Int J Syst Assur Eng Manag (March 2022) 13(Suppl. 1):S273–S280
https://doi.org/10.1007/s13198-021-01389-0

ORIGINAL ARTICLE

Nonlinear dynamic measurement method of software reliability based on data mining

Yinsheng Fu[1] · Jullius Kumar[2] · Bibhu Prasad Ganthia[3] · Rahul Neware[4]

Received: 12 August 2021 / Revised: 27 August 2021 / Accepted: 8 September 2021 / Published online: 30 October 2021

Abstract Developing high-quality software is the ultimate goal of any software development organization. But the major challenge is to achieve good quality. It can usually only be measured after delivery, and reliability is the primary measure of software quality. During development, there are many attempts to assess software quality. To solve the reliability problem of evaluating software, the data mining model of BP neural network is proposed to predict the reliability of software. Firstly, data mining is carried out on the number of faults of the software, and data such as the cumulative execution time and the corresponding observed cumulative number of faults in the testing process of the software within a set of specific times are collected. Secondly, the training model of BP neural network is built according to the failure data samples, and the software is trained and learned according to the historical data, it is used to test the cumulative execution time of the future stage, calculate the corresponding predicted cumulative failure number of the software, and then verify the reliability of the target software. The example proves that the BP neural network is more accurate in predicting the 17th, 18th, and 19th groups of cumulative failure times compared with the traditional nonlinear modes, Jelinski-Moranda model, Goel-Okumoto model and Yamada S-shaped model, the number of faults predicted is more the prediction accuracy is higher, and it is more suitable for application and reliability evaluation of software.

Keywords Data mining · Software reliability · Nonlinear measurement · BP neural network · Jelinski-Moranda model (JM) · Goel-Okumoto model (GO)

✉ Jullius Kumar
Julian.vns@gmail.com

Yinsheng Fu
YinshengFu8@163.com

Bibhu Prasad Ganthia
jb.bibhu@gmail.co

Rahul Neware
rane@hvl.no

[1] Anyang Vocational and Technical College, Anyang 455000, Henan, China

[2] Department of Computer Applications, Dr. Rammanohar Lohia Avadh University, Ayodhya, Uttar Pradesh, India

[3] Department of Electrical Engineering, IGIT Sarang, Saran, Odisha, India

[4] Department of Computing, Mathematics and Physics, Høgskulen På Vestlandet, Bergen, Norway

1 Introduction

Developing high-quality software is the ultimate goal of any software development organization, but one of the most challenging aspects of quality, it can usually only be measured after delivery, and reliability is the primary measure of software quality. During development, there are many attempts to assess software quality. Such estimates are likely to aid the engineering of high-quality software by providing useful insights to project managers. Data mining is the use of various heuristic methods or tools to collect a large set of data, which is stored in a database or data warehouse for analysis, the goal is to discover hidden patterns and relationships in the data set and summarize the data in a form that decision makers can understand, that is to realize "data, information, knowledge, value"

transformation process. In the software, data mining technology can be used to mine, collect effective data, and store it in the database to sort out data, improve document security, and filter dangerous and useless information.

Visagan A. R, organizations are typically equipped with data related to their past software versions that can be used to build models that can help predict the reliability of the software being developed. This study uses data collected from Sonar Cloud to build a model that can help predict software reliability (Visagan et al. 2020a). He H. proposed an optimized software fault mode miner based on complex network. By analyzing the relationship between multiple execution trajectories of software and functions, a weighted software execution dependency graph model is finally established, the traversal database is generated by the depth-first search strategy, and the extraction of the software path traversal to evaluate the reliability of the software (He et al. 2020). Agrawal V proposed an unsupervised software fault prediction method to improve the reliability of the software. The purpose of this method is to determine whether degrees exist in the early life cycle: (requirements measurement), whether it is possible to select the measures available later in the lifecycle (code metrics) to use clustering techniques or classification techniques to identify modules prone to failure and determine which technique can get better results (Agrawal 2017; Bhola et al. 2021). Many researchers (Li et al. 2017; Rajbahadur et al. 2017; Kishor et al. 2021a; Lu et al. 2015) have tried to use the data of one project to predict the defects for some other projects and known as cross project defect prediction (CPDP).

