Journal of Integrated
SCIENCE & TECHNOLOGY

Software reliability prediction and optimization using machine learning algorithms: A review

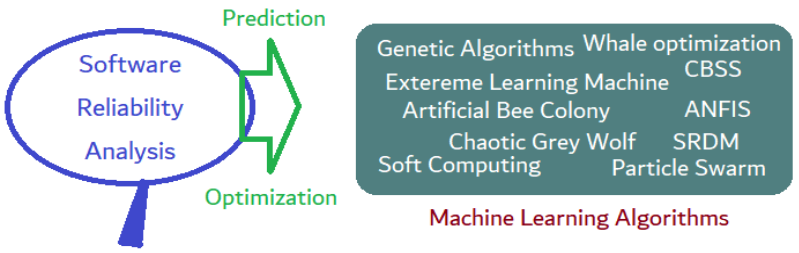
Neha Yadav[1,2*], Vibhash Yadav[3]

[1]KIET Group of Institutions, Delhi NCR, Ghaziabad, India. [2]Dr. A.P.J. Abdul Kalam Technical University, Lucknow, India. [3]Rajkiya Engineering College, Banda, India

Submitted on: 06-Oct-2022; Accepted and Published on: 01-Dec-2022

ABSTRACT

Software reliability is an important part while evaluating software quality. Many challenges are faced while developing highly reliable software such as its usability, performance, service, and maintenance, etc. Prediction and optimization of reliability estimation procedure is performed by optimizing parameters of the model. Several traditional models are existing to evaluate the reliability of the model, but it is quite difficult to directly estimate optimal parameters. Therefore, researchers adopted intelligent prediction and optimization algorithms for software reliability check. But still there a lot of limitations that needed to be focused and solved. In this paper, a detailed study is presented for software reliability prediction using machine learning. The paper also presents an analytical analysis for software reliability prediction.



Keywords: Software Reliability; Quality; Intelligent Prediction; Machine Learning.

INTRODUCTION

Generally, in technical terms, reliability is the probability. It guarantees that a product or system in a defined environment will function properly for a set amount of time. As computer programs pervade every aspect of modern life, each breakdown of those programs has an influence on human beings. The production of high-quality software systems that meet user expectations is a critical challenge in the development of such software systems. Developers seek to evaluate the reliability of their software as part of software engineering process and review the actual degree of reliability with the product's previous record. If a software gives reliable performance, the system experiences less failure with time.[1]

This distribution is described by software reliability models that have been modified to data from a software development process. When a strategy has proven a strong match to the data, it could be used to determine the software's real dependability and predict future dependability. The issue was whether software applications has became sufficiently sophisticated and computer programmers are not any more able to adequately test the program to ensure that it functions correctly. These could be owing to assertions implemented by different software reliability theories, or it could have been due to the interdependence of subsequent programmed executions. The extent whereby the project's inner structure has indeed been influenced, as well as the type of the activities done for executing continuation, impact the probabilistic dependence of following existing systems.[2]

Recognizing techniques or linkages that can be utilized to assess the value of software products more precisely while visiting a substantial percentage of their conceivable states to address these problems. Considering the connections between the breakdowns. However, in some implementations, no approach is sufficiently reliable.

Most techniques to modelling software dependence and reliability, such as parametric framework, non-linear time series analysis, and information retrieval, have lately been investigated.[3,4] Several studies indicate to the use of computer vision advances to aid human programmers in the design of software systems by managing various forms of unpredictability seen in software

*Corresponding author: Neha Yadav, KIET Group of Institutions, Ghaziabad, India
Email: nehayad564@gmail.com

Cite as: *J. Integr. Sci. Technol., 2023, 11(1), 457. URN:NBN:sciencein.jist.2023.v11.457*

systems. In modelling intricate models and assist on taking decision and predicting things in dynamic settings, because of the infrequent and unanticipated development of errors and not sufficient data or erroneous data. To attain robustness, predictive validity, and lower costs, these AI methods are increasing collections of methodologies that allow error, ambiguity, and half-truth. Fuzzy logic, NN, GA (Genetic algorithm), genetic programming, are one of the most important core methods.

In most cases, the dependability prediction method is divided into two parts. The first portion is referred to as the training phase, while the second is referred to as the prediction stage. The prediction model is built in the initial step of training, and it uses methods level or class level software metrics with fault information connected with all the components of software programmes. Subsequently, the very same approach is used to anticipate fault proneness in the following version of the software. By utilising metrics associated with fault data, classification techniques are used to designate classes as fault-free or faulty. Fault prediction models are used to locate defective classes in software, which improves software quality. The model methodology,[5] as well as metrics,[6] are influenced by the model performance. Many academics have developed and approved ML and statistical approaches for improving the performance of dependability prediction models using datasets, metrics, and feature reduction methodologies. The software quality is improved by locating faulty.

Software Defect Prediction

In this paper, software defect predictions are presented using machine learning and optimization approaches. For this a systematic critical analysis is presented, as presented in figure 1.

