International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 11, No.1 (Feb-2022)

https://dx.doi.org/10.12785/ijcds/110169

A Literature Survey on Optimization and Validation of Software Reliability Using Machine Learning

Ms. Poorva Sanjay Sabnis[1] **and Dr. S.D. Joshi**[2]

[1]*Research Scholar, Bharti Vidyapeeth (Deemed to be University) College of Engineering, Pune*
[2]*Professor, Bharti Vidyapeeth (Deemed to be University) College of Engineering, Pune*

Received 27 Jul. 2021, Revised 14 Sep. 2021, Accepted 28 Jan. 2022, Published 15 Feb. 2022

Abstract: The important characteristics of a software quality assurance system is software reliability. It is the probability of error-free software process in a given environment for a given interval. A huge number of research projects have been attempted to increase the software's reliability. Software modelling, software measurement, and software improvement are a three-step process for enhancing software reliability. Decision Trees (DT), Support Vector Machines (SVM), Artificial Neural Networks (ANN), and other Machine Learning (ML) techniques for forecasting software reliability are reviewed in this study. Software Reliability is a measure for the success whether the software is functioning as per expectations in a given time (interval or point) in the environment that is prevailing. The purpose of ML methods to forecast software stability has yielded careful and impressive findings. To identify the value of every approach in assessing the competence of software dependability prediction models, a comparative study is also undertaken. In this review paper, several methods of Software Reliability and outcomes using ML are explained and reviewed.

Keywords: Software Reliability, Software Reliability Model, Machine Learning Techniques, Prediction Tools

1. Introduction

Software reliability is described as the capacity to execute a method without malfunction for a specific period of time. There are numerous obstacles in forecasting and estimating software reliability. To make the system highly effective, less maintenance with fewer errors, new techniques and methodologies must be used to anticipate and estimate software reliability. Other factors impacting reliability and reusability can make the system more suitable for integrating them. The study is underway to lower the system's complexities and failure rate. Calculating the best cost when there is a wide area with a population and the random movement of various components is a difficult task. Component-Based Software Development (CBSD) facilitates the creation of a reliable model in a simple and fast manner. This is a complicated undertaking to create a modern system that is more efficient and takes less time.

As a result, Component-Based Software Engineering (CBSE) can be utilized to create new software that saves money and time while also delivering high quality and dependability. CBSE, according to Goulo, is a branch of software engineering which relies on component reuse. By combining numerous components to accomplish a specific purpose, a software component created a component model.

Software and software-controlled frameworks are becoming increasingly important in society. A part of this commodity is safety-related such as the product that controls trucks, ships and other forms of rapid transportation. Defects in protection basic software can result in real harm or decease. Business-basic software, such as that which runs on mobile phones, drives web servers and manages server farms, accounts for a much larger percentage of software. Imperfections in this type of software can result in significant financial losses. Frameworks software, which includes low-level working frameworks, device drivers, systems management and compilers software, is supporting these territories [1].

As per the composition standard, the components can be individually positioned and built without changing their properties. As a result, Component Based Software Reliability (CBSR) is reliant on the interaction of returnable components. CBS depend on component connections. It is challenging to determine CBSR when component connectivity is complex. The interfaces among components and the reusability of components influence component selection is the integration of these components. Soft computing is becoming increasingly popular in the field of dependability estimation and prediction research. Many optimization strategies have been employed in order to provide the lowest local/global cost in order to meet the goal [1].

E-mail address: poorva.sabnis89@gmail.com

The requirement for high-quality, maintainable software at lower prices has grown as the software's complexity and dependability have grown. Before a system is implemented, software defect prediction is a critical and crucial activity for improving software quality and reducing maintenance effort [2].

Early defect identification can lead to fast fault rectification and the supply of maintainable software. In the literature, there are a variety of software metrics. Fault data and software metrics can be utilized to build models that can forecast problematic modules/classes early on in the software development life cycle. By identifying modules/classes as fault prone.

Software Fault Prediction (SFP) can be performed. SFP models can be built using software fault and software metrics data taken from a previous release or comparable project. The model created can then be applied to actual software projects' modules/classes to identify them as fault prone or not. Knowing a software program's level of reliability is both an important and difficult topic. Claiming a level of reliability that is significantly different from the genuine value can have serious consequences depending on the context in which the software is utilized. Testing is the most popular method for determining software reliability and, more crucially, the level of confidence in the assessment [3][4].