Based on the extensive literature survey it is found that while developing new software it is very important to achieve the optimum quality level for the software. There are number of techniques, methods and models which can be used for training and testing of the software. The key point of software reliability evaluation is how to model the reliability of a software system. BP artificial neural network is an important branch of data mining technology. It must have the self—learning, self—organization, good fault—tolerance, and good nonlinear approximation ability. So the purpose of this study is to verify the advantages of artificial neural network as a nonlinear model compared with traditional linear model in software reliability prediction and evaluation.

2 Methods

2.1 Principle of BP neural network

The learning rule of BP neural network is to use the gradient descent method to adjust the weight and threshold of the network constantly through back propagation, to minimize the sum of error squares of the network. The topological structure of BP neural network model includes Input Layer, hid Layer and Output Layer, as shown in Fig. 1. The hidden layer can be one or more layers, and there is no coupling in the nodes of the same layer. The input signal is transmitted from the input layer node to the hidden layer node and then to the output layer node. The output of each layer of nodes only affects the output of the next layer of nodes.

Each node cell mimics the three most basic and important functions of biological neurons: Weighting, summation and transfer. The neuron model is shown in Fig. 2, where $x_1, x_2, \ldots, x_i, \ldots, x_n$, representing neuron 1 from the upper layer, the input of 2, …, *i*. *n*. $w_{j1}, w_{j2}, \ldots, w_{ji}, \ldots, w_{jn}$ represents the connection strength between the neuron in the upper layer and the jth neuron in this layer, the weight. b_j is the threshold. $f(\cdot)$ is the transfer function. y_j is the output of the jth neuron in this layer. As shown in formula (1).

$$yj = f\left(\sum_{i=1}^{n} w_{ji} \cdot x_i + b_j\right) \tag{1}$$

In BP neural network, the transfer function of a node element is usually sigmoid type function, but in the output layer, the node element is sometimes a linear function. BP neural network can be regarded as a highly nonlinear mapping of input to output. Homik et al. have proved that: If the input layer and output layer adopt the linear transformation functions, and the hidden layer adopts sigmoid transformation functions, then a network with a hidden layer can approximate any rational function with arbitrary precision.

2.2 BP network learning algorithm

The standard BP neural network learning algorithm is as follows:

Compute the output of nodes at each layer in the network.As shown in formula (2).

$$\text{Out}_j = f\left(\sum_{i=1}^{n} w_{ji} x_i + b_j\right) \quad j = 1, 2, \ldots, m \tag{2}$$

Outj represents the output of the jth neuron in this layer, transfer function adopts $f(x) = 1/(1+\exp(-x))$, w_{ji} stands for weight, b_j stands for threshold, n is the number of neurons in the upper layer, and m is the number of neurons in this layer.

Calculate the total error E to determine whether the tolerance threshold is reached, the total error E between the expected output and the actual output is formula (3):

