ISSN 2278-3091
Volume 9 No.2, March -April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse30922020.pdf
https://doi.org/10.30534/ijatcse/2020/30922020

A Hybrid Quartile deviation-based Support vector regression model for software reliability datasets

Y. Geetha Reddy[1] , Dr. Y Prasanth[2]
Research Scholar[1], Professor[2]
Department of Computer science and Engineering
KLEF, Guntur District, A.P., India.

ABSTRACT

Software reliability prediction models are used to predict the fault rate of the software systems using machine learning models. A large number of traditional reliability measures are used to test the software faults in the debugging and testing process. Most of the traditional machine learning based fault prediction models are integrated with standard software reliability growth measures for reliability severity classification. However, these models are used to predict the reliability level of binary class with less standard error. In this paper, a hybrid support vector regression-based quartile deviation growth measure is implemented on the training fault datasets. Experimental results are simulated on various reliability datasets with different configuration parameters for fault prediction.

Key words : Software fault detection, reliability prediction ,support vector machine.

1. INTRODUCTION

Reliability in its simplest form means that a failure cannot occur within a certain period of time. The reliability concept thus stresses the probability, expected function(s), time and operating conditions of four components. Reliability also depends on the conditions of the system that may or may not change over time. Software systems have increased significantly in size and complexity in recent decades, and the trend is expected to continue in the future[1]. Computer reliability and accessibility, usability, performance, serviceability, capabilities and documentation are important attributes of software quality. Software reliability is difficult to achieve, since software complexity seems to be high. While it is difficult to achieve a certain degree of reliability of any highly complex system, including software, system developers tend to upgrade the software layer with complexity and rapidly developing system sizes.The Software Reliability Growth Model (SRGMs) is a software reliability model (SRMs) design recognition class which is converted into a mathematical model. The reliability assessment of recent system updates is an important challenge in IT software management[2].

The probabilistic models are based on dynamic models and represented as time-based statistical distributions. All these models are used to predict current trends and to predict future trends in reliability. Probabilistic software reliability prediction models use statistical methods to estimate variables such as system error numbers, failure rates, software complexity and program failure, etc. There exists a number of software models in the literature, but none of them is ideal. The selection of an appropriate estimate model based on a specific application is a major research problem[3]. A data set that includes instances of defined classes and a test data set for which the class must be decided must therefore be entered. The quality of the data provided for learning, and also the type of algorithm used in machine learning, depends greatly on the ability to classify successfully. Categorical labels (discrete, unorderly) estimate classification results of continuously valued function models. It implies that numerical data values are expected instead of class marks to be incomplete or inaccessible. Regression analysis is the most widely used statistical method for numerical forecasting. Although other methods are available, the prediction also consists in identifying distribution trends based on available data. Genetic algorithms are also implemented to maximize the number of delayed input neurons and the number of neurons in the neural network's hidden architectural layer. We have demonstrated, using the software model for online adaptation, that good-fitness and next-step predictability is better than traditional methods when cumulative software failure times are forecast. Because those variables ' meanings are certainly not known. Many potential values can equate to the likelihood of occurrence. Therefore, we really don't know when the next loss will happen. We know only a few possible failure times and their likelihood. T Two types of fault data, namely time-domain data and interval-domain data, were widely used in software reliability modeling. The time-domain form is determined by the time the failure occurred. Learning supervised is a methodology for machine learning to build a data structure for preparation. Maximum Likelihood Assessment (MLE) is a common statistical method for the determination of the probability distribution parameters

underlying a given dataset. Throughout literature there are many predictive models of the reliability of software based neural networks, which are better known than certain statistical models[4-6].Computer reliability is one of the key factors taken into account in maintaining the accuracy of the computer. Simply put, software reliability is about system failure or failure[7]. Success and success are two distinct variables commonly included in our software development. Fault could be identified as a fault or error during the development phase.

These models use failure data obtained during the testing period of software development [78 to determine the growth behavior and hence derive reliability prediction. Various types of SRGMs have been developed and implemented in many different industry sectors since the 1970s . These models are further classified into two types, namely: failure rate models, and failure intensity models or as known as non-homogeneous Poisson process(NHPP) models.

