

**ANALYSIS AND DESIGN OF SOFTWARE RELIABILITY
MODEL FOR BIG FAULT DATA USING SOFT
COMPUTING**

A Thesis Submitted

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF**

**DOCTOR OF PHILOSOPHY
IN
COMPUTER APPLICATIONS**

By

SHALINI SHARMA

Regd. No. - 18SCSE3020005

Supervisor

Dr. Naresh Kumar

Professor, SCSE

Galgotias University

Greater Noida (U.P.), India

Co-Supervisor

Dr. Kuldeep Singh Kaswan

Professor, SCSE

Galgotias University

Greater Noida (U.P.), India

Presently-

Professor, ACSE

G L Bajaj Institute of Technology and Management

Greater Noida (U.P.), India



**GALGOTIAS UNIVERSITY
GREATER NOIDA, UTTAR PRADESH**

May, 2023

ANALYSIS AND DESIGN OF SOFTWARE RELIABILITY MODEL FOR BIG FAULT DATA USING SOFT COMPUTING

A Thesis Submitted

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER APPLICATIONS

By

SHALINI SHARMA

Regd. No. – 18SCSE3020005

Supervisor

Dr. Naresh Kumar

Professor, SCSE

Galgotias University

Greater Noida (U.P), India

Co-Supervisor

Dr. Kuldeep Singh Kaswan

Professor, SCSE

Galgotias University

Greater Noida (U.P), India

Presently-

Professor, ACSE

G L Bajaj Institute of Technology and Management

Greater Noida (U.P), India




**GALGOTIAS UNIVERSITY
GREATER NOIDA, UTTAR PRADESH**

May, 2023

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis, entitled "*Analysis and Design of software reliability model for big fault data using soft computing*" in fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Faculty and submitted in Galgotias University, Greater Noida is an authentic record of my work carried out during a period from 2018 to 2023 under the supervision of Dr. Naresh Kumar and Dr. Kuldeep Singh Kaswan.

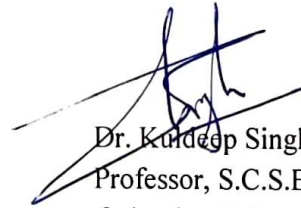
The matter embodied in this thesis has not been submitted by me for the award of any other degree at this or any other University/Institute.


(Shalini Sharma)

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.



Dr. Naresh Kumar
Professor, S.C.S.E
Galgotias University
Greater Noida (U.P), India



Dr. Kuldeep Singh Kaswan
Professor, S.C.S.E
Galgotias University
Greater Noida (U.P), India

Presently-
Professor
Applied Computational Science and Engineering (ACSE)
G L Bajaj Institute of Technology and Management
Greater Noida (U.P), India

The Ph.D. Viva-Voice examination of Ms. Shalini Sharma Research Scholar has been held on _____.

Sign. of Supervisor(s)

Sign. of Co-Supervisor(s)

ABSTRACT

With technological advancement in the current scenario, software usage became an integral part of our daily activities ranging from various applications usage through mobile to highly complicated medical devices used in surgeries. Software reliability plays a crucial part in the proper functioning of software at the site and rendering services to the customer. Therefore, it is of utmost importance to eliminate as many faults in the software as possible before its release. The need to deliver a high-quality product becomes a major concern in the industry. Product quality is the only factor that determines its success in the market and can be identified with its reliability. Development of a system became complex and costly due to technological advancement thus one needs to address the criteria regarding security, development cost, and reliability during the development phase to ensure a defect-free, cost-effective, and reliable final product. Moreover, a reliability assessment is required on the upgraded versions of the already existing systems. Existing systems are continuously monitored for any possible fault and additional components are added in the new version to address the resulting issues which further required an up-gradation. As the complexity of a system increases so do its functionality and capabilities but since the reliability is inversely proportional to the degree of software complexity achieving a balance between complexity and reliability became difficult.

Software companies undergo rigorous testing to remove any probable causes that result in problems and hinder the smooth functioning or reliability of software. Although rigorous testing is carried out to remove software faults they cannot be removed. Developers make use of software reliability models for reliability estimation either by selecting or developing a reliability model suitable for their environmental conditions. The usage of Software reliability growth models (SRGM) for reliability assessment of software quality is centered around the reliability phenomenon. Although reliability models are ideal for measuring and predicting reliability, it's challenging to find an optimal model that works well in all environmental conditions and on different types of datasets. Reliability models are abundant in the literature. Still, not all models can exactly depict reality since while determining the model parameters, there is always

the possibility of uncertainty. Also, model selection depends on the evaluated parameter's value, comparison criteria for model selection, and fault data set. The parameter evaluation and hence the model's capability is tied to the usage of a particular data set making predictions less accurate. With the extensive usage of Big-data, the usage of a distributed, high-capacity storage system with a fast-accessing mechanism is required to handle its high speed, high volume, and variety, characteristics which may arise errors in software due to hardware malfunctioning. Similarly, because of unfamiliarity with specific software and an abundant amount of data to handle, give rise to errors in software because of human negligence. Several reliability models have been developed to determine the reliability of a software product assuming that the fault in software results only due to incorrect specifications or errors in code they didn't consider the fault induced in software due to external factors. A great deal of research has been carried out to make use of the various combination of existing models in developing new hybrid models. Parameter evaluation of such hybrid models based on the mathematical equation is very difficult. Their non-linearity and complexity make the statistical parameter evaluation a challenging task. Software reliability models are generally used for the development of such mathematical models which further depends on accurate prediction and parameter optimization based on experimental data. Thus, the motivation of this work is to develop a hybrid model using a combination of NHPP models to handle not only pure software errors but also the errors resulting in software due to hardware malfunctioning and manual intervention without any assumption.

This research aims to develop a hybrid model that, apart from pure software errors, also considers induced errors in software due to environmental factors into consideration. A direct modification in an NHPP model that can successfully handle software errors is to combine it with other NHPP models to tackle induced errors resulting from hardware and user. We developed 33 hybrid models by combining NHPP models in various combinations according to their characteristics. To access these developed models, we formulated an estimation function and ranking methodology to select the best model based on the accuracy of estimation. The developed hybrid models were compared with existing traditional models using thirteen comparison criteria. Lastly, soft computing techniques like Genetic Algorithm, Simulated Aneling, and Particle Swarm Optimization were utilized for parameter evaluation and optimization.