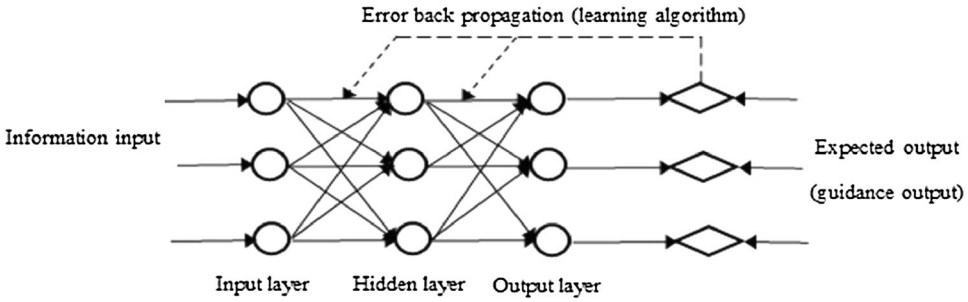

Fig. 1 Neural network structure based on BP algorithm

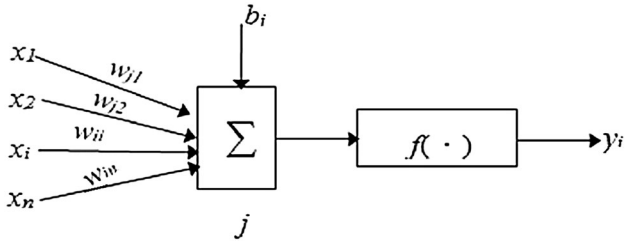

Fig. 2 BP neurons

$$E = \frac{1}{2}\sum_{p}\sum_{i}\left(d_{p_i} - o_{p_i}\right)^2 \tag{3}$$

where, d_{pi} is the expected output value of neuron i corresponding to the pth input mode. o_{pi} is the actual output of the neuron. If the total error E reaches the tolerance threshold, the learning is finished, otherwise, reverse transitional learning is carried out.

Calculate the network error

For neurons in the output layer, the error signal δ_j is formula (4)

$$\delta_j = \mathrm{Out}_j\left(1 - \mathrm{Out}_j\right)\left(d_{pi} - o_{pi}\right) \tag{4}$$

For neuron j in the hidden layer, there is no special target value. The error signal δ_j is determined recursively in terms of all neurons k that are directly connected to neuron j and their weights. As shown in formula (5).

$$\delta_j = \mathrm{Out}_j\left(1 - \mathrm{Out}_j\right)\left(\sum_{k}\delta_k w_{kj}\right) \tag{5}$$

(4) Modify the network weight.

During the nth iteration, the weights are adjusted according to the following criteria.As shown in formula (6), (7):

$$\Delta W_{ij}(n+1) = \eta\delta_j \mathrm{Out}_i \tag{6}$$

$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij(}n+1) \tag{7}$$

Type: $W_{ij}(n+1)$ is between neuron i, which provides input, and neuron j, which lies on the next hidden or output layer, the weight in the n+1 round. η is learning rate. Out_i is the output of the ith neuron.

The entire learning process iterates until the total error E reaches the tolerance threshold. In practice, people will also provide a maximum number of iterations M. Learning also ends when the number of iterations exceeds M.

2.3 Reliability modeling based on BP network

Software reliability evaluation is to predict the reliability of target software based on the analysis of software failure data. Dynamic prediction of the number of software failures can be simply defined as: firstly, a set of cumulative execution times collected in the process of software testing up to the current time and the corresponding cumulative number of observed faults are given, then, the cumulative execution time of the future test phase is given to calculate the corresponding predicted cumulative failure number (Ding and Xing 2020). According to the mapping method of neural network, it can be simply described as: $P : \{(I_t, O_t), i_{t+h}\} \rightarrow O_{t+h}$, among them, $\{(I_t, O_t)\}$ represents the failure history of the software system, O_{t+h} is the prediction result of i_{t+h} in the future time.

After a certain number of tests are carried out on the software system, the collected software failure data can be used as the training samples of the neural network to build a model and predict the cumulative failure number in the future. The basic method of neural network is: according to the training algorithm and user experience, the structure of BP network is statically organized, and then the historical data of software failure is used to train (or learn) the network until its performance meets certain requirements. The trained neural network can evaluate the reliability of the software in the future (Visagan et al. 2020b).

The BP network structure is designed as 3 layers: Input layer, hidden layer, and output layer, while the dynamic prediction of software failure number has only one node in the input layer, which represents the cumulative execution time 1, and the number of nodes in the hidden layer is determined by experience, generally, it is $2 \sim 3$ times of the input node. The output layer also has only one node, which represents the cumulative number of failures. In practice, it