Typically, stimulating real-world test operational usage (known as operational testing) are undertaken during the assessment process, and (failure) data is collected to evaluate dependability [5]. The main challenge is to offer an impartial (i.e., its expectation is the genuine dependability) and efficient estimate of reliability. Statistical sampling methods are a logical way to deal with this challenge because they aim to create sampling plans that are customized to the population under study and produce estimators with the qualities described. While unbiasedness (and other basic criteria such as consistency and sufficiency) [6] are easier to attain, the most important factor to consider when choosing an estimator is its efficiency in proportion to the number of observations needed. However, there are few attempts to go beyond the typical random or operational testing in the present literature on sampling-based software testing. Simple sampling approaches that, despite generating unbiased estimates, necessitate a high number of test cases to achieve acceptable confidence, especially when the software contains few residual errors (e.g., in critical systems) [7].

During the early stages of software development, software practitioners can focus existing testing resources on the fault-prone portions of the product. If only 30 percent of testing resources are obtainable, for example, knowing the weaker sections can allow testers to spend the remaining efforts on repairing the classes/modules which are further prone to mistakes. As a result, in the time and budget allotted, high-quality, low-cost and workable software can be built. Soft computing approaches are particularly suited for real-world situations that require strategies for extracting usable information from complicated and intractable difficulties in a shorter amount of time.

These are tolerant of data that is sloppy, incomplete, or unreliable. ML techniques are one of the most significant components of soft handling methods. In the literature, Accepted Manuscript 3 learning (ML) techniques have been utilized to forecast models for assessing incorrect modules/classes. Singh et al. [8], for example, used artificial neural networks and decision trees to forecast fault classes at varied difficulty levels. Using these classification algorithms for anticipating malfunctioning modules/classes has various advantages [9].

Although this is the Introduction of Software Reliability, after this the next heading is Literature Review that shows some related work about software reliability using ML. In the next heading (Software Reliability Forecast Utilizing ML Methods) several ML methods are explained. In the next section, The Metrics Applied for Model Evaluation Correlation coefficient are explained in which Mean absolute Error (MAE), Root mean Square Error (RMSE), Linear Regression, Sensitivity are discussed. The Result Analysis section shows the detailed information of ML methods. In Last a conclusion heading is added that conclude all the information about software reliability using ML.

2. Literature Review for Software Reliability

In this section, some latest related work of Software Reliability utilizing ML techniques are explained.

Sudharson et al. (2018) [10] described the crucial quantitative attribute in attaining dependability in software products by measuring problems during testing. Time-based software dependability models are used to identify product flaws, but these are ineffective in dynamic scenarios. In a few excursions, the test effect is employed instead of time, and it is not feasible to test for an unlimited amount of time. In software reliability models, identifying the amount of defects is critical, and this research paper uses an ANN and a Pareto distribution (PD) to forecast the fault distribution of software under homogeneous and nonhomogeneous settings. This method allows for the concurrent evolution of a product using Neural Network models that have been predicted to be Pareto optimal in terms of several error measurements. Current growth models, like the 2 fuzzy time series-based software reliability models and homogeneous poison process, do not accurately represent types of failure data, while the suggested PD-ANN-based SRGM does. By creating solutions for various product and developer indices, experimental proof for broad application and the proposed framework is shown.

Das et al. (2018) [11] suggested a method for estimating the frequency of faults or failures in software using a recurrent neural network. Using a dataset gathered from the literature, the effectiveness of deep learning is compared to

that of typical ML methods. ML approaches like support vector regression, naive bayes, decision trees, and random forest algorithms have been found to be effective in categorizing bug data from data with interdependent feature sets. In this study, a deep learning approach to estimating software reliability is suggested.

Tamura et al. (2018) [12] suggested a beneficial method based on deep learning for OSS reliability enhancement operations. Additionally, apply the existing software reliability model to the bug tracking system's fault data. Develop an application software for visualizing and assessing the dependability of defect data recorded on OSS, in particular. This study also includes various numerical illustrations of the established application software in the actual OSS project. Then, utilizing failure data sets from genuine OSS projects [12], present the analysis results related to the built application software.

Rajasekhar et al. (2017) [13] suggested Wall and Ferguson Reliability Growth Model are used to calculate the mean time for the cumulative observations of failure data. Also have collected the data sets from different source and calculated the mean time for each and every failure data, for these apply SPC technique to get the control charts in order to observe the failures. The failure data sets are given as input to the weka which is a data machine tool.

Malhotra, Ruchika et al. (2016) [14] predicted software reliability on various platforms, researchers used ML methods such as general regression neural network, SVM adaptive neuro fuzzy inference system (ANFIS), M5P, Bagging, multilayer perceptron, feed forward back propagation neural network, cascading forward back propagation neural network, reduced error pruning tree, instance-based learning, linear regression, and M5Rules.

Bhuyan et al. (2016) [15] during the system testing phase with fault removal, the model was used to data sets obtained across numerous mainstream software projects. Our approach, unlike most connectionist models, tries to calculate normalized root mean square error (NRMSE), average error (AE), RMSE, and MAE all at the same time. The performance of the proposed feed-forward neural network is compared to that of some traditional parametric software reliability growth models.