ACKNOWLEDGMENTS

At this moment it is a pleasant task to extend my sincere thanks to the people who so generously contributed to the completion of my doctoral dissertation. First and foremost, my deep sense of gratitude to my research supervisor Dr. Naresh Kumar, Dean, PG & Ph.D., Galgotias University. I thank him wholeheartedly for accepting me as a Ph.D. student and for all his time, efforts, patience, and valuable guidance that he invested in and provided throughout this study. All my research knowledge is the result of his tremendous academic help, support, and encouragement. Besides my supervisor, I wish to express my sincere gratitude to my co-supervisor Dr. Kuldeep Singh Kaswan, Professor, School of Computing Science and Engineering, Galgotias University for his insightful guidance, kind concern, and timely help. His constant encouragement, motivation, and inspiration helped me in the completion of this study and made it an unforgettable experience for me.

My sincere thanks to Dr. Alok Katiyar, Professor & Research Coordinator, School of Computing Science and Engineering, Galgotias University for his support related to the smooth functioning of research work and documentation.

I am also indebted to my colleagues and all my well-wishers whose support and affection strengthened my resolve. Special thanks to my husband Sanjay whose consistent support and motivation made it all possible and my beloved daughter Aanya for coping well without my consistent nurturing guidance. Without their support, care, and understanding it would have been very difficult for me to complete this work.

Last, but not least, I owe it all to Almighty God who has always been the invisible divine force behind my accomplishments.

To acknowledge I solemnly owe this work.

Place: Grater Noida, U.P, India

Shalini Sharma

Date:

TABLE OF CONTENT

CHAPTER 1	: INTRODUCTION	1-8
1.1.	Background of Problem	1
1.2.	Problem Statement	2
1.3.	Motivation	3
1.4.	Nature of Problem	4
1.5.	Objective of Research	4
1.6.	Research Methodology	4
1.7.	Significance of the Research	5
1.8.	Organization of Thesis	6
CHAPTER 2	: LITERATURE REVIEW	9-45
2.1.	Big Data	9
2.1.1.	Big Data Analytics	9
2.1.2.	Big Data Challenges	10
2.2.	Understanding Reliability	11
2.2.1.	Software V/S Hardware Reliability	11
2.2.2.	Software Development Life Cycle (SDLC)	12
2.3.	Importance of Modelling	14
2.4.	Elements of Reliability Modelling	16
2.4.1.	Failures, Outages, and Faults	16
2.4.2.	Calendar, Execution and Clock Time	17
2.5.	Measuring Software Reliability	17
2.5.1.	Reliability Function	17
2.5.2.	Mean Time to Failure	18
2.5.3.	Mean Time to Repair	18
2.5.4.	Mean Time Between Failure	18
2.5.5.	Failure Rate	18
2.5.6.	Hazard Rate	19

2.6.	Model Classifications	19
2.6.1.	Early Prediction Models	19
2.6.2.	Architecture Based Models	20
2.6.3.	Software Reliability Growth Models	21
2.6.4.	Input Domain-Based Models	21
2.6.5.	Hybrid Black Box	22
2.6.6.	Hybrid White Box	22
2.6.7.	Non-Homogeneous Poisson Process Model	22
2.6.8.	Big Data and Reliability	24
2.7.	Parameter Evaluation of Reliability Models	25
2.8.	Criteria For Comparing Reliability Models	26
2.9.	Literature Review	28
2.9.1.	Big Data Applications	29
2.9.2.	Big Data Reliability	31
2.9.3.	Software Reliability Hybrid Model	37
2.9.4.	Parameter Estimation Technique	41
2.10.	Findings of Critical Review	43
CHAPTER 3	: MODEL DEVELOPMENT, SELECTION FUNCTION, COMPARISON CRITERIA AND RANKING METHODOLOGY	46-70
3.1.	Induced Errors in Software	
3.1.1.	Volume	46
3.1.2.	Velocity	47
3.1.3.	Variety	47
3.2.	Hybrid Model Development	47
3.2.1.	Pure software errors	47
3.2.2.	Hardware generated errors	48
3.2.3.	Human errors	48
3.3.	Mathematical Formulation of Hybrid Models	48

3.3.1	Sharma Kumar Kaswan – 1 model	50
3.3.2	Sharma Kumar Kaswan – 2 model	51
3.3.3	Sharma Kumar Kaswan – 3 model	52
3.3.4	Sharma Kumar Kaswan – 4 model	52
3.3.5	Sharma Kumar Kaswan – 5 model	52
3.3.6	Sharma Kumar Kaswan – 6 model	52
3.3.7	Sharma Kumar Kaswan – 7 model	53
3.3.8	Sharma Kumar Kaswan – 8 model	53
3.3.9	Sharma Kumar Kaswan – 9 model	53
3.3.10	Sharma Kumar Kaswan – 10 model	54
3.3.11	Sharma Kumar Kaswan – 11 model	54
3.3.12	Sharma Kumar Kaswan – 12 model	54
3.3.13	Sharma Kumar Kaswan – 13 model	54
3.3.14	Sharma Kumar Kaswan – 14 model	55
3.3.15	Sharma Kumar Kaswan – 15 model	55
3.3.16	Sharma Kumar Kaswan – 16 model	55
3.3.17	Sharma Kumar Kaswan – 17 model	55
3.3.18	Sharma Kumar Kaswan – 18 model	56
3.3.19	Sharma Kumar Kaswan – 19 model	56
3.3.20	Sharma Kumar Kaswan – 20 model	56
3.3.21	Sharma Kumar Kaswan – 21 model	56
3.3.22	Sharma Kumar Kaswan – 22 model	57
3.3.23	Sharma Kumar Kaswan – 23 model	57
3.3.24	Sharma Kumar Kaswan – 24 model	57
3.3.25	Sharma Kumar Kaswan – 25 model	57
3.3.26	Sharma Kumar Kaswan – 26 model	58
3.3.27	Sharma Kumar Kaswan – 27 model	58
3.3.28	Sharma Kumar Kaswan – 28 model	58
3.3.29	Sharma Kumar Kaswan – 29 model	59
3.3.30	Sharma Kumar Kaswan – 30 model	59
3.3.31	Sharma Kumar Kaswan – 31 model	59
3.3.32	Sharma Kumar Kaswan – 32 model	59
3.3.33	Sharma Kumar Kaswan – 33 model	60
3.4.	Selection Function	60
3.5.	Comparison Criteria	61
3.5.1.	Mean Value	61
3.5.2.	Mean Square Error	62
3.5.3.	Mean Absolute Deviation	62
3.5.4.	Predictive Ratio Risk	62
3.5.5.	Root Mean Square Error	63