There are many SRGMs has been proposed or developed. Most of them are designed with their own limitations, assumptions and unique characteristics. Each model suited and produced good result for certain data set, but no model is good enough for all data sets from different domains [9]. The generalization problem of SRGM as further complicates model selection forreliability prediction process. However, these studies are using numerical methods like least square estimation (LSE) and nonlinear regression (NLR) as the SRGM parameter estimation methods which can be improved by computational intelligence (CI)method such as PSO. This approach [10] uses fuzzy logic with neural networks in software reliability prediction. The recurrent neural network is trained using the back-propagation algorithm. The number of failures and cumulative execution time in the failure dataset is used as input to the network to predict the next step failure.

2. RELATED WORKS

Lazarova et al. have developed various SRGMs concerning the growth rate software reliability index for error detection[11]. Liet.al, proposed a measuring method as an indicator collection, gathering data for the testing of all those metrics[12].Mirchandaniet al. suggested the non-homogeneous Poisson method-based software reliability growth pattern because the detection of these errors might also lead to detection of other errors without failure[13].Nagaraju proposed an evolutionary model of the neural network to estimate and predict the software reliability based on a multimedia architecture input and output. In this study, the development of neural network models for software-reliability predictions was proposed using an Exponential and Logarithmic Encoding Scheme. Neural network models with the two encoding schemes above have shown a better prediction of cumulative failures than some statistical models. However,[14] the value of the encoding parameter is calculated by repeated hit / test experiments. This paper presents recommendations for encoding parameter selection, which provide consistent results for various data sets. The proposed solution is implemented using 18 separate data sets and a clear result for all datasets is observed. The method was compared to known statistical models using three sets of change points.

Rani [15] proposed a neural network approach focused on predictions of software reliability. He compared the approach to parametric model recalibration with some meaningful predictive measures with the same data sets. We concluded that better predictors are neural network methods.

Rizviet al.[16] proposed a system in which software reliability based on the neural network would be expected. They used the reverse propagation algorithm for instructions. They used several failure times in the last 50 to estimate the next failure as output. The performance of approaches was calculated by changing the number of input nodes and hidden nodes. We concluded that the success of the strategy usually depends on the quality of the data sets.

Sagar[17] submitted a neural network approach focused on the evolutionary prediction of device reliability. They used single output architecture with multiple delayed inputs. Vojdani[18] suggested two models for cumulative system failure estimation, such as exponential neural network encoding (NNEE) and logarithmic encoding (NNLE). He encoded the data with the above two encoding scheme, i.e. the time of execution. He used the four dataset method and compared the results with some statistical models and found better results than those models.

Wanget al.[19] have proposed to reuse it data from previous projects / releases for failure to boost early reliability for current projects / releases. Wang et al.[20] proposed the combinational dynamic weighted model (DWCM) based on a neural network for the prediction of device reliability. Based on the software-reliability growth model (SRGM), they used various activation functions within the secret layer. The method was used on two sets of data and the effect was compared with certain statistical models. The experimental results indicate that the DWCM approach is more successful than traditional models. The neural network is a methodology for performance computation. The machine performance can previously be predicted on the basis of our neural network architecture. The system is also trained unless its desired output or destination can be achieved. For training purposes, we use different learning techniques that are freely described as supervised and unattended learning.[21] Software reliability is a quantitative study of every software designed since it affects directly software quality[22]. An efficient software reliability model is required in order to achieve good results. The previously developed reliability model is based on

the analysis of faults linked to the code and context in which it was implemented [23]. All software reliability models are designed based on the execution time and calendar time. The time required or spent by the processor in the execution of instructions from the program is the execution time of any program [24].