Kumar et al. (2012) [16] suggested models for which have mean absolute error, root mean squared error, correlation coefficient, and precision were used to evaluate. For empirical studies, the 16 software life cycle databases were employed. For software, these datasets are obtained from the analysis and data center. To identify the importance of each approach for assessing the competence of software reliability prediction models, a comparison analysis is undertaken.

3. Software Reliability Forecast Utilizing ML Methods

It is well-known and commonly applied ML approaches like ANN, SVMs, DTs, and Fidelity National Information Services (FIS) to forecasting programming unwavering consistency in light of past programming association dissatisfaction behavior. The potential of ANNs to demonstrate abstract non-direct relations and estimate some quantifiable potential permits them attractive candidates for deciphering complicated errands without needing the construction of a specific model for the system. SVM in other words, is a perusing mechanism that generates an N-dimensional hyper plane which preferably separates the data into 2 groups.

The basic pre-position of SVM showing is to describe the perfect hyper plane which separates classes of vectors, like the cases with a classification of the dependent variable on one side of the cases and plane with the additional classification of the autonomous variable on the different side of the plane. Vectors closest to the hyper plane are known as support vectors. SVMs can be applied as an alternative preparation method for spiral premise work, multilayer and polynomial perception schemes utilizing a section of this manner.

The weights of the work in SVMs are calculated by explaining a quadratic programming problem with straight criteria rather than by explaining an unconstrained minimization, non-arched, problem as in the planning of the neural net example. It has been observed that SVM is capable of summarizing fine, smooth in high-dimensional spaces, under minimal preparation test conditions. As a result of the auxiliary threat minimization, SVMs have a superior potential to speculate.

A. SVM

These are fascinating for remotely sensed data categorization by various technique which is unaffected by data dimensionality and so does not involve a dimensionality-reduction analysis in preprocessing. A series of classification evaluations using 2 hyperspectral sensor data sets demonstrates that the precision of an SVM classification varies with the amount of features used. Importantly, it is demonstrated that adding characteristics to a classification can reduce its accuracy significantly (at the 0.05 threshold of statistical significance), especially if a limited training sample is employed.

SVMs are a group of supervised learning techniques [17]. SVM belong to the kind of general linear grouping. SVM has the unusual property of lowering the experiential classification error while rising the geometric margin. It is also identified as Max Margin Classifiers as a result of this. The Structural Risk Minimization (SVM) approach is aimed at reducing structural risk (SRM). The Input vector is projected into a higher-dimensional space, where a greatest separating hyperplane is assembled.

Data points of the form (x1,y1), (x2,y2), (x3,y3), (xn,yn) are regarded. Where yn=1 / -1 is a continuous representative

the class that xn belongs to. The no. of samples is denoted by the letter n. xn represents a p-dimensional real vector. Appropriate Scaling is use for avoiding variables (attributes) with higher variance. To see this Training info, use the dividing (or separating) hyperplane that has the formula-

$$w.x + b = 0 \quad (1)$$

In this equation w is a p-dimensional Vector and b is a scalar. The dividing hyperplane is vertical to the vector w. It can maximize the margin by using the offset parameter b. The hyperplane is required to travel by the origin in the absence of b, decreasing the solution. Researchers are interested in SVM and parallel hyperplanes in the comparable way as they are interested in the overall margin. Equation can be useful to characterize parallel hyperplanes equation-(x1,y1), (x2,y2), (x3,y3), (xn, yn) are all valid data points. The aim of appropriate scaling is to avoid variables (attributes) with a high level of variance. SVM and parallel hyperplanes pique our curiosity in the same way as the average margin does. Parallel hyperplanes equation can be defined using the equation:

$$w.x + b = 1 \quad (2)$$

$$w.x + b = -1 \quad (3)$$

If the training data are linearly divisible, these hyperplanes are created with no points among them, and then the distance between them is optimised. Corresponding to geometry, the interval among the hyperplanes is 2 / w. As a result, to keep w to a bare minimum. Kernel approaches and broad margin classifiers are combined in SVMs. SVM has also been applied in function compilation, time series analysis, turbulent structure reconstruction, and non-linear principal component analysis. In the near future, further research in these fields is needed. SVMs and methods based on them are becoming more common in real-world data mining. Figure 1 illustrates the maximum margin hyperplanes for a SVM trained with samples from 2 classes.