3.5.6.	Noise	63
3.5.7.	Relative Absolute Error	63
3.5.8.	Root Relative Squared Error	63
3.5.9.	Mean Magnitude of Relative Error	64
3.5.10.	Predictive Power	64
3.5.11.	Coefficient of Determination	64
3.5.12.	Accuracy of Estimation	65
3.5.13.	Theil's Statistics	65
3.5.14.	Comparison Criteria Values for all Proposed Models	65
3.6.	Ranking Methodology	66
CHAPTER 4 : EXPERIMENTAL WORK		71-112
4.1.	Failure Data Sets	71
4.1.1.	Dataset #1	72
4.1.2.	Dataset #2	74
4.1.3.	Dataset #3	76
4.1.4.	Dataset #4	80
4.2.	Parameter Evaluation	83
4.2.1.	Least Square Estimation	84
4.2.2.	Maximum Likelihood Estimation	85
4.2.3.	Algorithms used in Evaluating Parameters	85
4.3.	Soft Computing Optimization	85
4.3.1.	Genetic Algorithm	86
4.3.2.	Fuzzy Logic	87
4.3.3.	Neural Network	87
4.3.4.	Support Vector Machines	88
4.3.5.	Particle Swarm Optimization	88
4.3.6.	Simulated Annealing	89
4.4.	Parameter Evaluation and Optimization using Soft Computing Methods and DS#1	89
4.4.1.	Proposed Methodology for Parameter Selection	90
4.4.2.	Parameters of all Proposed Models	91
4.4.3.	Parameters of Existing NHPP Models	92

4.4.4.	Parameter Evaluation using LSQ, GA, SA & PSO	92
4.5.	Parameters Evaluation and Soft Computing Optimization using DS#3	95
4.6.	Parameters Evaluation and Soft Computing Optimization using DS#4	
4.7.	Selection of Best Model from Ten Candidate Models using DS#1	
4.8.	Ranking Candidate Models using Optimized Parameters Values for DS#3	105
4.9.	Ranking Candidate Models using Optimized Parameters Values for DS#4	108
CHAPTER 5 : RESULTS AND DISCUSSION		113-141
5.1.	Model Validation	113
5.1.1.	Failure Estimation	113
5.1.2.	Graphical Representation	115
5.1.3.	Comparison Criteria	116
5.2.	A Statistical Test: t-test	124
5.3.	Ranking Methodology Validation using DS#2	125
5.4.	Validating Developed Models using Ranking Methodology for DS#3	133
5.5.	Validating Developed Models using Ranking Methodology for DS#4	136
5.6.	Parameter Optimization of SKK-28 Model using GA, SA and PSO Methods	139
CHAPTER 6 : CONCLUSIONS AND FUTURE SCOPE		142
6.1.	Conclusion	142

6.2. Scope for Future Research

143

REFERENCES

145

SUMMARY

- **Chapter 1:** The problem aspect of research regarding software reliability modeling is defined. Based on the problem's nature, background, and motivation factor the objectives of the research were formulated, and step-wise methodology is demonstrated using a flow chart. The significance and limitations of the research were explained concerning the conducted study.
- **Chapter 2:** This chapter discusses the literature published in leading journals regarding the topics relevant to research work. A total of four reviews were included in this chapter. Studies of literature concerning big data applications, big data reliability, hybrid model development, and parameter estimation techniques were carried out and presented as review reports in this chapter. findings and gaps in these critical reviews were also outlined in the chapter. Discussion regarding software development life cycle (SDLC) phases and various key terms in the context of reliability models were also penned down in this chapter. The chapter discusses the importance of reliability modeling and the broad classification of software reliability models into different categories. Because of the widespread use of NHPP models in developing hybrid models, a total of seventeen NHPP models were tabulated in this chapter stating their mean value function, and Intensity function.
- **Chapter 3:** This chapter presents the development of 33 hybrid reliability models specifying their mathematical formulation. Models were further filtered based on their estimation accuracy. An estimation function is formulated to determine the best-performing model for the same data set using estimation capabilities up to the five-point difference between the experimented and evaluated values. A ranking methodology based on estimation accuracy and section criteria is also formulated to rank the models.

- **Chapter 4:** This chapter includes the datasets description of all the fault data used in the research. The parameters of hybrid models as well as existing well-known NHPP models were evaluated. Parameters were evaluated using MLE and LSE statistical methods and various soft computing optimization techniques like GA, SA, and PSO. This chapter demonstrates a criterion set used to select the best model by comparing them against thirteen criteria values. Experimental work related to best model selection using the weighted selection method for DS#1 and ranking methodology to rank the models based on estimation accuracy and section criteria for DS#3 and 4 is also included in this chapter.
- **Chapter 5:** In this chapter developed model is validated using comparison criteria, Estimation accuracy, and a t-test. Graphical representation is included to have a better understanding of models. Validation of the proposed raking methodology is included to determine the efficiency of the method in comparison to the existing ones.
- **Chapter 6:** This chapter concludes the research work by presenting the obtained results during the study and gives ideas regarding the future scope of the work.