3.PROPOSED MODEL

In this section, a statistical quartile deviation-based improved SVR prediction model is proposed on the software reliability datasets. In this work, a novel approach to predict the software reliability on the training and test software fault data. This model is integrated with the quartile deviation growth function in order to fit the S shaped curve. The main objective of this model is to improve the prediction accuracy and to minimize the error rate for software quality and reliability estimation. The S-shaped models show the asymptotic behavior similar to the concave model. The failure data used to track the curve are analyzed in two software testing phases. Therefore, the S-shape curve acts in the same way as the concave curve at later testing stages. In the proposed model, reliability estimation is performed in two phases. In the initial phase, quartile deviation based error estimation is calculated on the training data for software reliability prediction. In the second phase, a hybrid support vector regression model is designed and implemented on the computed S-shaped training data as shown in figure 1.

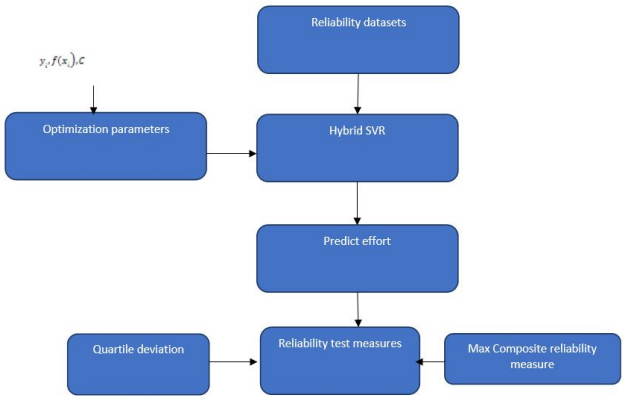

Figure 1: Proposed Framework

In the proposed model, a enhanced support vector regression is designed and implemented on the software fault dataset to improve the prediction rate and to minimize the error rate. The following proposed SVR model is implemented on the fault data. Initially, input data is given to hybrid SVR model to predict the effort rate. The prediction values of the SVR are tested using the Quartile deviation model and maximized composite reliability measures. These measures are used to find the deviation, skewness and shape of the dataset.

Let m(x) be the input data, $\hat{m}$ be the estimation function. $\hat{m}$ values are estimated by using multiple linear regression method.Then the objective function of the proposed SVR model is given as

$$C(x) := \frac{1}{2} \| w \|^2 + \xi.\psi(x).\phi(x)$$

where

$$\psi(x) = | m(x) - \hat{m}(x) | = | m(x) - MLR(x) |$$

$$MLR(x) = \text{Multiple linear regression}$$

$$\phi(x, x') = e^{-\|x-x;\|^2/2.\sigma^2}$$

$$\min_{\xi_k, \xi_k^*} C(x) = \frac{1}{2} \| w \|^2 + \lambda.\psi(x).\phi(x) + b$$

$$\min_{\xi_k, \xi_k^*} C(x) = \frac{1}{2} \| w \|^2 + \lambda.. | \xi_k - \xi_k^* | .\phi(x) + b$$

4.EXPERIMENTAL RESULTS

Experimental results are carried out on the software failure datasets taken form the DS1 reported by K.Okumoto. During 56 weeks of testing, a total of 124 faults are identified to test the stability. The second , third and fourth datasets DS2,DS3,DS4 are taken from Rome air development center(RADC) projects.

Table 1: DS1 for fault prediction based on severity level

W	CF	Label
1	16	L
2	24	L
3	27	L
4	55	M
5	41	L
6	49	L
7	54	M
8	58	M
9	69	M
10	75	H
11	81	H
12	86	H
13	90	H
14	93	H
15	96	H
16	98	H
17	99	H
18	100	H
19	100	H
20	100	H

Table 2: DS2 for fault prediction based on severity level

W	CF	Label
1	28	L
2	29	L
3	29	L
4	29	L
5	29	L
6	37	M
7	63	M
8	92	H
9	116	H
10	125	H
11	139	H
12	152	H
13	164	H
14	164	H
15	165	H
16	168	H
17	170	H
18	176	H

Table 3: DS3 for fault prediction based on severity level

W	F	label
40	71	M
41	72	M
42	74	M
43	74	M
44	80	M
45	84	M
46	84	M
47	84	M
48	84	M
49	85	H
50	86	H
51	89	H
52	90	H
53	90	H
54	92	H
55	108	H
56	120	H
57	128	H
58	129	H
59	139	H
60	146	H