It can be pick by these hyperplanes because there are no points between them and it purpose to increase their distance if the training data are linearly divisible. The Maximum margin hyperplanes for an SVM trained with examples from 2 groups are shown in Figure 1. Also want to test that for every I, w. xi – b+ 1 / w. xi – b -1 would stimulate data points. It's possible to write it as:

$$y_i(w.x_i - b) >= 1, 1 <= i <= n \quad (4)$$

SVM Advantages

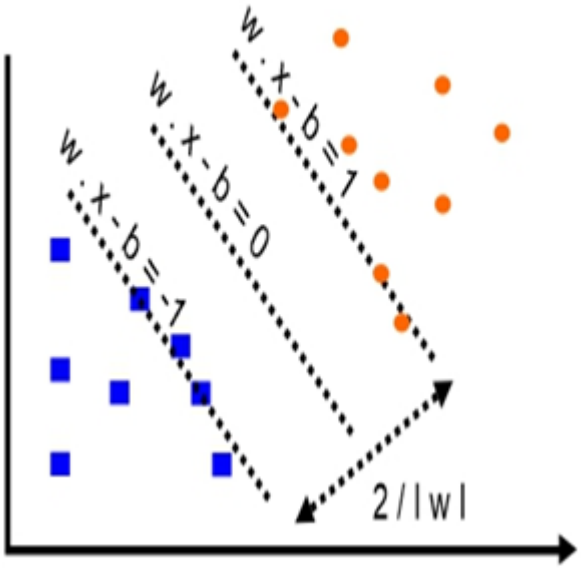

Figure 1. Maximum margin hyperplanes for a SVM trained with samples from 2 classes [17]

i SVM is not answered for local optima
ii It ranges fairly high for dimensional data
iii The risk of over-fitting is less in SVM
iv ANN and SVM are always compared. SVMs outperform ANN models in terms of accuracy

SVM Disadvantages

i Selecting a "good" kernel function is not easy
ii Lengthy training time for huge datasets
iii The variable weights, final model and individual effect are problematic to comprehend and explain

B. ANN

It is a high-performance computing device whose core theme is inspired through biological neural networks. Parallel distributed processing systems, connectionist systems and artificial neural systems are all terms utilized to explain ANNs. it collects a larger amount of units that are connected in several way to enable contact among them. These modules, also recognized as neurons or nodes, are elementary processors which work in a parallel manner [18].

A connection relation connects every neuron to another neuron. Every communication link is assigned a weight that provides information about the incoming signal. Because the weight generally motivates or hinders the signal from being sent, this is the highly valuable information for neurons in resolving a specific problem. Every neuron has an interior state called an activation signal [20]. The output signals formed through combining incoming signals with the activation rule can be sent to other units. .Several studies

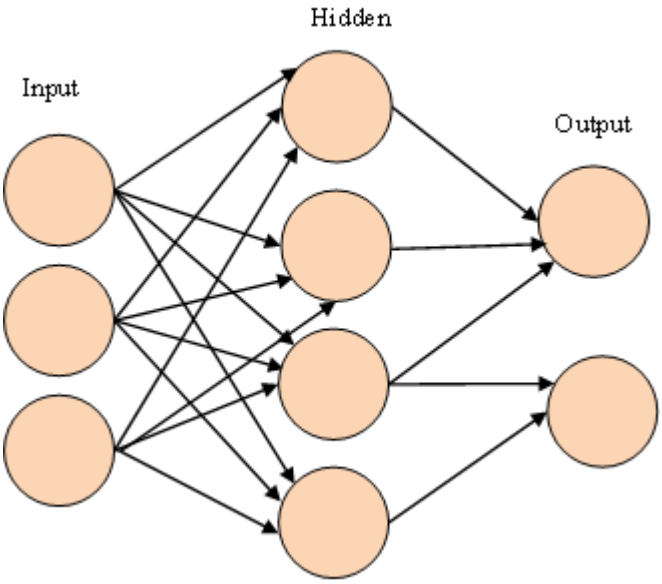

Figure 2. ANN

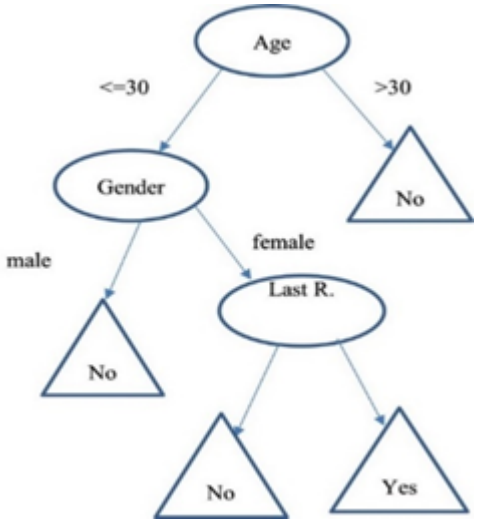

Figure 3. Decision Tree

have identified the use of models supplied by ANN in estimating the bearing capacity of driven piles.