LIST OF TABLES

Table 2.1.	MVF and InF of existing NHPP models
Table 2.2.	Criteria used by researchers for the reliability model's comparison
Table 3.1.	Proposed hybrid software reliability models
Table 3.2.	Comparison criteria of proposed candidate models
Table 3.3.	Weight values assigned to the model's individual criteria rank
Table 3.4.	Weighted estimation vector
Table 4.1.	Fault datasets used by researchers in developed models
Table 4.2.	Failure data set collected using big data analysis
Table 4.3.	Data set of a middle-size project
Table 4.4.	Real-time control application dataset
Table 4.5.	Failure count data
Table 4.6.	Evaluated parameters value using MLE of proposed hybrid models
Table 4.7.	Evaluated values of parameters of traditional models using MLE
Table 4.8.	Parameters evaluated of proposed ten best models using LSE
Table 4.9.	Parameters evaluated of proposed ten best models using GA
Table 4.10.	Evaluated parameters of proposed ten best models using SA
Table 4.11.	Parameters evaluated of proposed ten best models using PSO
Table 4.12.	Parameters value of proposed models using LSE, GA, SA and PSO
Table 4.13.	Parameters value of existing models using LSE, GA, SA and PSO
Table 4.14.	Parameters value of proposed models using LSE, GA, SA and PSO
Table 4.15.	Parameters value of existing models using LSE, GA, SA and PSO
Table 4.16.	The deviation between observed and estimated values
Table 4.17.	Weighted estimation function
Table 4.18.	Weight matrix corresponding to criteria measures rank
Table 4.19.	WRM using LSE
Table 4.20.	WRM using GA
Table 4.21.	WRM SA
Table 4.22.	WRM using PSO

Table 4.23.	Rank matrix using LSE
Table 4.24.	Rank matrix using GA
Table 4.25.	Rank matrix using LSE
Table 4.26.	Rank matrix using GA
Table 4.27.	Average optimized rank determination of candidate models
Table 4.28.	WRM using LSE
Table 4.29.	WRM using GA
Table 4.30.	WRM using LSE
Table 4.31.	WRM using GA
Table 4.32.	Rank matrix using LSQ
Table 4.33.	Rank matrix using GA
Table 4.34.	Rank matrix using LSE
Table 4.35.	Rank matrix using GA
Table 4.36.	Average optimized rank determination of candidate models
Table 5.1.	Intensity function (estimation value) calculation of models
Table 5.2.	t-test for observed and estimated values
Table 5.3.	t-test for cumulative observed and estimated values
Table 5.4.	Parameters value of NHPP model using DS#2
Table 5.5.	Criteria's value for NHPP models
Table 5.6.	Ranking of models based on individual criteria measure
Table 5.7.	Weight matrix
Table 5.8.	Weighted rank matrix
Table 5.9.	Ranking of existing models based on developed methodology
Table 5.10.	WRM using LSE
Table 5.11.	WRM using GA
Table 5.12.	WRM using SA
Table 5.13.	WRM using PSO
Table 5.14.	Rank matrix using LSE
Table 5.15.	Rank matrix using GA
Table 5.16.	Rank matrix using SA
Table 5.17.	Rank matrix using PSO

- Table 5.18. Given below gives the average rank of selected models
- Table 5.19. WRM using LSE
- Table 5.20. WRM using GA
- Table 5.21. WRM using SA
- Table 5.22. WRM using PSO
- Table 5.23. Rank matrix using LSE
- Table 5.24. Rank matrix using GA
- Table 5.25. Rank matrix using SA
- Table 5.26. Rank matrix using PSO

LIST OF FIGURES

- Figure 1.1. Adopted research methodology for the study
- Figure 3.1. Flowchart of ranking methodology
- Figure 4.1. Number of failures and cumulative failures with time (days) of DS1
- Figure 4.2. Number of failures and cumulative failures with time (days) of DS2
- Figure 4.3. Failures and cumulative failures plot concerning time for DS3
- Figure 4.4. Number of failures and cumulative failures with the time of DS4
- Figure 5.1. Observed values and estimation values comparison between models
- Figure 5.2. Cumulative experimental and estimation value of model SKK-28
- Figure 5.3. Comparison of SKK with other models for bias value
- Figure 5.4. Comparative view of MSE values of various models
- Figure 5.5. Comparative view of MAD values of various models
- Figure 5.6. Comparative view of PRR values of various models
- Figure 5.7. Comparative view of NOISE values of various models
- Figure 5.8. Comparative view of RRSE values of various models
- Figure 5.9. Comparative view of RAE values of various models
- Figure 5.10. Comparative view of RRSE values of various models
- Figure 5.11. Comparative view of MMRE values of various models
- Figure 5.12. Comparative view of PP values of various models
- Figure 5.13. Comparative view of R^2 values of various models
- Figure 5.14. Comparative view of AE values of various models
- Figure 5.15. Comparative view of TS values of various models
- Figure 5.16. Min value of various models for all measures in comparison criteria
- Figure 5.17. Individual criteria measures rank of all models
- Figure 5.18. Weighted rank sum calculations for existing models
- Figure 5.19. Estimation function rank sum calculations for existing models
- Figure 5.20. Models ranking using existing method in literature
- Figure 5.21. Models ranking using developed methodology
- Figure 5.22. Criteria set values of SKK-28 using various methods

Figure 5.23. Maximum number of minimum criteria values of SKK-28

Figure 5.24. Estimation capacity of SKK-28 using various methods

ABBREVIATIONS USED

3D_SDE	Three Dimensional - Stochastic Differential Equation
ABC	Artificial Bee Colony
ABM	Architecture-Based Models
ACO	Ant Colony Optimization
AE	Accuracy of Estimation
ALO	Ant Lion Optimization
ARIMA	Autoregressive Integrated Moving Average
BA	Bat Algorithm
BDA	Big Data Applications
BDR	Big Data and Reliability
BDSCAN	Density-Based Spatial Clustering of Applications With Noise
BLS	Boneh-Lynn-Shacham Signature
BPNN	Back Propagation Neural Network
CFSL	Considering Fault Severity Levels
CLARA	Clustering Large Applications
CRAFT	Compile R-Assisted Fault Tolerance
CS	Cuckoo Search
DE	Differential Evolution
DEA	Differential Evaluation Algorithm
DM	Modified Duane Model
EACO	Extended Ant Colony Optimization
EARF	Exponential Reliability Function
EPM	Early Prediction Model
FCM	Fuzzy C Means
FFMLP	Feed-Forward Multi-Layer Perceptron
FINF	Failure Intensity Function
FIS	Fuzzy Inference System
FL	Fuzzy Logic