Table 4: DS4 for fault prediction based on severity level

W	F	Label
33	79	L
34	80	L
35	82	L
36	83	L
37	83	L
38	84	L
39	84	L
40	85	M
41	85	M
42	87	M
43	87	M
44	87	M
45	89	M
46	89	M
47	91	H
48	91	H
49	94	H

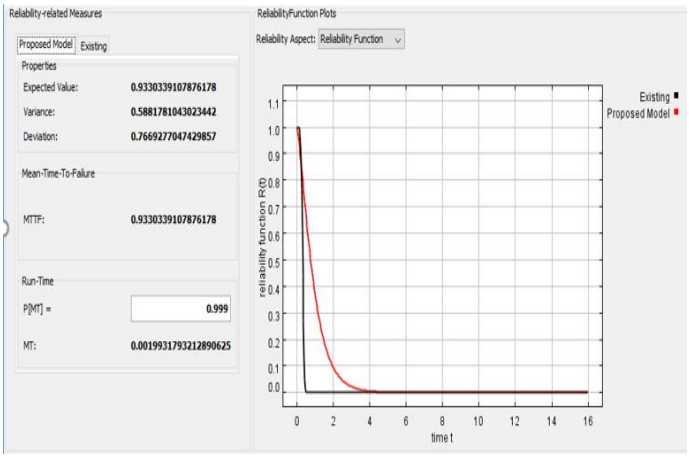

Figure 2: Mean time to failure rate and runtime of the proposed model to the exponential model.

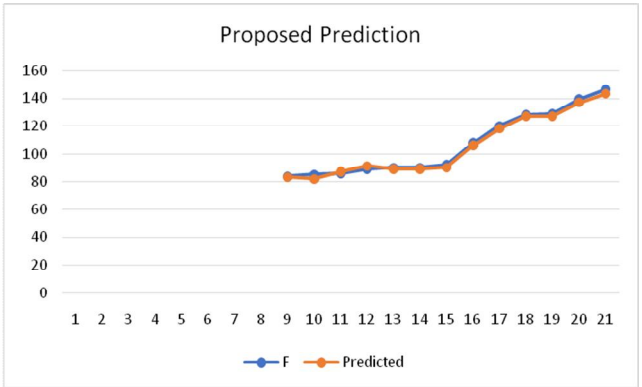

Figure 3: Comparison of proposed fault prediction model on all datasets.

Figure 2, describes the mean time to failure rate of the proposed model to the traditional exponential model on testing data. From the figure, it is clear that the present model has low error rate and better mean time to failure rate than the traditional model.

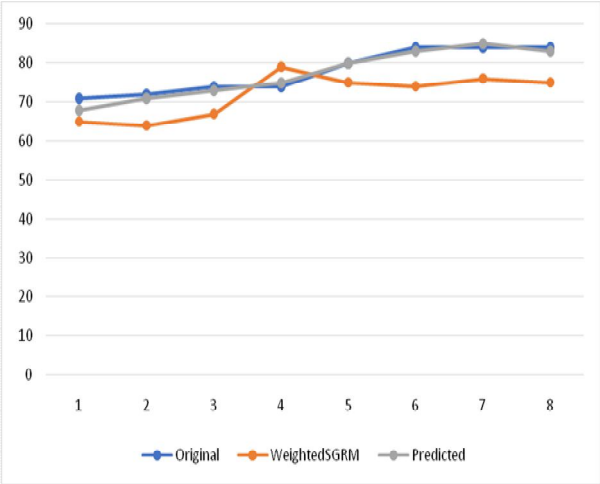

Figure 4: Comparison of proposed fault prediction model to existing weighted SGRM model on all datasets.