It have seen a burst of enthusiasm in the last couple of years and they are now successfully linked across a wide variety of problem spaces, including finance, response, architecture, material science and topography. Statsoft.com is a website dedicated to statistics. It completely started in 1943, when Pitts and McCullock showed that a neuron can have 2 countries one of which is dependent on some kind of limit esteem.

The revelations of McCullock and Pitt paved the way for cunning devices. If the human brain could be as easy as a CPU desktop with the usage of the procedure if ANN operates in the same way as the human brain. Any human being will almost certainly have been similar at some point in the future. Any one of them should have made the identical choices in any of these cases. What can be the distinguishing factor accountable for such diversity among people is the individual concerns, which can behave similarly in different locations. What, in my opinion splits the human mind from a vehicle [19].

A modest computer with a series of algorithms that translates Input to Output. When a computer is complicated the similar inputs still get to go to the similar outputs. When a human brain is complicated the human mind produces several results simply since the presence of neurons and the scope of the brain causes it to respond differently to conditions.

Functioning of ANN

Let a topological system that is linked by arrows pointing right that signify a link between two neurons and it shows the information flow pathway. Every link has a weight, which is an integer number which signifies the symbol variation among the 2 neurons. If the network has a low or undesirable outcome and if there is a fault, the device adjusts the weights to maximize the resulting impact. If there are no faults, the no. of the weights related through nodes is played out, subsequent in an optimal explanation.

Advantages of ANN

i It have the capability to model non-linear and complex relationships
ii It can infer unseen relationships on unseen data as well, thus creating the model predict and generalize on unseen data
iii It doesn't require any limits on the input variables

Disadvantages of ANN

i Hardware dependence
ii Unexplained functioning of the network

C. Decision Tree

It is a looping division of case space that is used to categories outcomes. The nodes in the decision tree form a rooted tree, which is a directed tree with no incoming edges and a node called "root." There is one incoming tip for each of the other nodes. A node with outer edges is known as an internal or assessment node. The remaining nodes are mentioned to as leaves (called as terminal or decision nodes). A decision tree's interior nodes partition the example space into two or more sub-spaces based on a particular feature of the Input attribute values [20].

In the easiest and most basic example, each test includes a single attribute, and the case space is partitioned depending on the attribute's value. In the case of numeric

properties, the state is utilized to a continuum. A class is allocated to each leaf based on the best target worth. Alternatively, the leaf can store a probability vector representing the possibility that the target attribute would be persuaded. Instances are rated by leading them from the root of the tree down to a point based on the result of the valuations along the way.

Figure 3 illustrates a decision tree for deciding whether or not a potential customer should respond to a direct mailing [21]. Leaves are represented as triangles, while interior nodes are shown as squares. This decision tree has both trivial and integer properties, which is worth remembering. When it comes to direct mailing, this classifier can prediction a future customer's response and consider the behavioral characteristics in the whole community of potential customers. Any node is labelled with the attribute it's evaluating and every branches are labelled with the attribute's values.

Advantages of Decision Tree

i People are able to understand decision tree models after a short explanation
ii Valuable insights can be produced based on experts explaining a situation (its alternatives, probabilities, and costs) and their preferences for results
iii It can determine best, worst, expected values for different scenarios

Disadvantages of decision tree

i These are unstable
ii These are often inaccurate
iii Information gain in decision trees is biased in favor of those traits with preference levels

4. The Metrics Applied for Model Evaluation Correlation Coefficient

The association coefficient tests how well forecasts match the real grade. This metric demonstration how thoroughly real and expected values are related.

A. MAE

It is the amount that is dependent on the minimal error prediction for the individual subjects. MAE assesses the performance of each model for each observation to guarantee that the chosen model produces the most effective results and does not over or underestimate. The model's results by n experiments are the forecast and observed values.

B. RMSE

It is frequently utilized to calculate the variance among values predicted through a template and values actually found in modelled domain. Precision is calculated by dividing the number of properly expected types by the overall no. of grades. ROC stands for receiver operating characteristic curve, and it is a method of assessing the success of prediction models. The ROC curve is a graph that proves how well a graph of sensitivity on the organize vs its (1-specificity) on the x coordinate is referred to as a ROC curve.

TABLE I. Performance measurement of Dataset 1 at security level 2

Evaluation Parameter	SVM	Decision Tree
MAE	1.2921	1.3674
RMSE	1.8582	1.7222

TABLE II. Performance measurement of Dataset 2 at security level 2

Evaluation Parameter	SVM	Decision Tree
MAE	2.7957	2.3258
RMSE	3.0806	2.8859

TABLE III. Performance measurement of Dataset 2 at security level 2

Evaluation Parameter	SVM	Decision Tree
MAE	1.2921	3.3258
RMSE	1.8582	4.5103

As a result, the Area Under the ROC Curve (AUC) is a metric which combines compassion and precision. F-Measurement is a form of integrating memory and accuracy ratings into a single test of public speaking that looks like this: F-measure = 2 9 recall 9 accuracy; measure = 2 9 recall 9 exactness; F-measure = 2 9 recall 9 precision; F-measure.