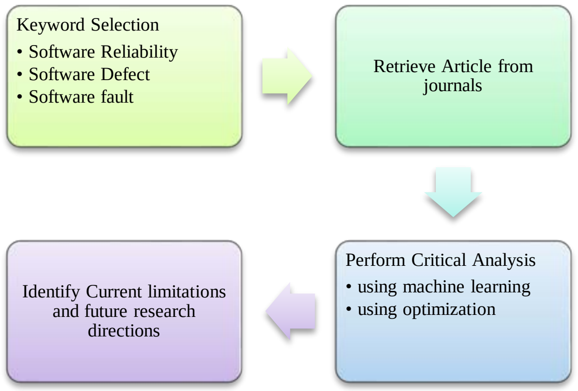


Figure 1. Approach for Systematic Meta-Analysis

Unintentionally, software flaws are created throughout the coding process of programmers. One of the most efficient ways to solve this problem is to use feature selection. Inspired by the notion of IT based engineering, Xiang Chen et al.,[1] formalised the challenge as a multi-objective optimization problem, proposes an unique technique MOFES, and compares MOFES to various traditional baseline approaches using the PROMISE dataset obtained from actual projects. The final findings demonstrate that the technique has the benefit of picking fewer features and providing higher prediction performance in major implementations, all while having a reasonable and effective cost. Performance Prediction Graphically, several techniques for PROMISE datasets are compared.

Ensemble methods have been established by Kiran et al.[2] to properly anticipate software dependability. The ensembles given are made up of a variety of statistical (multiple linear regression and multivariate adaptive regression splines) and intelligent approaches (backpropagation trained neural network, dynamic evolving neuro–fuzzy inference system, and TreeNet). Three linear and one nonlinear ensembles were created and tested. The non-linear ensemble outperforms all previous ensembles, as well as the constituent statistical and intelligent approaches, according to studies conducted using software reliability data gathered from the literature.

Cong et al.[3] proposed a hybrid IEDA-SVR model is suggested. The chaotic mutation IEDA-SVR is being used to forecast software dependability in order to sustain population diversity. In the trials, there were two genuine software failure datasets. The suggested model's prediction performance was compared to that of other models. The experimental findings show that using IEDA-SVR to forecast software dependability is extremely successful, and IEDA-SVR has a superior prediction performance than the other comparison methods, as well as a pretty accurate prediction capabilities. Research Finding also suggest that maintaining population variety might help the prediction model perform better. The mean square absolute error is 0.0848, IEDA-SVR R value is 0.918 , mean square value is 0.201 which is least when compared with EDA-SVR and norman and kalman filter

An on-line adaptable software reliability predicting framework is described using an evolving connectionist methodology based on multiple-delayed-input format.[4] Studies showed that its proposed method performs admirably the original NN model for cumulative failure time prediction in aspects of predictions along a wide wide variety of computer initiatives. The highest results of the prediction are 95 percent. Based on already known software failure time data, a genetic algorithm is utilized to maximize the number of delayed input neurons and neurons in the hidden layer of the NN architecture.

Kassaymeh et al.[5] coupled the SALP swarm technique (SSA) with a back - propagation NN to tackle software effort prediction (SEP) and software test prediction. BPNN is by far the most widely utilized forecasting technique. Model parameter changes, including bias and weights weight have a significant impact on the effectiveness of BPNN. The findings demonstrate that SSA-BPNN outperforms BPNN across every samples. BPNN has an R2 of 0.996,MAE of 0.0360, an RMSE of 0.1907, a RAE in percent of 2.4, and RRSE of 9.7 percent.

Zhen et al.[6] predicted software reliability model using a hybrid technique of WPA and PSO. To predict the values of the GO model and generate predictions, five types of data from industry were employed. Each algorithm runs 20 times and picks the best results after 500 iterations. When the algorithms are performed 20 times, WPA- PSO has the lowest error rate among the five data sets.

Roy et al.[7] offered a software reliability model based on an artificial neural network (ANN) and trained using an unique particle swarm optimization (PSO) technique for improved software reliability predictions. The suggested ANN is based on the fault generation phenomena that occurs during software testing at various levels of fault complexity. The suggested approach takes into account three different forms of software flaws. Using software failure data, a neighborhood-based fuzzy PSO method for competent learning of the suggested ANN was developed. The neighbourhood fuzzy PSO-based proposed neural network model's fitting and prediction performances are compared to the traditional PSO-based proposed Neural Network model. The results shows that the model allows the faster release of software. The model allowed the uncertainty in prediction.

The software industry's primary goal is stakeholder satisfaction. Producer and consumer satisfaction refers to delivering a high-quality software product. Gheisari et al.[8] presented a novel optimum mathematical model for predicting stakeholder satisfaction levels (Q). The relationship implications of various quality parameters are used to validate the real data in optimal models. It employs constraint equations. The maximum and lowest values for Q are determined by the optimal model. Constraints are aspects of software quality. It establishes that the given result is the best option. The result demonstrates that if the value of any software quality feature changes, the value of Q decreases. It establishes that the supplied outcome is the best option.

A novel paradigm for software reliability is proposed by Sedlacek et al.[9] Although there are numerous software reliability models, the majority of them cannot be utilized for system component analysis, such as calculating significance measures. As a result, a new model was presented. This model is built from source code, which is then utilized to build a syntax tree in the next phase. A syntax tree is a hierarchical representation of source code that is unrelated of the computer language used. This tree is then converted into a structural function and utilized to construct a reliability model, in this instance a fault tree. The ability to employ standard methods for system assessment, like logic differential calculus, is the major benefit of the proposed approach. However, there are a few drawbacks to this design, such as its high dimensionality. The Reliability calculated form the syntax model is 0.7567 and the unreliability calculated from the probability is 0.245.