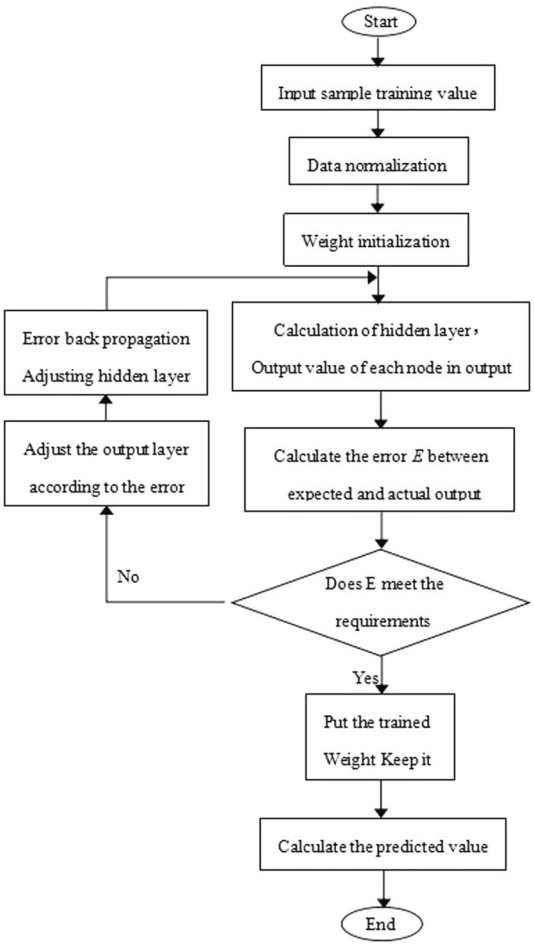

Fig. 3 Neural network training and prediction process

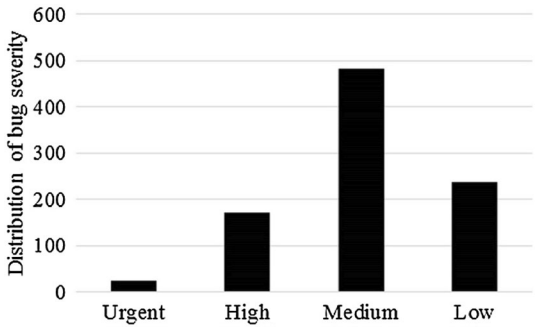

Fig. 4 Bug severity distributions

is found that such a network structure is not easy to converge. The main reason is that the input pattern space is one-dimensional and does not provide enough information for network learning. Therefore, the input pattern space must be extended (Krejsa et al. 2017; Jairath et al. 2021). The usual method is to use an orthogonal complete sets, such as the form extension function of $\sin \pi x$, $\cos \pi x$ and $\sin 2\pi x$, $\cos 2\pi x$, x, x^1, x^2, $x^3 \ldots x^{12}$, the prediction effect is better.

Therefore, 12–24-1BP forward network is adopted, that is, there are 12 nodes in the input layer, 24 nodes in the hidden layer, and 1 node in the output layer. The predicted network training process is as follows:

Normalize the time t, t^2, …, t^{12}, of the input node to $0 \sim 1$, and the expected value of the, output node, that is the cumulative number of, faults tested is reduced to $0 \sim 1$.

Initialize the weights of each layer of the network.

Compute the output of each layer of nodes in the network.

The global error E is calculated to determine whether the tolerance threshold is reached. If the allowable range is reached, the training ends. Otherwise, go to Step (5).

To reduce the error between the actual output and the expected output.

After all weights and thresholds are corrected, return to step (3) to continue the training. After the network is trained, the cumulative execution time parameters of the software are sent to the network, and the output value of the output layer is the cumulative fault number predicted by the BP network. The flow chart is shown in Fig. 3.

3 Examples and conclusions

3.1 System structure of test software

A bank's general security management system BHRF is tested according to BP network model. After several rounds of modification and testing, the function of BHRF system has become stable, and its reliability needs to be tested and evaluated. To this end, a large amount of failure data of the software was collected in advance. The severity of the bug was divided into four levels: Urgent (the most serious), High (High), Medium (middle), and Low (Low). The distribution of bug severity is shown in Fig. 4.

Since bugs with low severity are mostly related to the display of interface elements, while software reliability refers to the failure of software functions, only bugs with medium severity are counted (Nosratian et al. 2020; Kishor et al. 2021b). The cumulative test time of THE BHRF system and the corresponding cumulative failure times are shown in Table 1.

Software safety reliability evaluation to verify the nonlinearity of BP network needs to be compared with the traditional linear evaluation model.