FMNN	Feed-Forward Multi-Layer Neural Network
FR	Failure Rate
FTA	Fault Tree Analytics
GA	Genetic Algorithm
GG	Generalized Goel Model
GM	Gompertz-Makeham Model
GMDH	Group Method of Data Handling
GMPZ	Gompertz Model
GO	Goel-Okumoto Model
GPS	Geographical Positioning System
GSA	Gravitational Search Algorithm
GWO	Grey Wolf Optimizer
HBBM	Hybrid Black Box Models
HGP	Hybrid Gaussian Process
HWBM	Hybrid White Box Models
IDBM	Input Domain Based Models
INF	Intensity Function
IOT	Internet of Things
JM	Jelinski-Moranda Model
KF	Kalman Filter
K-NN	K-Nearest Neighbours Algorithm
LGC	Logistic Growth Curve Model
LSE	Least Square Estimation
LSQ	Least Square Method
LVM	Littlewood- Verrall Model
MAD	Mean Absolute Deviation
MD	Duane Model
MEOP	Mean Error of Prediction
MERF	Multinomial Exponential Reliability Function
MGP	Multivariate Gaussian Process

MHT	Multiple Huffman Table
MLE	Maximum Likelihood Estimation
MMRE	Mean Magnitude of Relative Error
MO	Musa-Okumoto Model
MSE	Mean Square Error
MTBF	Mean Time Between Failure
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
MVF	Mean Value Function
NHPP	Non-Homogeneous Poisson Process
NN	Neural Network
OSS	Open System Software
PAM	Partitioning Around Medoids
PNZ	Pham-Nordmann-Zhang Model
PP	Predictive Power
PRR	Predictive Ratio Risk
PSF	Performance Shaping Factors
PSO	Particle Swarm Optimization
PZ	Pham-Zhang Model
PZIFD	Pham-Zhang Imperfect Fault Detection Model
R^2	Coefficient of Determination
RAE	Relative Absolute Error
RBF	Radial Bias Function
RDBMS	Relational Database Management System
RF	Reliability Function
RF	Random Forest
RM	Reliability Model
RMSE	Root Mean Square Error
RMT	Random Matrix Theory
RRSE	Relative Root Square Error

RRSQ	Relative Root Squared Error
SA	Simulated Annealing
SCT	Soft Computing Techniques
SDLC	Software Development Life Cycle
SFLA	Shuffled Frog Leaping Algorithm
SHARPE	Symbolic Hierarchical Automated Reliability and Performance Evaluator
SKK-1	Sharma, Kumar and Kaswan-1
SKK-2	Sharma, Kumar and Kaswan-2
SKK-3	Sharma, Kumar and Kaswan-3
SKK-4	Sharma, Kumar and Kaswan-4
SKK-5	Sharma, Kumar and Kaswan-5
SKK-6	Sharma, Kumar and Kaswan-6
SKK-7	Sharma, Kumar and Kaswan-7
SKK-8	Sharma, Kumar and Kaswan-8
SKK-9	Sharma, Kumar and Kaswan-9
SKK-10	Sharma, Kumar and Kaswan-10
SKK-11	Sharma, Kumar and Kaswan-11
SKK-12	Sharma, Kumar and Kaswan-12
SKK-13	Sharma, Kumar and Kaswan-13
SKK-14	Sharma, Kumar and Kaswan-14
SKK-15	Sharma, Kumar and Kaswan-15
SKK-16	Sharma, Kumar and Kaswan-16
SKK-17	Sharma, Kumar and Kaswan-17
SKK-18	Sharma, Kumar and Kaswan-18
SKK-19	Sharma, Kumar and Kaswan-19
SKK-20	Sharma, Kumar and Kaswan-20
SKK-21	Sharma, Kumar and Kaswan-21
SKK-22	Sharma, Kumar and Kaswan-22
SKK-23	Sharma, Kumar and Kaswan-23
SKK-24	Sharma, Kumar and Kaswan-24

SKK-25	Sharma, Kumar and Kaswan-25
SKK-26	Sharma, Kumar and Kaswan-26
SKK-27	Sharma, Kumar and Kaswan-27
SOTERIA	Socio-Technical Risk Analysis
SRA	Software Reliability Analysis
SRGM	Software Reliability Growth Models
SRHM	Software Reliability Hybrid Models
SRM	Software Reliability Model
SSA	Sparrow Search Algorithm
SSE	Sum Of Squared Errors
SVM	Support Vector Machine
SWIFT	Software-Only, Transient-Fault-Detection Technique
TS	Theil's Statistics
TSP	Traveling Salesman Problem
UGP	Univariate
WPA	Wolf Pack Algorithm
YDM	Modified Duane Model (Md)
YE	Yamada Exponential Model
YIDM1	Yamada Imperfect Debugging Model-1
YIDM2	Yamada Imperfect Debugging Model-2
YIM	Yamada Delayed S-Shaped Model
YR	Yamada Rayleigh Model
ZTP	Zhang-Teng-Pham

LIST OF PUBLICATIONS

1. Sharma, S., Kumar, N., & Kaswan, K. S. (2021). Big data reliability: A critical review. *Journal of Intelligent & Fuzzy Systems*, 40(3), 5501-5516. doi: 10.3233/JIFS-202503(**ESCI, SCOPUS, WOS**).
2. Sharma, S., Kumar, N., & Kaswan, K. S. (2021). Ranking of Reliability Models based on Accurate Estimation and Weighted Function. *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 1679-1685, doi: 10.1109/ICAC3N53548.2021.9725534 (**SCOPUS**).
3. Sharma, S., Kumar, N., and Kaswan K. S. (2023). Hybrid Software Reliability Model for Big Fault Data and Selection of Best Optimizer Using an Estimation Accuracy Function. *Int. J. Recent Innov. Trends Comput. Commun.*, 11(1), 26–37, doi: 10.17762/ijritcc.v11i1.5984 (**Scopus**)
4. Sharma, S., Kumar, N., & Kaswan, K. S. (2022). Development of Hybrid Reliability Model using Big Fault Data. (**Communicated**)
5. Sharma, S., Kumar, N., & Kaswan, K. S. (2022). Evaluation of Parameters using Statistical and Soft Computing Methods. (**Communicated**)
6. Sharma, S., Kumar, N., & Kaswan, K. S. (2019). Big Data Applications in Real World. *UGC Sponsored National Conference on Emerging Trends in Information Technology* (pp. 158-173). Delhi: Bloomsbury