The modified differential evolution (MDE) technique is proposed by Tahere et al.[10] for tackling a nonlinear optimization techniques. The problem is determining the constraints of a NHPP-SR model using maximum likelihood estimation (MLE). DE has two modifications: one is a mutation strategy came up with a new linear mixture of multiple atleast 3 points for boosting the algorithm's exploration capacity, and another is a uniform scaling crossover technique for improving the algorithm's exploitation ability. The suggested scheme's effectiveness is tested empirically using three software failure datasets and five software reliability models. The MDE's effectiveness were verified on the collection of 15 test scenarios, and the quantitative results have been compared to the fundamental DE and two additional comparable techniques. In contrast to the basic DE, the suggested algorithm improved the convergence speed by 53%, according to the research data. For something like the test suites included in this work, the suggested method also obtained a correct AR. In contrast to Laplace DE and RGA, the MDE exhibited a 57 percent and 43 percent improvement, respectively, and produced appropriate ARs. The findings' robustness was demonstrated by sensitivity analysis tests.

The CGWO heuristic is proposed as a novel way to quantifying SRGM characteristics by Dhavakumar et al.[11] The suggested approach outsmarts a number of existing techniques' obstacles. Using the parameter estimation approach, datasets were used to obtain the findings of the assessment criteria. The results demonstrate that the suggested technique reduces prediction error, relies on data properties rather than assumptions, has an acceptable forecasting capability, and the entire prediction process is automated with no user involvement. The Chebyshev graph has a decent convergence rate of 78 percent based on the chaotic graph data. In all, 86 percent of the data indicated a link between the choice variable and the CGWO ftness criterion. The intended outcome in the SRGM utilising CGWO is fully automated and does not require any customer involvement.

Optimized the cost and release time by predicting release time, technique used Optimization. Techniques based on software dependability models are used to achieve the best possible release time. Prashant et al.[12] proposed that both the original and anticipated release times be used to complete the task, resulting in an optimal release time cost. This will aid in determining the software's effective cost of dependability. Further, this will assist the client in making a more informed decision when selecting effective software. Thus, using Python, we were able to optimise the cost and determine the product's release time, allowing us to determine that this software will produce the best results.

The optimal test activity allocation issue is formulated by Vidhyashree et al.[13] to optimise fault discovery despite budget restrictions or to reduce the budget necessary to find a certain no. of defects. 2-data sets were used to develop and test expectation conditional maximization methods. The optimal test activity distribution fault was then addressed to show how dividing limited testing resources across the testing activities conducted might enhance the number of defects found. 40.64 is the predicted value the best allocation for exposing three more problems. Asymptotically. For the three covariates, the effort dedicated to the variables converges little than 60, 40, and 10% of the total effort.
The dependability of software for lifetime distributions based on non-homogeneous Poisson processes is explored by Kim et al.[14] The exponential distribution, that is usually utilized in different fields of software reliability, and the inverse-exponential distribution, which is frequently applied in the economic and environmental field, as well as the Burr-Hatke-exponential distribution, which decreases the hazard function, were used as lifetime distributions. The exponential distribution model has a lower mean square error than the inverse-exponential distribution model and the Burr-Hatke-exponential distribution in this investigation. As the predicted value for coefficient of determination is 95 percent, the Burr–Hatke-exponential model can be considered an efficient model in terms of goodness-of-fit. All models are considered efficient models if the estimated value for coefficient of determination is 95 percent or more. The software failure occurs at the final test failure time of x27 =5.529 in the NHPP model, and reliability is the chance that the software failure does not occur between 5.529 and 5.529 t+1. when the mission time

passes, the non-increasing pattern emerges in the form of the dependability function, and the inverse-exponential distribution model outperforms the Burr-Hatke-exponential distribution model and the exponential distribution model.

Li[15] presented an extended software reliability model that takes into account the unpredictability of field settings. NHPP is the basis behind this. Several current models are evaluated as special instances of the generalized model based on the general framework, and a novel model by 3-parameter S-shaped fault-detection rate and a random environmental component following the Weibull distribution is presented.

To forecast software dependability, the suggested fuzzy neural approach was implemented. When using bell labs' time to failure data, several parameters like as mean time to failure, duration to failure, and so on, may be used to forecast reliability. Sahu et al.[16] employed a hybrid fuzzy logic and neural network technique. In this technique, we used algorithm-based fuzzy methodology to implement data of dependability, and the fuzzy output was then applied to the MATLAB tool of neural network. We also utilized a neural network approach called the Levenberg Marquardt algorithm to forecast dependability. The average normalized RMSE error was calculated to assess the effectiveness of the reliability prediction model. We discover that the fuzzy-neural approach has the lowest error among the four offered ways, with 0.0546 as its useful better result, demonstrating supremacy over the other methods.

The Ant Colony Optimization (ACO), Neural Network (NN), Artificial Neural, Svm Classifiers (SVM), Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Artificial Bee Colony (ABC) were all considered as important soft computing approaches by Diwaker et al.[17] Their article explains how soft computing approaches operate and how to analyses them in order to forecast reliability. The parameters that are taken into account while estimating and predicting dependability are also examined. This study might be used to predict and evaluate the dependability of a variety of medical equipment, as well as digital, technology, and fluid mechanics and engineering.

Using ANFIS, a reliability estimate model for CBSS was presented by Dubey et al.[18] The model took into account the most important parameters that influence CBSS dependability. In ANFIS, a hybrid NN was employed, which was trained using data sets. This FIS-based NN guided regulation. This model relied on ANFIS's adaptive learning and decision-making capabilities. A Mamdani FIS assessment was also proposed. The ANFIS model was shown to be more efficient than the FIS model in determining the consistency of a CBSS. Future studies might involve developing a model that takes into account the elements mentioned in CBS. Internal, external, and development process aspects will all be considered.