Table 1 BHRF failure data statistics

Serial number	Cumulative test time(hours)	Cumulative failure times	Serial number	Cumulative test time(hours)	Cumulative failure times
1	40	17	11	440	155
2	8o	32	12	480	168
3	120	50	13	520	176
4	160	67	14	560	183
5	200	81	15	600	190
6	240	93	16	640	197
7	280	105	17	680	203
8	320	115	18	720	211
9	360	125	19	760	223
10	400	140	20	800	233

3.2 Reliability evaluation based on BP network

Using MATLAB 6.5 to design and simulate the BP neural network. According to the reliability evaluation principle based on BP network described above, the structure of BP neural network is designed. The network consists of three layers: input layer, hidden layer and output layer (Zhang et al. 2018; Mahajan et al. 2021). The dimension of the input layer is 12, the hidden layer has 24 neurons, and the output layer contains 1 neuron, as shown in Fig. 5. The hidden layer uses the logs transfer function (LOGSIG), which outputs data as $\log sig(n) = 1/(1 + \exp(-n))$. The output layer uses a linear transfer function (PURELIN), which outputs the data according to the original value according to pureline $(n) + n$.

TRAINGDM was used as the training function. TRAINGDM uses the momentum gradient descent back propagation algorithm to train the network. When updating weights and thresholds, this function not only considers the current gradient direction, but also the gradient direction of the previous moment, to reduce the sensitivity of network performance to parameter adjustment and effectively suppress local minimization (Eppelbaum 2015). Other training parameters are set as follows:

netrain.Param.epochs = 100,000.
net.train.Param.goal = 0.00001.
net.train.Param.Ir = 0.01.
nettrain.Param.max_ fail = 5.
net.train.Param.mc = 0.9
net.train.Param.min_ grad = le-010.
net.rain.Param.show = 25.
net.train.Param.time = inf.

Taking the data in Table 1 as the training set, the input mode space is orthogonal extended according to the x, x^2, $\ldots x^{12}$ function form described above, the cumulative test time and cumulative failure times in the training set were normalized. Then input the network for training. When the network converges after 100,000 cyclic iterations, the training ends.

The last four groups of data in Table 1 are taken as the test set. After the processing of test data through the Orthogonal, extension and normalization the cumulative test time of the tests is recorded. After completion of processing the data is used as input into the BP neural network trained above (Chakraborty et al. 2015; Dash et al. 2021). The simulation results are obtained. After the reverse normalization, the cumulative failure times of the software when the cumulative test time reaches 680, 720,

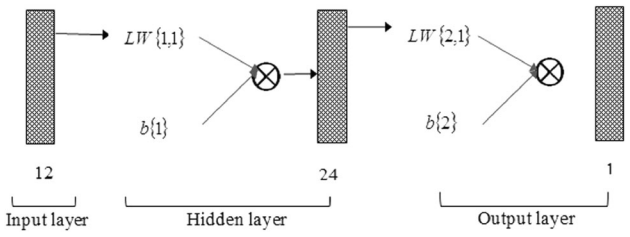

Fig. 5 Neural network simulation structure designs

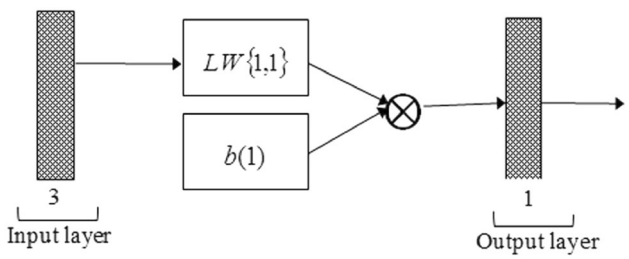

Fig. 6 Neural network simulation structure design

Table 2 Actual cumulative failure times and simulation results of each model

Serial number	Actual accumulation Failure (Times)	JM	GO	S-Shape	BP	Comprehensive model
1	17	17	17.79	5.32	19	15.57
2	32	34	34.66	18.25	35.71	33.48
3	50	50	50.67	35.35	51.11	49.93
4	67	65	65.84	54.26	65.2	65.21
5	81	80	80.23	73. 46	78.17	79.61
6	93	93	93 88	91.98	90.33	93.16
7	105	106	106.82	109.25	102.18	106.09
8	115	119	119.09	124.97	1 14.36	118.44
9	125	131	130.73	139.03	127.56	130.24
10	140	I 42	141.77	151.42	142.01	141.52
11	155	153	152.24	162.22	156.45	152.34
12	168	163	162.16	171.55	167.52	162.69
13	176	172	171.58	179.55	173.06	172.573
14	183	182	180.50	186.36	184.36	182.1
15	190	190	188.97	192.13	189.98	191.13
16	197	199	197.00	197.00	197.04	199. 84