CHAPTER 1

INTRODUCTION

The literature on software reliability is replete with reports of the development of numerous reliability models. These models, suggested by several scholars, relied on certain assumptions and limitations. We proposed various hybrid Non-Homogeneous Poisson Process models developed using existing models for software reliability.

1.1. BACKGROUND OF THE PROBLEM

People have become dependent on using various forms of software in their day-to-day lives due to the rapid development of technology in the modern world. The usage of social media increases both the size and complexity of data as well as the software required to process such a huge amount of information. Because of the economic digitization and usage of software in business, it became very challenging and hard to handle complex software. Any software failure will be fatal for the company as well as the client consequently, before releasing the product to the public, developers must do rigorous testing to ensure the product's reliability. The software development process mainly emphasizes the product time estimation, human resources required, the structure of the product, technologies, and tools required, Quality assessment, and feasibility study to deliver high quality and well-managed controlled product within the stipulated time and of a specific standard. It is a time-consuming task and involves factors that are required to be controlled and accessed to determine the quality of the final product but it will eliminate many problems resulting due to the software failure like service termination or financial loss. Therefore, it is of utmost importance to eliminate as many faults in the software as possible before its release. The need to deliver a high-quality product becomes a major concern in the industry. Product quality is the only factor that determines its success in the market and can be identified with its reliability. The

development of a system became complex and costly due to technological advancement thus one needs to address the criteria regarding security, development cost, and reliability during the development phase to ensure a defect-free, cost-effective, and reliable final product. Moreover, a reliability assessment is required on the upgraded versions of the already existing systems. Existing systems are continuously monitored for any possible fault and additional components are added in the new version to address the resulting issues which further required an up-gradation. As the complexity of a system increases so do its functionality and capabilities but since the reliability is inversely proportional to the degree of software complexity achieving a balance between complexity and reliability became difficult. Although rigorous testing is carried out to remove software faults they cannot be totally removed.

1.2. PROBLEM STATEMENT

Hidden bugs in the system are the reason for software failures. A universal model that gives a good fit for different types of datasets and works accurately in all environmental conditions and circumstances. Widespread usage of big data and related technologies and tools give rise to the development of many new reliability models predicting software faults resulting due to external interactions arises.

When trying to assess a system's dependability, the two most pressing issues are appropriate model selection and parameter evaluation. If there isn't already a good model available, we'll have to create one that takes into account all the factors outside the system that prevent it from working as intended.

Parameter estimation in the mathematical models built to describe a large number of interactions is typically complex and non-linear. Furthermore, failure data from a reliable source is required for parameter estimation, but it is very difficult to obtain data from the companies due to their reluctance in releasing fault data regarding their product. Due to the scarcity of new required data researchers are bound to use published or previously appeared fault data for the development of their model.

Researchers are experimenting with various combinations of existing models in order to develop new hybrid models. Parameter evaluation of such hybrid models based

on mathematical equations is very difficult. Their non-linearity and complexity make the statistical evaluation of parameters challenging and difficult. The development of such mathematical models to assess reliability further depends on accurate prediction and parameter optimization based on experimental data. In order to avoid economic loss and product failure, it is crucial to keep an eye on a software product's reliability.

The goal of modeling is to anticipate the issues arising due to existing models and try to come up with a model to eliminate them. No single model is best for all data sets, so reliability analysis should be carried out for several candidate models for a specific data set and we must choose the best candidate model giving promising results.

Non-Homogeneous Poisson Process (NHPP) models can be coupled to create new hybrid NHPP models, which is one of their distinguishing features. NHPP models represent failure occurrences over a period of time and are simple to implement. The emphasis here is on making an accurate determination of the mean value function of the hybrid NHPP model that has been built.

1.3. MOTIVATION

Usage of software has increased many folds and so have its complexity and size. With the change in technology and disposal of the huge amount of data, software developers need to deliver a good quality product. Apart from the fault resulting due to wrong specification or errors in coding, a software product might fail due to hardware malfunctioning and also because of incorrect data entry. Due to its high velocity, volume, and variety, big data necessitates a distributed, high-capacity storage system with a fast-accessing mechanism to avoid software mistakes caused by hardware failure. Similarly, because of unfamiliarity with specific software and an abundant amount of data to handle, give rise to errors in software because of human negligence.

This work is motivated by the need to construct a hybrid model employing a combination of NHPP models to deal with faults in software originating from both pure software failure and hardware failure/human involvement.

1.4. NATURE OF THE PROBLEM

To develop a hybrid model various combinations of NHPP models must be formed that can handle pure software errors, hardware-induced errors, and user-induced errors. The next step is to use statistical methods to determine each model's parameters. It was necessary to formulate an estimation function that could anticipate the prediction capabilities of the created batch of models in order to select the best-performing model from the set of candidates. The fault data set of the big-data analytical module is then used to verify the selected one.

The parameter's values were obtained using statistical methods under specific constraints on initial guess values, so we need to optimize their values by employing various optimization results to improve the success rate of the developed model.

1.5. OBJECTIVES OF THE RESEARCH

Based on the literature review for this research work, it was found that there is a lack of hybrid prediction models which takes into account the induced errors due to hardware and user intervention. Most researchers use traditional reliability models or hybrid models with existing data sets without identifying the external interactions that influence the reliability of software directly or indirectly. The following were our primary goals as we set out to create a new hybrid model that makes use of big-fault data and is powered by soft computing.