For SRP logistic, Tong et al.[19] proposed chaotic time series and heterogeneous ensemble learning (HEEL). To find out if the failure data was chaotic, this method used chaos identification. Then, it used a large number of insufficient learners to train the model HEEL. In the end, a forecast was generated by running a trained model on the data. SRGMs and data-driven models were compared to see how well they predicted outcomes. After analysing the data, author found that the recommended approach had the best predictability and performance. This method uses MSE as a fitness function.

Li et al.[20] designed the hybrid algorithm using Ant bee colony-Particle swarm optimization (ABC-PSO) for estimation and predicate based on software reliability model. The model removes the unwanted solution in algorithm. The model was fast in operation but leads to low accuracy.

Lu et al.[21] proposed whale optimization model that can easily handle non-linear data. The three-stage optimization model was proposed and achieved better reliability but results in low accuracy.

P. R Bal et al.[22] proposed an extreme learning Machine ELM algorithm which gives an error rate of approximately 0.05

S. H. Aljahdali et al.[23] investigated Genetic Algorithms (GA) as a possible substitute to forecasting software reliability increase. GA is a strong ML approach that combines optimizations with machine learning to predict the values of very well estimation methods. The predictability of program dependability was assessed using an ensemble of GA-trained models. The R^2 single model, average ensemble and weighted average ensemble is 0.97, 0.98 and 1 respectively.

K Bithan et al.[24] proposed a Particle Swarm Optimization based reliability algorithm, a Swarm Intelligence-based stochastic search technique, was used in this work to evaluate growth models, resulting in better and optimized outcomes as well as attempting to avoid issues that can arise when estimating software reliability growth data set utilizing conventional methods. The NHPP-based Reliability Growth Model would be estimated using Particle Swarm Optimization with certain changes. Using PSO with certain changes, investigators were capable of predicting faults that were much closer to the real faults for both datasets, that further assisted in reaching faster convergence. The result shows that the iterations is reduced to almost 50%.

Sangeet et al.[25] proposed a new technique for optimising model parameters built on the idea of ecological space, differential evolution (DE), and intelligent behaviour of artificial bee colony (ABC). The paradigm of ecological space has boosted the exploring potential of the ABC algorithm. Four standard failure datasets were used to assess the hypothesized technique. Simulation findings demonstrates that the suggested holistic model is efficient in estimating software reliability and is comparable among conceptual optimization strategies. The proposed approach ABCDE algorithm calculates overall system reliability to be up to 85 %.

Ahere et al.[26] used a basic differential algorithm for optimization of software parameters. As compared to the basic DE, the proposed algorithm improved the convergence speed by 53%, according to research findings. For the test suites addressed in this work, the proposed algorithm also produced a valid AR.

P. Kumar et al.[27] proposed a feedforward NN based back -propagation algorithm method to enhance the precision of software reliability. Assessments of performance using a database of genuine evaluation issues reveal that the suggested prediction model outperforms standard methodology. The goal of this research is to look at the software's reliability utilizing soft computing techniques, that are the efficient way to assess its predictive capability. The SRGM (Software Reliability Growth Model) is employed. Result shows that the Linear regression has a maximum value of 16.60. The minimal FFNN values for Relative standard Deviation and Standard Deviation are 0.26 and 3.20, respectively, whereas the RSD linear regression is 0.41 and the Standard deviation result is 6.55.

Dhavankumar et al.[28] proposed the CGWO heuristic as a new technique to quantifying SRGM properties in this research. The number of failures after optimization has good convergence fitness, according to findings. The CGWO Chaotic grey wolf algorithm modification reduces errors by even more then 70%, while GWO only reduces errors by 40% and PSO only reduces failures by 25%. The Chebyshev graph reveals a 78.0% convergence rate.

Diwakar et al.[29] explained how soft computing approaches function and how to examine them in order to forecast reliability. The parameters that are taken into account when evaluating and predicting reliability were all addressed. This research can be used to estimate and anticipate the reliability of numerous devices utilized.

L Yang et al.[30] proposed a Particle Swarm Optimization algorithm for software reliability. The benefits and drawbacks of the swarm intelligence algorithm, as well as enhancement ideas, were discussed in this work.
Critical analysis of these works is presented in table 1.