Table 3 Test data sets

Serial number	Actual accumulation failure (Times)	JM	GO	S-Shape	BP	Comprehensive model
17	203	206	208.08	210.12	207.35	225.17
18	211	204.61	214	216.02	223.86	209.71
19	223	201.08	211.83	221	223.55	239.05
20	233	204.89	204.5	218.67	228	230.76

760, and 800 h are, respectively: (204.8970, 210.1200,223.8680,239.0520).

3.3 Reliability evaluation of the comprehensive model based on linear network

To compare the model performance, three classic software reliability models are selected to model the software failure data in Table 1, the selected models are: Jelinski-Moranda model (JM), Goel-Okumoto model (GO), and Yamada S-Shaped model (S-Shaped) (Bhuyan et al. 2016). Linear neural network structure, including input layer and output layer with two levels. The dimension of the input layer is 3, and the dimension of the output layer is 1, as shown in Fig. 6. The output layer uses a linear transfer function (PURELIN), which outputs data according to the original value purelin(n) = n.

Similarly, we modeled the first 16 groups of data in Table 1 and predicted the last 4 groups of data. The actual cumulative failure times, THE simulation results of JM model, GO model, S-shaped model, BP neural network model, and the comprehensive model based on linear neural network are listed in Tables 2 and 3.

It can be seen from the comparison results in Tables 2 and 3 that the BP neural network prediction method has higher prediction accuracy than the traditional model (JM model, GO model, S-Shape model) the feasibility and effectiveness of artificial neural network for software reliability prediction and evaluation are proved. However, it can be seen from the forecast data in Table 3 that BP neural network can predict the 17th, 18th, and 19th groups of cumulative failure times more accurately, while the prediction deviation for the 20th group is large. Through similar multiple simulations, it is found that the short-term prediction effect of BP neural network is better than the long-term prediction effect, and the average error tends to become larger as time goes on (He et al. 2020). If long-term prediction is required, the rolling training method can

be adopted, that is, as time goes by, new samples are continuously added for training, which can improve the long-term prediction accuracy of the neural network.

4 Conclusions

Data mining plays an important role in many fields and plays a significant role in software reliability prediction and evaluation. After software development is completed, a large number of failures and invalid data will be generated in the process of testing. BP neuron model is built according to these data, through intensive learning of the data, the ability to judge and predict faults is formed, and the reliability of the software is evaluated. The topological structure of BP neural network model includes Input Layer, hid Layer, and Output Layer, which can be regarded as a highly nonlinear mapping from input to output. If the input layer and output layer adopt the linear transformation function and the hidden layer adopts sigmoid transformation functions, then the network with a hidden layer can approximate any rational function with arbitrary precision. In this study, the BP neuron model is compared with the traditional linear data mining model, according to the experimental data, the BP neuron data mining model as a nonlinear data mining method, compared with the traditional linear mining model, it has higher accuracy in fault prediction and troubleshooting.

Reliability of software is the first prerequisite for its use, and it is also an urgent computing problem to be solved. Software with poor reliability will not only fail to function as it should, but also cause failures and increase operating costs. It is proved that nonlinear data mining has a broad application prospect to realize the reliability evaluation problem of software. However, there are still some caves. For example, although some computer scientists have shown that the light layer in the BP neuron model can achieve infinite approximation to any function, it's just a corollary in the actual operation, it still needs to rely on the knowledge level and practical experience of experiencers, so it still has a broad research space in the future.

Based on the results it can be concluded from the forecast data in Table 3 that BP neural network can predict the 17th, 18th, and 19th groups of cumulative failure times more accurately, while the prediction deviation for the 20th group is large.

Funding This research work is self-funded.