C. Sensitivity

It is also known as recall rate because it is a measure of how accurately actual sites are remembered. It calculates the proportion of positive, which is fundamentally defined as true negatives' ratio to (number of true positives).

D. Linear Regression

Linear regression is not necessarily draw a simple line i.e. goes over each data point but gets close to the majority of them. As a consequence, the location signifies the expected average shift in X (testing period) and Y (failure rate) as rises through single unit in analysis. Table 1 and Table 2 illustrate the comparison between SVM and Decision Tree for Dataset 1 and 2 [22]. Table 3 demonstrates the Performance measurement of Dataset 2 at security level 2.

Data Set 1 and Data Set 2 were applied of public area which contains complex, minimum and maximum errors. The cruelty of faults is considered into sub-categories like severity1, severity2 dependent on the behavior of the practice signifying 1 the most severe or serious issue with 2 being the main complicated and 3 being a slight issue. DS1 is occupied from Misra (1983) contains of complex, big and small faults. DS2 is On-line Communication System (OCS) project at ABC Software Company (Pham 2003a).

TABLE IV. Different Software Reliability Models

S.No.	References	Author(s)	Technology used	Software Reliability Models (Advantages and Disadvantages)
1.	[23]	Katiyar et al.	Neuro Fuzzy	In terms of parameters, the suggested algorithm is resilient, and this is particularly well suited to very large-scale examples of the reliability issues
2.	[24]	C.Y. Huang et al.	Tabu Search	The ANFIS increases the FIS technique's reliability rating, according to the findings
3.	[25]	Y. Singh and P. Kumar	Stimulated Annealing	The suggested adaptive simulated Annealing was decreased from a population of solutions for the Remote Center AirGround (RCGA) to one solution for suggested Simulated Annealing, it outperformed the RCGA in terms of execution time
4.	[26]	Latha et al.	Ant Colony	Experiments with 3 common models reveal that this ant colony-based technique has a wide range of application
5.	[27]	Ding et al.	Neural Network	The Elman model is superior to the Jordan model and is quite powerful
6.	[23]	Nirvikar et al.	Neural Network	System compared to single NN and standard SRGMs, the system achieves much lower prediction error
7.	[28]	Singh et al.	Neural Network	Models with a smaller normalized RMS of error than regression model
8.	[29]	Sultan et al.	Neural Network	Elman recurrent NNs are a reliable method for function forecast because they capture the data set's dynamic nature
9.	[30]	Aliahdali et al.	Fuzzy Logic	Models that have been developed provide high-performance modelling capabilities
10.	[31]	Kaswan et al.	Fuzzy Logic	The established model can accurately anticipate outcomes in the majority of the target database's points
11.	[32]	El-Telbany et al.	Genetic Algorithm	The reliability of software was measured utilizing a group of models, that worked better than an individual model and the weighted average mixing approach for band performed better than the average method
12.	[33]	Sandeep and Manu	Neural Network	In addition, the faults forecast behaviour in the set of components was calculated over a cumulative execution time period, and the faults prediction for the entire software was estimated
13.	[34]	Satyaprasad et al.	Genetic Algorithm	This model is evaluated as superior to the others when all conditions are considered
14.	[35]	Lakshamanan	DS-1 John Musa	Suggested a good technique for finding an equation to model software reliability and for figuring out which equation best represents the facts
15.	[31]	Kuldeep et al.	Ant Colony	The Enhanced Ant Colony Optimization Method outperforms the existing ACO method in terms of estimation accuracy. Complexity of time and space is also decreased
16.	[36]	Naila et al.	Cuckoo Search	Better parameters tested using identical datasets, CS outperforms both PSO and ACO

It includes of a unit-manager, single operator interface software programmer and 10 software engineers/testers. Table 4 demonstrates the Software reliability Models.