Table 1. Critical Analysis of Approaches Used

Ref	Year	Method	Discussions
[1]	2017	MOFES, PROMISE	Several Techniques For PROMISE Datasets Are Compared. Performance Prediction. Reasonable Computational Cost.
[3]	2014	Hybrid IEDA-SVR model,	R^2 value is 0.9179, Mean Square Error was 0.201 and mean square absolute error is 0.0848.
[4]	2005	Genetic algorithm	Next-step-predictability is maximum at 95%.
[5]	2021	SALP Swarm method (SSA), Software Effort Prediction (SEP), BPNN	BPNN has 0.99531 R^2, MAE is 0.036, RMSE is 0.19069, RAE in % is 2.39 and RRSE is 9.72 %.
[6]	2020	WPA-PSO	Hybrid algorithm outperforms having accuracy, optimization performance, prediction accuracy, and algorithm stability PSO has the lowest error rate
[7]	2019	ANN, (PSO	Faster release of software Allowed uncertainty in prediction.
[9]	2021	Syntax Tree	The Reliability calculated is 0.7567
[10]	2020	Modified Differential Evolution (MDE), Non-Homogeneous Poisson Process (NHPP)	Improved the convergence speed by 53%.
[11]	2021	CGWO heuristic	Does not require any customer involvement. Chebyshev graph has a decent convergence rate of 78 percent
[12]	2019	Optimization. Techniques based on software reliability models	Effective cost Determined the product's release time.
[14]	2020	Non-homogeneous Poisson processes, exponential distribution	Lower mean square error. Coefficient of determination is 95%
[15]	2019	NHPP	MSE is 171.1531, AIC is 280.1920, and PP is 0.0517, $R^2 = 0.9974$.
[16]	2018	Fuzzy neural approach	Lowest error around 0.0546.
[18]	2017	CBSS, ANFIS	Determined the consistency. More efficient than the FIS model.
[19]	2017	SRP based on HEEL	RMSE and average relative error. MSE is used as the fitness function. Most effective. Good predicting and performance.
[20]	2019	Ant bee colony-Particle swarm optimization (ABC-PSO)	The model removes the unwanted solution in algorithm. The model was fast in operation but leads to low accuracy.
[21]	2018	Whale Optimization	The model easily handles non-linear data. Achieved better reliability.
[22]	2020	Extreme Learning Machine (ELM)	The error rate was approx. 0.05.
[23]	2009	Genetic Algorithms (GA)	The R^2 single model, average ensemble and weighted average ensemble is 0.97, 0.98 and 1 respectively.
[24]	2014	Particle Swarm Optimization	The result shows that the iterations is reduced to almost 50%
[25]	2020	Artificial Bee Colony (ABC).	The proposed approach ABCDE algorithm calculates overall system reliability to be up to 85 %.
[26]	2020	Differential Algorithm	Improved convergence speed by 53%
[27]	2021	SRDM (Software Reliability Growth Model)	RDS and SD are 0.26 and 3.20, respectively, whereas the RSD linear regression is 0.41 and SD result is 6.55.
[28]	2021	Chaotic grey wolf algorithm	reduces errors by even more then 70%.
[29]	2018	Soft computing Methods	PSA, Grey wolf, SALP, genetic algorithm is discussed
[30]	2021	Particle Swarm Optimization	Fitness function is introduced to increase reliability rate at MSE

Reliability Prediction Techniques

Countless investigations have been conducted on Reliability Prediction. Machine learning and statistical techniques are among the methods advised for predicting software flaws. Before the actual testing process starts, software reliability prediction attempts to detect fault-prone software components by utilizing some fundamental features of the software program. It aids in achieving target software quality while minimizing cost and effort. In the realm of software development, various prediction techniques are included, including forecasting of effort, privacy, quality, defect, cost, and re-usability using Swarm evolutionary algorithms,[31] Parameter Estimation,[32] whale optimization,[33] ant Colony,[35] apart from them many unified algorithm,[36] swarm particle optimization Algorithms[33,37,38] Optimization based[39] and Hybrid wolf algorithms.[40] All of these prediction techniques are still in their infancy. Experimentation and investigation are being carried out in order to develop a robust model. Software Reliability Prediction (SRP) is the process of developing a model that software practitioners may use to discover defective classes/modules prior to the testing phase.

Usually, machine learning is concerned with the creation and computer aided design using methods. These are used to extract patterns in the data from large datasets. Historically, neural networks (NN) were used in programming to construct system integration model for predicting total alteration or re - usability measures. Instead of developing formulae or rules, the NN model is trained to repeat a specified set of precise instance classifications. To handle faulted classes, the Multilayer Perceptron (MLP) is utilized. Radial Base techniques are used to categorize defects according to different fault types.

Various statistical techniques are used to identify a basic straightforward mathematical arithmetic expression that clearly identifies how categorization would be performed. Univariate binary logistic regression and logistic regression are two statistics techniques. Both methods are beneficial for looking into data containing binary variables. In BI (Bayesian inference), the model technique tries to correlate measurements with software faults and fault propensity.

Research Contributions for Software Reliability Prediction using Optimization

The exponential software reliability model is generalized in study presented by Rani and Mahapatra[7] to quantify numerous aspects, particularly fault initiation and time-varying fault diagnosis frequency. The software lifetime is built on module design, including testing effort expended in testing phase and found software defects, among other things. The allocation of resources issue is an important step in the software reliability modeling testing stage. To achieve the necessary degree of reliability, judgments on optimum allocation of resources among the components must be made. To define dynamic scheduling of total projected cost and testing effort, we propose a multi-objective software reliability model of testing resources with a novel exponential distribution reliability function. To optimize software reliability while lowering allocation costs, an expanded particle swarm optimization (EPSO) is presented. Researchers do trials utilizing completely random testing-resource sets and the entropy function to modify performance. In a standard modular testing phase, the multi-objective models were tested to modules using a weighted cost function and test effort metrics, and the reliability is shown to be 99%.

Gupta et al.[41] proposed a nonlinear multi-objective optimization model regarding data envelopment analysis (DEA) for choosing software systems in the context of optimum redundancy to assure software reliability. For selection and data, the suggested optimizing model incorporates both construct and purchase choices. For assessing the efficiency of software parts based on many inputs or outputs supplied by diverse individuals of the deliberation group, researchers employ the DEA approach. The aggregated data is used to calculate the total efficiency rate of each software application. Using constraints relating to compatibility of chosen components, reliability, execution time, and software system delivery time, the suggested optimization model reduces the overall cost of software system and optimizes the combined value of purchase. It also contains data on the testing that must be done on in-house designed components. To demonstrate the efficacy of the suggested optimization methodology, a real-world case analysis of modularity software engineering is addressed. To their knowledge, no prior research has been done on an integrative optimizer for the software component selection issue including build and/or purchase choices with optimum redundancy. The entire software reliability is 80 percent, with an execution time of 7.647 x 10das.