- To analyse the existing techniques for software reliability analysis for big data.
- To develop and propose a hybrid reliability model using big fault data.
- To optimize the result using appropriate soft computing methods.

1.6. RESEARCH METHODOLOGY

A research methodology is a plan outlining the specific actions to be taken in order to complete a study. The initial phase in the methodology is the extensive literature survey to have an understanding of recent developments in the research area and determine the

gaps in published research so far. The literature review helps us to have an understanding of the problem and we formulated the objectives based on our study. Models were developed after reviewing various reliability models their performance and constraints. The parameters of developed hybrid NHPP models were evaluated using statistical techniques. Best-performing model is selected based on specified measures which then need to be validated. Lastly, the result is optimized by using various soft computing algorithms for parameter optimization. Figure 1.1 depicts the methodology steps graphically.

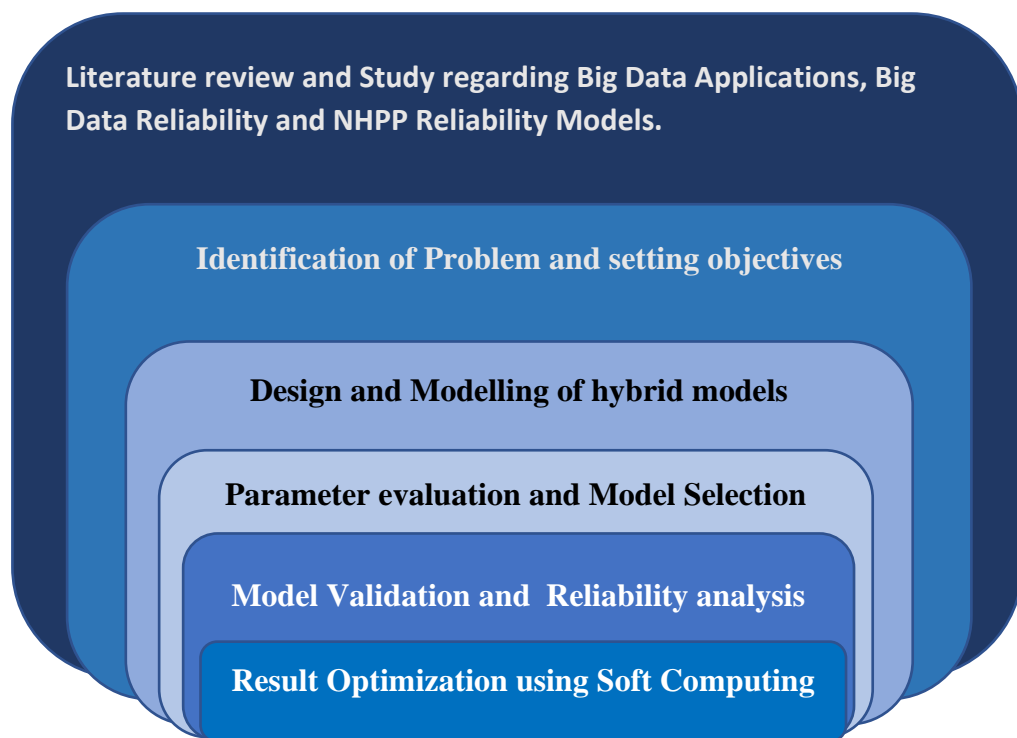


Figure 1.1. Adopted research methodology for the study

1.7. SIGNIFICANCE OF RESEARCH

This research utilizes the fault data from the big data analytical system project in conjunction with the various NHPP models and other hybrid models to create a reliability model for improved reliability prediction. Various Big data applications and

reliability techniques were reviewed. soft computing methods (SCM) were utilised to give accurate predictions using the developed model.

- I. Examining how big data has been used in the past can shed light on which industries are being impacted and provide inspiration for finding fresh use cases for big data analysis.
- II. Reliability assessment related to Data, Hardware, and Software of big data gives an idea regarding the work done, limitations, and scope of research in these fields.
- III. Identification of new techniques helps the researchers in utilizing the appropriate methods for the reliability assessment of big data systems.
- IV. The developed ranking methodology can be applied to rank reliability models based on accurate estimation.
- V. Developed 33 hybrid models can be utilized for reliability estimation using a different data set.
- VI. The suggested technique can be utilized in future studies to determine which model is the most accurate estimator up to a given error threshold.
- VII. Comparison criteria having thirteen values to choose the best model among the group of various candidate models can be utilised for the model's comparison.

1.8. ORGANIZATION OF THESIS

The thesis consists of eight chapters including this chapter. Undermentioned is The outline of various chapters included in the study.

- **Chapter 1:** The problem aspect of research regarding software reliability modeling is defined. Based on the problem's nature, background, and motivation factor the objectives of the research were formulated, and step-wise methodology is demonstrated using a flow chart. The significance and limitations of the research were explained concerning the conducted study.

- **Chapter 2:** This chapter discusses the literature published in leading journals regarding the topics relevant to research work. A total of four reviews were included in this chapter. Studies of literature concerning big data applications, Big data reliability, hybrid model development, and parameter estimation techniques were carried out and presented as review reports in this chapter. findings and gaps in these critical reviews were also outlined in the chapter. The chapter discusses the importance of reliability modelling and the broad classification of software reliability models into different categories. Because of the widespread use of NHPP models in developing hybrid models, a total of seventeen NHPP models were tabulated in this chapter stating their mean value function, and Intensity function.
- **Chapter 3:** This chapter presents the development of 33 hybrid reliability models specifying their mathematical formulation. Models were further filtered based on their estimation accuracy. An estimation function is formulated to determine the model that performs better than others, using estimation capabilities up to the five-point difference between the experimented and evaluated values. A ranking methodology based on estimation accuracy and section criteria is also formulated to rank the models.
- **Chapter 4:** This chapter includes the datasets description of all the fault data used in the research The parameters of hybrid models as well as existing well-known NHPP models were evaluated. Parameters were evaluated using two standard statistical methods and the three most commonly used soft computing methods. In this section, we show how to select the best model by comparing them to 13 different criteria values taken from the literature. Experimental work related to best model selection using the weighted selection method for DS#1 and ranking methodology to rank the models based on estimation accuracy and section criteria for DS#3 and 4 is also included in this chapter.
- **Chapter 5:** In this chapter developed model is validated using comparison criteria, Estimation accuracy, and a t-test. Graphical representation is included

to have a better understanding of models. The efficiency of the proposed raking approach is evaluated in comparison to the already used methods.