5. Result Analysis

Following points shows the analysis of the result:

1) A main benefit of SVMs is that while ANNs will finally suffer the negative consequences of neighborhoods minima, the result for an SVM is global and exclusive
2) SVMS has two additional preferences: they offered a simple geometric translation and contributed a sparse structure. The computational numerous-sided credentials of SVMs, unlike ANNs that have a simple geometric considerate and can be implemented to a sparse arrangement
3) It differ significantly from related methodologies. SVM is planning to reliably locate a global minimum and its modest geometric translation delivers fertile ground for additional study"
4) Gaussain pieces are often used and the SVM solution understands the system's multifaceted structure as an outcome. If the aftereffect of the hidden bolster and neurons vectors corresponds to each other, the scale of the shrouded layer is increased and the inside problem of the RBF mechanism is compact like a bolster vector satisfies in as the principle task emphases.
5) Unlike traditional multilayer discernment neural models which embrace the ominous consequences of the existence of neighborhoods minima system and the nonlinear SVM classifier's convexity is a vibrant and striking property.
6) The absence of neighborhoods minima in the SVM approximation shows a major departure from traditional fabrics like the neural structures. Table 4 shows the Different Software Reliability models.

6. Conclusion and Future Work

The most of software stability technique have been verified and also, these are still demanding to apply in applications. In the present time a major of consistency work has been suggested for measuring Software reliability. Corresponding to a study of the literature, the major task has been undertaken at the scripting and testing phases of the software growth life cycle. As an outcome, the development of software reliability to create stable systems remains unresolved, and system customers are unhappy. One of the pitfalls of the growth of stable applications is software difficulty. In an approaching review paper, a new combined paradigm for Software Reliability Estimation Model will be planned with the goal of preserving correctness and difficulty. In the early stages of software development, reliability estimate is a mathematical process for predicting and estimating dependability. The capacity to measure and anticipate reliability is critical in determining the position of the software business in the market. The software developers must choose the correct software prediction model and reliability growth for the growth of the product.

References

[1] C. Diwaker, P. Tomar, R. C. Poonia, and V. Singh, "Prediction of software reliability using bio inspired soft computing techniques," vol. 42, no. 5. Springer, 2018, pp. 1–16.

[2] D. Kumar, Y. Kansal, and P. Kapur, "Integrating neural networks with software reliability," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2016, pp. 4072–4077.

[3] J. Lv, B.-B. Yin, and K.-Y. Cai, "On the asymptotic behavior of adaptive testing strategy for software reliability assessment," *IEEE transactions on Software Engineering*, vol. 40, no. 4, pp. 396–412, 2014.

[4] R. K. Behera, S. Shukla, S. K. Rath, and S. Misra, "Software reliability assessment using machine learning technique," in *International Conference on Computational Science and Its Applications*. Springer, 2018, pp. 403–411.

[5] I. H. Chang, H. Pham, S. W. Lee, and K. Y. Song, "A testing-coverage software reliability model with the uncertainty of operating environments," *International Journal of Systems Science: Operations & Logistics*, vol. 1, no. 4, pp. 220–227, 2014.

[6] D. Cotroneo, R. Pietrantuono, and S. Russo, "Relai testing: a technique to assess and improve software reliability," *IEEE Transactions on Software Engineering*, vol. 42, no. 5, pp. 452–475, 2015.

[7] R. Pietrantuono and S. Russo, "On adaptive sampling-based testing for software reliability assessment," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2016, pp. 1–11.

[8] K. Sheoran, P. Tomar, and R. Mishra, "Software quality prediction model with the aid of advanced neural network with hcs," *Procedia Computer Science*, vol. 92, pp. 418–424, 2016.

[9] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, vol. 27, pp. 504–518, 2015.

[10] D. Sudharson and D. Prabha, "A novel machine learning approach for software reliability growth modelling with pareto distribution function," *Soft Computing*, vol. 23, no. 18, pp. 8379–8387, 2019.

[11] S. Das, R. K. Behera, S. K. Rath *et al.*, "Real-time sentiment analysis of twitter streaming data for stock prediction," *Procedia computer science*, vol. 132, pp. 956–964, 2018.

[12] Y. Tamura and S. Yamada, "Fault identification and reliability assessment tool based on deep learning for fault big data," *Software Networking*, vol. 2018, no. 1, pp. 161–167, 2018.

[13] R. Rajasekhar and A. Murty, "Application of machine learning techniques on failure data to estimate parameters of reliability," *International Journal of Pure and Applied Mathematics*, vol. 115, no. 8, pp. 157–162, 2017.

[14] R. Malhotra, S. Shukla, and G. Sawhney, "Assessment of defect prediction models using machine learning techniques for object-oriented systems," in *2016 5th International Conference on Relia-*

bility, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO). IEEE, 2016, pp. 577–583.

[15] M. K. Bhuyan, D. P. Mohapatra, and S. Sethi, "Software reliability prediction using fuzzy min-max algorithm and recurrent neural network approach." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 6, no. 4, 2016.

[16] P. Kumar and Y. Singh, "Comparative analysis of software reliability predictions using statistical and machine learning methods," *International Journal of Intelligent Systems Technologies and Applications*, vol. 12, no. 3-4, pp. 230–253, 2013.