Declarations

Conflict of interest The authors declare that they have no conflict of interest and all ethical issues including human or animal participation has been done. No such consent is applicable.

References

Agrawal V (2017) Reliability of Software Fault Prediction Using Data Mining and Fuzzy Logic 34(21):120–128

Bhola J, Shabaz M, Dhiman G et al (2021) Performance evaluation of multilayer clustering network using distributed energy efficient clustering with enhanced threshold protocol. Wireless Pers Commun. https://doi.org/10.1007/s11277-021-08780-x

Bhuyan MK, Mohapatra DP, Sethi S (2016) Prediction strategy for software reliability based on recurrent neural network. Springer India 25(31):156–163

Chakraborty C, Gupta B, Ghosh SK (2015) Identification of chronic wound status under tele-wound network through smartphone. Int J Rough Sets and Data Analysis 2(2):58–77. https://doi.org/10.4018/ijrsda.2015070104

Dash S, Chakraborty C, Giri SK, Pani SK (2021) Intelligent computing on time-series data analysis and prediction of COVID-19 pandemics. Pattern Recogn Lett 151:69–75. https://doi.org/10.1016/j.patrec.2021.07.027

Ding Z, Xing L (2020) Improved software defect prediction using pruned histogram-based isolation forest. Reliab Eng Syst Saf 12(32):201–212

Eppelbaum L (2015) High-precise gravity observations at archaeological sites: how we can improve the interpretation effectiveness and reliability? Assem Eur Union of Geosci 42(28):156–162

He H, Shan C, He H, Zhao G, Tian X (2020) osfpminer: an optimal weighted traversal software pattern miner based on complex network. Chin J Electron 29(2):255–264

Jairath K, Singh N, Jagota V, Shabaz M (2021) Compact ultrawide band metamaterial-inspired split ring resonator structure loaded band notched antenna. Math Probl Eng 2021:1–12. https://doi.org/10.1155/2021/5174455

Kishor A, Chakraborty C, Jeberson W (2021a) Reinforcement learning for medical information processing over heterogeneous networks. Multimed Tools Appl 80:23983–24004. https://doi.org/10.1007/s11042-021-10840-0

Kishor A, Chakraborty C, Jeberson W (2021b) Intelligent healthcare data segregation using fog computing with internet of things and machine learning. Int J Eng Syst Model Simul 12(2–3):188–194

Krejsa M, Janas P, Krejsa V (2017) Application of the doproc method in solving reliability problems. Appl Mech Mater 8(21):717–724

Li Z, Jing XY, Zhu X, Zhang H, Xu B, Ying S (2017) On the multiple sources and privacy preservation issues for heterogeneous defect prediction. IEEE Trans softw Eng 45(4):391–411

Lu J, Behbood V, Hao P, Zuo H, Xue S, Zhang G (2015) Transfer learning using computational intelligence: a survey. Knowl-Based Syst 80:14–23

Mahajan K, Garg U, Shabaz M (2021) CPIDM: a clustering-based profound iterating deep learning model for HSI segmentation. Wireless Commun Mobile Comput. https://doi.org/10.1155/2021/7279260

Nosratian S, Moradkhani M, Tavakoli MB (2020) Fuzzy-based reliability prediction model for secure routing protocol using ga and Tlbo for the implementation of black hole attacks in Wsn. J Circuits, Syst Comput 12(33):521–533

Rajbahadur GK, Wang S, Kamei Y, Hassan AE (2017) The impact of using regression models to build defect classifiers. In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR) 135–145. IEEE

Visagan AR, Sumathi M, & Sujatha G (2020) Building a Data Mining Based Software Reliability Estimation Model. 2019 1st International Conference on Advances in Information Technology (ICAIT). IEEE.24(35),56–64.

Visagan AR, Sumathi M, & Sujatha G (2020) Building a Data Mining Based Software Reliability Estimation Model. 2019 1st International Conference on Advances in Information Technology (ICAIT), IEEE,56(27),138–143.

Zhang GL, Yang QH, Cheng XM, Jiang K, Wang S, Tan WK et al (2018) Application of sequence pattern mining in communication network alarm prediction. Comput Sci 20(51):325–337