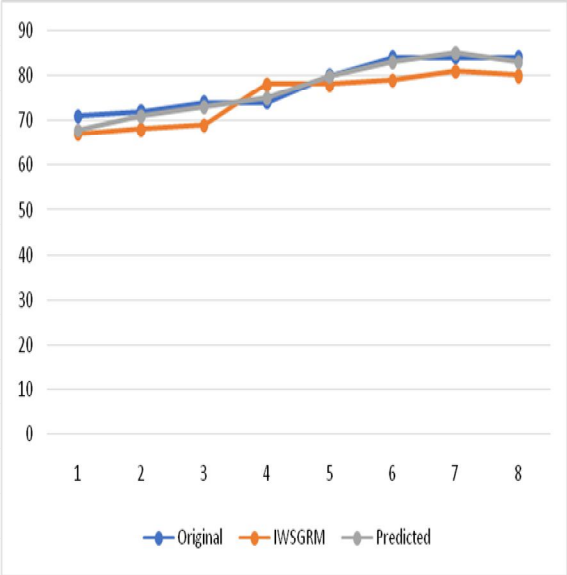

Figure 5: Comparison of proposed fault prediction model to existing improved weighted SGRM model on all datasets.

5.CONCLUSION

Software reliability fault prediction plays a vital role in small- and large-scale software applications. In this paper, a hybrid support vector regression-based quartile deviation model is implemented on the different software reliability datasets. Most of the traditional machine learning based fault prediction models are integrated with standard software reliability growth measures for reliability severity classification. However, these models are used to predict the reliability level of binary class with less standard error. Experimental results proved that the proposed reliability fault prediction model has better performance in terms of prediction and time are concerned.

REFERENCES

[1]J. Cho, S. M. Shin, S. J. Lee, and W. Jung, “Exhaustive test cases for the software reliability of safety-critical digital systems in nuclear power plants,” Nuclear Engineering and Design, vol. 352, p. 110151, Oct. 2019, doi: 10.1016/j.nucengdes.2019.110151.

[2]L. V. Utkin and F. P. A. Coolen, “A robust weighted SVR-based software reliability growth model,” Reliability Engineering & System Safety, vol. 176, pp. 93–101, Aug. 2018, doi: 10.1016/j.ress.2018.04.007.

[3]E.Abuta and J. Tian, “Reliability over consecutive releases of a semiconductor Optical Endpoint Detection software system developed in a small company,” Journal of Systems and Software, vol. 137, pp. 355–365, Mar. 2018, doi: 10.1016/j.jss.2017.12.006.

[4]C.Jin and S.-W. Jin, “Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization,” Applied Soft Computing, vol. 40, pp. 283–291, Mar. 2016, doi: 10.1016/j.asoc.2015.11.041.

[5]M. S. Alhammadi, B. S. Almaqrami, and B. Cao, “Reliability of Beta-angle in different anteroposterior and vertical combinations of malocclusions,” Orthodontic Waves, vol. 78, no. 3, pp. 111–117, Sep. 2019, doi: 10.1016/j.odw.2019.02.002.

[6]D. Amara and L. B. ArfaRabai, “Towards a New Framework of Software Reliability Measurement Based on Software Metrics,” Procedia Computer Science, vol. 109, pp. 725–730, Jan. 2017, doi: 10.1016/j.procs.2017.05.428.

[7]J.-E. Byun, H.-M. Noh, and J. Song, “Reliability growth analysis of k-out-of-N systems using matrix-based system reliability method,” Reliability Engineering & System Safety, vol. 165, pp. 410–421, Sep. 2017, doi: 10.1016/j.ress.2017.05.001.

[8]F.Febrero, C. Calero, and M. Ángeles Moraga, “Software reliability modeling based on ISO/IEC SQuaRE,” Information and Software Technology, vol. 70, pp. 18–29, Feb. 2016, doi: 10.1016/j.infsof.2015.09.006.

[9]A.Lanna, T. Castro, V. Alves, G. Rodrigues, P.-Y. Schobbens, and S. Apel, “Feature-family-based reliability analysis of software product lines,” Information and Software Technology, vol. 94, pp. 59–81, Feb. 2018, doi: 10.1016/j.infsof.2017.10.001.