[17] A. Jindal, A. Dua, K. Kaur, M. Singh, N. Kumar, and S. Mishra, "Decision tree and svm-based data analytics for theft detection in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1005–1016, 2016.

[18] H. Bhavsar and M. H. Panchal, "A review on support vector machine for data classification," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 10, pp. 185–189, 2012.

[19] H. Moayedi and D. J. Armaghani, "Optimizing an ann model with ica for estimating bearing capacity of driven pile in cohesionless soil," *Engineering with Computers*, vol. 34, no. 2, pp. 347–356, 2018.

[20] S. Sathyadevan and R. R. Nair, "Comparative analysis of decision tree algorithms: Id3, c4. 5 and random forest," in *Computational intelligence in data mining-volume 1*. Springer, 2015, pp. 549–562.

[21] Z.-T. Liu, M. Wu, W.-H. Cao, J.-W. Mao, J.-P. Xu, and G.-Z. Tan, "Speech emotion recognition based on feature selection and extreme learning machine decision tree," *Neurocomputing*, vol. 273, pp. 271–280, 2018.

[22] P. Waldmann, "On the use of the pearson correlation coefficient for model evaluation in genome-wide prediction," *Frontiers in genetics*, vol. 10, p. 899, 2019.

[23] N. Katiyar and R. Singh, "Effect of neural network for prediction of software reliability," *VSRD-IJCSIT*, vol. 1, pp. 490–500, 2011.

[24] C.-Y. Huang and M. R. Lyu, "Estimation and analysis of some generalized multiple change-point software reliability models," *IEEE Transactions on reliability*, vol. 60, no. 2, pp. 498–514, 2011.

[25] Y. Singh and P. Kumar, "Prediction of software reliability using feed forward neural networks," in *2010 International Conference on Computational Intelligence and Software Engineering*. IEEE, 2010, pp. 1–5.

[26] L. Shanmugam and L. Florence, "Enhancement and comparison of ant colony optimization for software reliability models," 2013.

[27] S. Ding, Y. Zhang, J. Chen, and W. Jia, "Research on using genetic algorithms to optimize elman neural networks," *Neural Computing and Applications*, vol. 23, no. 2, pp. 293–297, 2013.

[28] Y. Singh and P. Kumar, "Application of feed-forward neural networks for software reliability prediction," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 5, pp. 1–6, 2010.

[29] S. H. Aljahdali and K. A. Buragga, "Employing four anns paradigms for software reliability prediction: an analytical study," *ICGST-AIML Journal, ISSN*, pp. 1687–4846, 2008.

[30] S. Aljahdali, "Development of software reliability growth models for industrial applications using fuzzy logic," *Journal of Computer Science*, vol. 7, no. 10, p. 1574, 2011.

[31] K. S. Kaswan, S. Choudhary, and K. Sharma, "Software reliability modeling using soft computing techniques: Critical review," *Journal of Information Technology and Software Engineering*, vol. 5, p. 144, 2015.

[32] S. H. Aljahdali and M. E. El-Telbany, "Genetic algorithms for optimizing ensemble of models in software reliability prediction," *ICGST-AIML J*, vol. 8, no. 1, pp. 5–13, 2008.

[33] S. K. Jain and M. P. Singh, "Estimation for faults prediction from component based software design using feed forward neural networks," *IJARCCE*, 2013.

[34] R. SatyaPrasad, O. NagaRaju, and R. LKantam, "Srgm with imperfect debugging by genetic algorithms," *International Journal of software engineering & applications (IJSEA)*, vol. 11, no. 2, 2010.

[35] I. Lakshmanan and S. Ramasamy, "An artificial neural-network approach to software reliability growth modeling," *Procedia Computer Science*, vol. 57, pp. 695–702, 2015.

[36] D. AL-Saati, N. Akram, and M. Abd-AlKareem, "The use of cuckoo search in estimating the parameters of software reliability growth models," *arXiv preprint arXiv:1307.6023*, 2013.

Ms. Poorva Sanjay Sabnis Ms. Poorva Sanjay Sabnis is a research scholar at Computer Engineering Department of Bharati Vidyapeeth Deemed to be University College of Engineering, Pune. She has completed Masters of Technology in Computer Engineering in the year 2014. Her research interests are Software Engineering, Software Reliability and Machine Learning.

Dr. S. D. Joshi Dr. S. D. Joshi is working as Dean of Faculty of Engineering and Technology at Bharati Vidyapeeth Deemed to be University College of Engineering, Pune. He is experienced Professor with a demonstrated history of working in the IT industry. His research area is Software Engineering and Science, Software Security, Data Engineering. He is a strong Engineering professional with a Doctor of Philosophy (Ph.D.) focused in Computer Engineering from Bharati Vidyapeeth.