 2020] Kumaresan, K., Ganeshkumar, P.: "Software reliability prediction model with realistic assumption using time series (S) ARIMA model"; *Journal of Ambient Intelligence and Humanized Computing*, 11, 11 (2020), 5561-5568.
- [Li et al. 2013] Li, K., Zhao, K., Liu, W.: "Neural network ensemble based on K-means clustering individual selection and application for software reliability prediction"; *Proc. Fourth World Congress on Software Engineering*, IEEE (2013), 131-135.
- [Li et al. 2019] Li, Z., Yu, M., Wang, D., Wei, H.: "Using hybrid algorithm to estimate and predicate based on software reliability model"; *IEEE Access*, 7 (2019), 84268-84283.
- [Li and Malaiya 1993] Li, N., Malaiya, K. Y.: "Enhancing Accuracy of Software Reliability Prediction"; *Proc. 1993 IEEE International Symposium on Software Reliability Engineering*, (1993), 71-79.
- [Lou et al. 2016] Lou, J., Jiang, Y., Shen, Q., Shen, Z., Wang, Z., Wang, R.: "Software reliability prediction via relevance vector regression"; *Neurocomputing*, 186 (2016), 66-73.
- [Lu et al. 2015] Lu, W., Chen, X., Pedrycz W., Liu, X., Yang, J.: "Using interval information granules to improve forecasting in fuzzy time series"; *International Journal of Approximate Reasoning*, 57, (2015), 1-18.
- [Ma et al. 2018] Ma, Z. Y., Wang, J. P., Zhang, W., Shan, Z. W., Liu, F. S., Han, K.: "Software reliability prediction based on optimized Support Vector Regression"; *Proc. 2018 International Conference on Big Data and Computing*, (2018), 129-133.
- [Mallikharjuna and Kodali 2017] Mallikharjuna, R. K., Kodali, A.: "An Efficient Method for Enhancing Reliability and Selection of Software Reliability Growth Model through Optimization Techniques"; *Journal of Software*, 12, 1 (2017), 1-8.
- [Mohanty et al. 2013] Mohanty, R., Ravi, V., Patra, M.R.: "Hybrid Intelligent Systems for Predicting Software Reliability"; *Applied Soft Computing*, 13 (2013), 189-200.
- [Pandey et al. 2020] Pandey, S. K., Mishra, R. B., Tripathi, A. K.: "BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques"; *Expert Systems with Applications*, 144 (2020), 113085.

- [Pham 2006] Pham, H.: "System Software Reliability"; Springer, London, England (2006)
- [Ramakrishna et al. 2012] Ramakrishna, V., Rao, M. N., Padmaja, T. M.: "Software Reliability Prediction using Neural Networks"; *International Journal of Computer Applications*, 60, 7 (2012), 44-48.
- [Roy et al. 2014] Roy, P., Mahapatra, G. S., Rani, P., Pandey, S. K., Dey, K. N.: "Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction"; *Applied Soft Computing*, 22 (2014), 629-637.
- [Roy et al. 2017] Roy, P., Mahapatra, G. S., Dey, K. N.: "An efficient particle swarm optimization-based neural network approach for software reliability assessment"; *International Journal of Reliability, Quality and Safety Engineering*, 24, 4 (2017), 1750019.
- [Roy et al. 2019] Roy, P., Mahapatra, G. S., Dey, K. N.: "Forecasting of software reliability using neighborhood fuzzy particle swarm optimization based novel neural network"; *IEEE/CAA Journal of Automatica Sinica*, 6, 6 (2019), 1365-1383.
- [Singh 2018] Singh, P.: "Rainfall and financial forecasting using fuzzy time series and neural networks-based model"; *International Journal of Machine Learning and Cybernetics*, 9, 3 (2018), 491-506.
- [Song and Chissom 1993] Song, Q., Chissom, B. S.: "Forecasting enrollments with fuzzy time series—Part I"; *Fuzzy sets and systems*, 54, 1 (1996), 1-9.
- [Song and Chang 2014] Song, K. Y., Chang, I. H.: "Parameter estimation and prediction for NHPP software reliability model and time series regression in software failure data"; *Journal of the Chosun Natural Science*, 7, 1 (2014), 67-73.
- [Sudharson 2020] Sudharson, D.: "Hybrid software reliability model with Pareto distribution and ant colony optimization (PD-ACO)"; *International Journal of Intelligent Unmanned Systems*, 8, 2 (2020), 129-140.
- [Tian and Noore 2005] Tian, L., Noore, A.: "Evolutionary neural network modeling for software cumulative failure time prediction"; *Reliability Engineering & system safety*, 87, 1 (2005), 45-51.
- [Yu 2005] Yu, H. K.: "Weighted fuzzy time series models for TAIEX forecasting"; *Physica A: Statistical Mechanics and its Applications*, 349, 3-4 (2005), 609-624.
- [Zadeh 1965] Zadeh, L.A.: "Fuzzy Sets"; *Information and Control*, 8, (1965), 338-353.
- [Zemouri and Patic 2010] Zemouri, R., Patic, P. C.: "Recurrent Radial Basis Function Network for Failure Time Series Prediction"; *International Journal of Computer and Information Engineering*, 4, 12 (2010), 1920-1924.
- [Zhen et al. 2020] Zhen, L., Liu, Y., Dongsheng, W., Wei, Z.: "Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization"; *IEEE Access*, 8 (2020), 29354-29369.