[10]V. Ivanov, A. Reznik, and G. Succi, “Comparing the reliability of software systems: A case study on mobile operating systems,” Information Sciences, vol. 423, pp. 398–411, Jan. 2018, doi: 10.1016/j.ins.2017.08.079.

[11]S.Lazarova-Molnar and N. Mohamed, “Reliability

Assessment in the Context of Industry 4.0: Data as a Game Changer," Procedia Computer Science, vol. 151, pp. 691–698, Jan. 2019, doi: 10.1016/j.procs.2019.04.092.

[12]Q. Li and H. Pham, "NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage," Applied Mathematical Modelling, vol. 51, pp. 68–85, Nov. 2017, doi: 10.1016/j.apm.2017.06.034.

[13]C.Mirchandani, "Adaptive Software Reliability Growth," Procedia Computer Science, vol. 140, pp. 122–132, Jan. 2018, doi: 10.1016/j.procs.2018.10.309.

[14]V.Nagaraju, V. Shekar, J. Steakelum, M. Luperon, Y. Shi, and L. Fiondella, "Practical software reliability engineering with the Software Failure and Reliability Assessment Tool (SFRAT)," SoftwareX, vol. 10, p. 100357, Jul. 2019, doi: 10.1016/j.softx.2019.100357.

[15]P. Rani and G. S. Mahapatra, "A novel approach of NPSO on dynamic weighted NHPP model for software reliability analysis with additional fault introduction parameter," Heliyon, vol. 5, no. 7, p. e02082, Jul. 2019, doi: 10.1016/j.heliyon.2019.e02082.

[16]S. W. A. Rizvi, V. K. Singh, and R. A. Khan, "Fuzzy Logic Based Software Reliability Quantification Framework: Early Stage Perspective (FLSRQF)," Procedia Computer Science, vol. 89, pp. 359–368, Jan. 2016, doi: 10.1016/j.procs.2016.06.083.

[17]B. B. Sagar, R. K. Saket, and Col. Gurmit Singh, "Exponentiated Weibull distribution approach based inflection S-shaped software reliability growth model," Ain Shams Engineering Journal, vol. 7, no. 3, pp. 973–991, Sep. 2016, doi: 10.1016/j.asej.2015.05.009.

[18]A.Vojdani and G. H. Farrahi, "Reliability assessment of cracked pipes subjected to creep-fatigue loading," Theoretical and Applied Fracture Mechanics, vol. 104, p. 102333, Dec. 2019, doi: 10.1016/j.tafmec.2019.102333.

[19]J. Wang, Z. Wu, Y. Shu, and Z. Zhang, "An optimized method for software reliability model based on nonhomogeneous Poisson process," Applied Mathematical Modelling, vol. 40, no. 13, pp. 6324–6339, Jul. 2016, doi: 10.1016/j.apm.2016.01.016.

[20]J. Wang and C. Zhang, "Software reliability prediction using a deep learning model based on the RNN encoder–decoder," Reliability Engineering & System Safety, vol. 170, pp. 73–82, Feb. 2018, doi: 10.1016/j.ress.2017.10.019.

[21]J. Yang, Y. Liu, M. Xie, and M. Zhao, "Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes," Journal of Systems and Software, vol. 115, pp. 102–110, May 2016, doi: 10.1016/j.jss.2016.01.025.

[22]O.Yazdanbakhsh, S. Dick, I. Reay, and E. Mace, "On deterministic chaos in software reliability growth models," Applied Soft Computing, vol. 49, pp. 1256–1269, Dec. 2016, doi: 10.1016/j.asoc.2016.08.006.

[23]M. Zhu and H. Pham, "A two-phase software reliability modeling involving with software fault dependency and imperfect fault removal," Computer Languages, Systems & Structures, vol. 53, pp. 27–42, Sep. 2018, doi: 10.1016/j.cl.2017.12.002.

[24]B. Zou, M. Yang, E.-R. Benjamin, and H. Yoshikawa, "Reliability analysis of Digital Instrumentation and Control software system," Progress in Nuclear Energy, vol. 98, pp. 85–93, Jul. 2017, doi: 10.1016/j.pnucene.2017.03.006.