Kumar et al.[42] examined the software's reliability using soft computing approaches, which are the most efficient way to assess its predictive capability. Based on the software reliability concept, it presents a novel comparison analysis to determine the most appropriate and accurate artificial neural network. In this study, we present a backpropagation-based feedforward neural network for improving software reliability accuracy. Comparisons of performance using a dataset of genuine software assessment issues reveal that the suggested prediction model outperforms the old methodology. The minimal MMRE for FF-NN is 7.21, according to the results. 16.60 is the maximum value for linear regression.

Jabeen et al.[44] proposed a highly precise error iteration analysis technique (HPEIAM) based on error-residuals is suggested to improve the predictive performance of current PSRGMs. SRGMs compute residual errors repeatedly, improving and correcting prediction accuracy to the intended level. HPEIAM's performance is evaluated using various PSRGMs and two sets of actual software failure data, with three quality criteria in mind. Furthermore, researchers compared HPEIAM's projected failures to a GA. In the first few rounds, the findings show that HPEIAM improves goodness-of-fit and predicting performance for each PSRGM. The R2 and RMSE are 99.2 % and 48.8%, respectively. Similarly, after the second cycle, the desired accuracy is attained for 56 of the data intervals, with the RMSE value changing to 2.67. From 170.91 to 143.01, the predicted accuracy increases.

Figure 2 shows number of papers of different transaction including IEEE, springer, Science direct etc. for Software reliability using different evolutionary techniques.[41-51] As per requirement, a number of articles were considered in this study. Maximum number of papers were based on Particle Swarm Optimization to evaluate the software reliability. However, simulation results from various studies show that PSO has some serious shortcomings. After that, the research was moved to ACO. This method was thoroughly investigated. When compared to PSO, their results achieve global

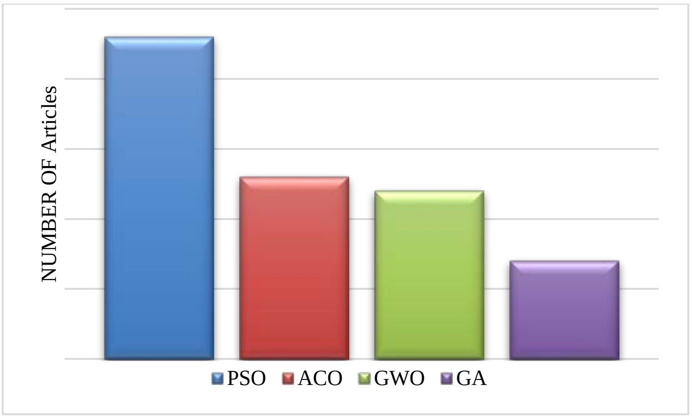


Figure 2. Optimization Algorithm for Software Reliability Analysis

minima in fewer iterations and at a lower value. As a result, it can be concluded that ACO outperforms PSO in distance optimization problems. Another evolutionary technique being researched is Grey Wolf Optimization. Few Contributions were studied in this area where from the result we can conclude that it has lower accuracy and slow convergence in the later stage of search. From figure 2, it is clearly seen that very little research is conducted on one of the evolutionary optimization technique called Genetic Algorithm. This method has numerous advantages and have better scope than any other optimization technique. Figure 3 shows the Software reliability comparison with existing techniques. Hybrid Swarm optimization[44] has 85% software reliability, while Multi objective optimization has 80%.[45] Syntax tree has minimum SR optimization of 75.67%.[46] While genetic algorithm has maximum software reliability around 98.47%.[7]

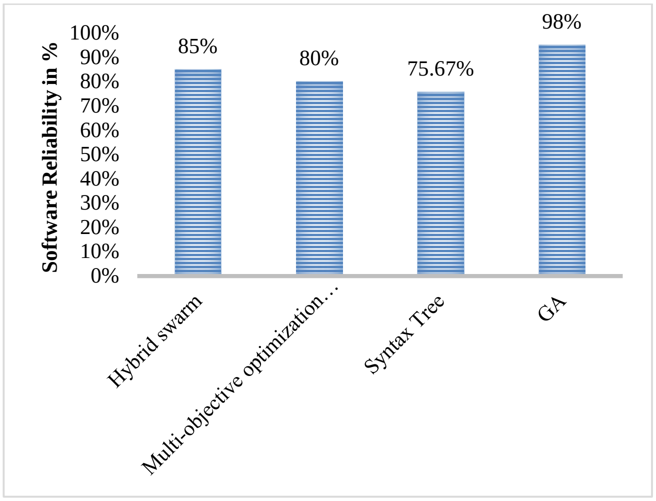


Figure 3. Reliability Performance of Optimization Algorithms

Research Challenges, Limitations, and Future Scope

As a result, improving software dependability is critical before the program's growing importance necessitates a greater capacity to trust it than before, and various areas of business and society confront distinct problems as a result. Defects in software can lead to significant losses such as financial. Software industry is a large industry; it contributes billion dollars to the world's economy annually. Software bugs make software misbehave and cause huge financial loss and security breaches. Reliable software must include extra, often redundant, code to perform the necessary checking for exceptional conditions. This fundamental role means that the reliability of systems software is of primary importance. As a result, prior to the software's deployment, it's critical to improve the software's reliability. The suggested effort will improve the reliability of software. Numerous studies have been conducted in the subject of Software Reliability Optimization, as seen by the literature overview given above.

Reliability growth methodologies have been introduced using a variety of techniques, including:

- Based on incomplete debugging
- Depending on the degree of the software's faults
- Efforts based on testing
- Using Unification Schemes as just a Base
- Transitioning from a single-dimensional to a multi-dimensional framework.

Following major issues occurs identified in above mentioned models:

- Finding the failure parameters.
- To determine statistical significance among determined failure points.
- To find best fit parameters for estimation of reliability on multi-dimensional framework.
- Finding the failure points in artificial intelligence related software.
- In future, this work will be extended to identify the weak reliable points that can lead to attack condition.

Conclusion

While predicting software reliability, the major issue arises due to imbalanced data. This causes chaos for software developers or researchers to deal with imbalanced faulty data. The early prediction causes major issues. Researchers are looking on ML algorithms-based software defect prediction approaches, although they haven't looked into it in depth yet. This work has examined and investigated effective ML approaches for software dependability prediction in terms of dealing with some of this circumstance.

Conflict of Interest

Authors declare no conflict of interest.

References

1. X. Chen, Y. Shen, Z. Cui, X. Ju. Applying feature selection to software defect prediction using multi-objective optimization. *Proc. - Int. Comput. Softw. Appl. Conf.* **2017**, 2, 54–59.
2. N. Raj Kiran, V. Ravi. Software reliability prediction by soft computing techniques. *J. Syst. Softw.* **2008**, 81 (4), 576–583.
3. C. Jin, S.W. Jin. Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms. *Appl. Soft Comput.* **2014**, 15, 113–120.
4. L. Tian, A. Noore. On-line prediction of software reliability using an evolutionary connectionist model. *J. Syst. Softw.* **2005**, 77 (2), 173–180.
5. S. Kassaymeh, S. Abdullah, M. Al-Laham, et al. Salp Swarm optimizer for modeling software reliability prediction problems. *Neural Process. Lett. 2021 536* **2021**, 53 (6), 4451–4487.
6. L. Zhen, Y. Liu, W. Dongsheng, Z. Wei. Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization. *IEEE Access* **2020**, 8, 29354–29369.

7. P. Roy, G.S. Mahapatra, K.N. Dey. Forecasting of software reliability using neighborhood fuzzy particle swarm optimization based novel neural network. *IEEE/CAA J. Autom. Sin.* **2019**, 6 (6), 1365–1383.
8. M. Gheisari, D. Panwar, P. Tomar, et al. An optimization model for software quality prediction with case study analysis using MATLAB. *IEEE Access* **2019**, 7, 85123–85138.
9. P. Sedlacek, E. Zaitseva. Software reliability model based on syntax tree. *Int. Conf. Inf. Digit. Technol. 2021, IDT 2021* **2021**, 73–82.
10. T. Yaghoobi. Parameter optimization of software reliability models using improved differential evolution algorithm. *Math. Comput. Simul.* **2020**, 177, 46–62.
11. P. Dhavakumar, N.P. Gopalan. An efficient parameter optimization of software reliability growth model by using chaotic grey wolf optimization algorithm. *J. Ambient Intell. Humaniz. Comput.* **2021**, 12 (2), 3177–3188.
12. P. Prashant, A. Tickoo, S. Sharma, J. Jamil. Optimization of cost to calculate the release time in software reliability using python. *Proc. 9th Int. Conf. Cloud Comput. Data Sci. Eng. Conflu. 2019* **2019**, 470–474.
13. V. Nagaraju, C. Jayasinghe, L. Fiondella. Optimal test activity allocation for covariate software reliability and security models. *J. Syst. Softw.* **2020**, 168, 110643.
14. H.C. Kim. A study on comparative evaluation of software reliability model applying modified exponential distribution. *Int. J. Eng. Res. Technol.* **2020**, 13 (5), 867–872.
15. Q. Li, H. Pham. A generalized software reliability growth model with consideration of the uncertainty of operating environments. *IEEE Access* **2019**, 7, 84253–84267.
16. K. Sahu, R.K. Srivastava. Soft computing approach for prediction of software reliability. *ICIC Express Lett.* **2018**, 12 (12), 1213–1222.
17. C. Diwaker, P. Tomar, R.C. Poonia, V. Singh. Prediction of software reliability using bio inspired soft computing techniques. *J. Med. Syst. 2018 425* **2018**, 42 (5), 1–16.
18. S.K. Dubey, B. Jasra. Reliability assessment of component-based software systems using fuzzy and ANFIS techniques. *Int. J. Syst. Assur. Eng. Manag.* **2017**, 8 (2), 1319–1326.
19. H. Tong, B. Liu, Y. Wu, B. Xu. Software reliability prediction using chaos theory and heterogeneous ensemble learning. *Risk, Reliability and Safety: Innovating Theory and Practice - Proceedings of the 26th European Safety and Reliability Conference, ESREL 2016*, **2017**, 389.
20. Z. Li, M. Yu, D. Wang, H. Wei. Using hybrid algorithm to estimate and predicate based on software reliability model. *IEEE Access* **2019**, 7, 84268–84283.
21. K. Lu, Z. Ma. Parameter Estimation of software reliability growth models by a modified whale optimization algorithm. *Proc. - 2018 17th Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci. DCABES 2018* **2018**, 268–271.
22. P.R. Bal, S. Kumar. WR-ELM: Weighted regularization extreme learning machine for imbalance learning in software fault prediction. *IEEE Trans. Reliab.* **2020**, 69 (4), 1355–1375.
23. S.H. Aljahdali, M.E. El-Telbany. Software reliability prediction using multi-objective genetic algorithm. *2009 IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA 2009* **2009**, 293–300.
24. K. Bidhan, A. Awasthi. Estimation of reliability parameters of software growth models using a variation of Particle Swarm Optimization. *Proc. 5th Int. Conf. Conflu. 2014 Next Gener. Inf. Technol. Summit* **2014**, 800–805.
25. Sangeeta, K. Sharma, M. Bala. An ecological space based hybrid swarm-evolutionary algorithm for software reliability model parameter estimation. *Int. J. Syst. Assur. Eng. Manag.* **2020**, 11 (1), 77–92.
26. T. Yaghoobi. Parameter optimization of software reliability models using improved differential evolution algorithm. *Math. Comput. Simul.* **2020**, 177, 46–62.
27. P. Kumar, S.K. Singh, S.D. Choudhary. Reliability prediction analysis of aspect-oriented application using soft computing techniques. *Mater. Today Proc.* **2021**, 45, 2660–2665.
28. P. Dhavakumar, N.P. Gopalan. An efficient parameter optimization of software reliability growth model by using chaotic grey wolf optimization algorithm. *J. Ambient Intell. Humaniz. Comput.* **2021**, 12 (2), 3177–3188.
29. 10. C. Diwaker, P. Tomar, R.C. Poonia, V. Singh. Prediction of Software Reliability using Bio Inspired Soft Computing Techniques. *J. Med. Syst.* **2018**, 42 (5), 1–16.
30. L. Yang, Z. Li, D. Wang, H. Miao, Z. Wang. Software defects prediction based on hybrid particle swarm optimization and sparrow search algorithm. *IEEE Access* **2021**, 9, 60865–60879.
31. X. Cai, S. Geng, D. Wu, J. Chen. Unified integration of many-objective optimization algorithm based on temporary offspring for software defects prediction. *Swarm Evol. Comput.* **2021**, 63, 100871.
32. S. Yadav, K. Mohan G. Parameter estimation techniques of software reliability growth models: a critical research with experimentation. *Int. J. Recent Technol. Eng.* **2019**, 8 (4), 7763–7770.
33. Jiarui Wang. Hybrid Wolf Pack and Particle Swarm Optimization Algorithm for Multihop Routing Protocol in WSN. *J. Netw. Commun. Syst.* **2020**, 3 (3), 37–44.
34. K. Lu, Z. Ma. A modified whale optimization algorithm for parameter estimation of software reliability growth models. *J. Algorithms Comput. Technol.* **2021**, 15.
35. A. Rajasekaran, R. Varalakshmi. An optimized feature selection using fuzzy mutual information based ant colony optimization for software defect prediction. *Int. J. Eng. Technol.* **2017**, 7 (1.1), 456–460.
36. H. Okamura, A. Murayama, T. Dohi. A Unified Parameter Estimation Algorithm for Discrete Software Reliability Models. *OPSEARCH 2005 424* **2017**, 42 (4), 355–377.
37. Sangeeta, K. Sharma, M. Bala. An ecological space based hybrid swarm-evolutionary algorithm for software reliability model parameter estimation. *Int. J. Syst. Assur. Eng. Manag.* **2020**, 11 (1), 77–92.
38. R.S. Wahono, N. Suryana. Combining particle swarm optimization based feature selection and bagging technique for software defect prediction. *Int. J. Softw. Eng. its Appl.* **2013**, 7 (5), 153–166.
39. R.S. Wahono, N. Suryana, S. Ahmad. Metaheuristic optimization based feature selection for software defect prediction. *J. Softw.* **2014**, 9 (5).
40. L. Zhen, Y. Liu, W. Dongsheng, Z. Wei. Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization. *IEEE Access* **2020**, 8, 29354–29369.
41. A. Jindal, A. Gupta, Rahul. Comparative analysis of software reliability prediction using machine learning and deep learning. *Proc. 2nd Int. Conf. Artif. Intell. Smart Energy, ICAIS 2022* **2022**, 389–394.
42. G. Jabeen, P. Luo, W. Afzal. An improved software reliability prediction model by using high precision error iterative analysis method. *Softw. Test. Verif. Reliab.* **2019**, 29 (6–7).
43. Sangeeta, K. Sharma, M. Bala. An ecological space based hybrid swarm-evolutionary algorithm for software reliability model parameter estimation. *Int. J. Syst. Assur. Eng. Manag.* **2020**, 11 (1), 77–92.
44. G. Jabeen, P. Luo, W. Afzal. An improved software reliability prediction model by using high precision error iterative analysis method. *Softw. Test. Verif. Reliab.* **2019**, 29 (6–7).
45. P. Sedlacek, E. Zaitseva. Software Reliability Model based on Syntax Tree. *Int. Conf. Inf. Digit. Technol. 2021, IDT 2021* **2021**, 73–82.
46. S. Sinha, N.K. Goyal, R. Mall. Survey of combined hardware–software reliability prediction approaches from architectural and system failure viewpoint. *Int. J. Syst. Assur. Eng. Manag.* **2019**, 10 (4), 453–474.
47. D. Singh, S. Sinha, V. Thada. A novel attribute based access control model with application in IaaS cloud. *J. Integr. Sci. Technol.* **2022**, 10 (2), 79–86..
48. P. Rani, G.S. Mahapatra. A neuro-particle swarm optimization logistic model fitting algorithm for software reliability analysis. *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* **2019**, 233 (6), 958–971.
49. M. Yazdani, M. Babagolzadeh, N. Kazemitash, M. Saberi. Reliability estimation using an integrated support vector regression – variable neighborhood search model. *J. Ind. Inf. Integr.* **2019**, 15, 103–110.
50. J.S. Wang, S.X. Li. An Improved Grey Wolf Optimizer Based on Differential Evolution and Elimination Mechanism. *Sci Rep,* **2019**, 9 (1).
51. R.R. Patil, A.U. Ruby, B.N. Chaithanya, S. Jain, K. Geetha. Review of fundamentals of Artificial Intelligence and application with medical data in healthcare. *J. Integr. Sci. Technol.* **2022**, 10 (2), 126–133.