- **Chapter 6:** This chapter concludes the research work by presenting the obtained results during the study and gives ideas regarding the future scope of the work.

CONCLUSION

Modern society's dependency on software usage becomes a crucial concern and motivated us to do reliability analyses concerning complex systems. In this chapter, we identify various key factors related to model development concerning big data. We developed the Problem Statement, listed our goals, and established the parameters of our study. The structure of the thesis is summarized at the end.

CHAPTER 2

LITERATURE REVIEW

Reliability models are abundant in the literature. Still, not all models can exactly depict reality since while determining the model parameters, there is always the possibility of uncertainty. Also, model selection depends on the evaluated parameter's value, comparison criteria for model selection, and used fault data set. A detailed study regarding these all aspects was done and presented in this chapter.

2.1. BIG DATA

Big data is increasing exponentially, we need sophisticated and complex techniques to maintain, operate, and analyze it effectively. The term "big data" was coined by the Oxford Dictionary as data that refers to the collection and storage of massive amounts of information in order to analyze it computationally to identify patterns, trends, and relationships, often pertaining to human behavior and interactions. The 3Vs of big data are the volume, velocity, and variety of the data being collected. IBM scientists' veracity adds a fourth dimension, representing redundant data. The dimensions of big data characteristics keep increasing and till now up to 17 characteristics were identified.

2.1.1. Big Data Analytics

Hadoop, NoSQL, Hive, Apache Cassandra, etc. are all examples of the types of frameworks that can be used to store "big data." In order to retrieve meaningful information, the stored data must be analyzed using the appropriate analytical technique (in-memory analytics, in-database analytics, descriptive analytics, predictive analytics, etc.) to uncover hidden patterns and find correlations necessary for business decisions. Analyst reveals their knowledge after analyzing the data and helps an organization improve efficiency, increase profit, and reduce cost by making decisions based on that insight. Big data delivers us the technology that provides insight in real-time, and many

private organizations, NGOs, and government sectors are benefited from the offered information. Organizations are adopting big data analytics to make better decisions, and thus the need to employ new methods, tools, and techniques required for data analysis is also growing. Big data collection, storage, processing, analysis, and prediction are not possible using traditional databases and infrastructure. Acquiring big data tools and techniques helps in managing data and increasing the organization's capability to capture required data. Fast data loading, efficient storage utilization, fast query processing, and adaptivity to divergent workload patterns are the basic requirements for analytical processing.

2.1.2. Big Data Challenges

The rate at which new methods and tools for storing and processing Big Data are being developed is outstripped by the rate at which Big Data itself is being generated. Increased usage of social media, e-commerce, mobile phones, video, and sensors generates a massive amount of data that will create an enormous volume that must be captured, cleaned, transferred, searched, shared, stored, analyzed, and visualized. Based on the data life cycle, Akerkar (2014) and Zicari (2014) categorize the difficulties into data, process, and management.

- **Data:** It solves problems brought on by data's volume, velocity, variety, validity, etc.
- **Process:** It addresses the challenges related to employing techniques to capture, analyze, integrate, transform, and deliver results regarding data.
- **Management:** Data privacy, security, and ethical usage come under management challenges.

In addition, "D. P. Acharjya and Kauser Ahmed P" classify the challenges related to big data in four broad categories. Sentiment analysis, text mining, Anomalies detection in the Human Ecosystem, distributed storage, data visualization, scalability of information security, and content validation are addressed in these categories.

2.2. UNDERSTANDING RELIABILITY

A reliable system consists of factors like correctness of design, system implementation, and reliable execution. It determines system quality and refers to the consistency of measurements using the same subject and method. A system is reliable if it delivers the same result for the same set of data inputs at any time. Whereas if the system delivers different outputs for the same set of parameter values using the same method every time or most of the time, then the system is termed unreliable.

In today's technological world high-quality software is in demand because of its extensive use. The software quality is measured by its reliability and must be controlled before its release. Several models were developed to determine a system's reliability. Researchers keep developing new models to achieve better predictions and eliminate the problems resulting from the use of existing models. Since its an era of technological advancement, therefore, change in the environment, functioning, and technology usage also give rise to new models' development, making better and more reliable predictions.

2.2.1. Software v/s Hardware Reliability

Failure of hardware, whether from defective components or simple aging, can have a negative impact on hardware reliability. Software systems do not degrade as they never wear out with time. Undermentioned are some of the characteristics regarding the difference in software and hardware reliability.

- Hardware reliability is time-dependent but not software reliability.
- Hardware degradation is due to its wear out with time or manufacturing defects, whereas software becomes obsolete due to changes in technology or the environment for which it was built due to changed or modified requirements in terms of expectations.

- Hardware failures are often physical in nature, whereas software problems can occur at any time without warning and are typically the result of erroneous requirements during the design process or incorrect coding.
- Software is developed, whereas hardware is manufactured.
- Hardware degrades due to environmental conditions (rust and moisture) and affects its reliability or functioning but not software reliability is not dependent on such conditions.
- Hardware reliability can be predicted by physical measures like the amount of degradation and wear and tear, but we cannot predict software reliability using physical measures.
- Hardware reliability can be improved by replacing the defective part, whereas software reliability can be improved by eliminating errors.
- Redundancy can improve hardware reliability, but modification can improve software reliability.

2.2.2. Software Development Life Cycle (SDLC)

Testing software against varied inputs and environments and including software growth models in the SDLC is the only way to ensure the high quality of software. Still, the software can give below-desired performance as the measure of reliability is not deterministic but probabilistic. SDLC is a process consisting of five phases required to deliver high-quality software. Undermentioned are the five phases of SDLC and their brief description.

- **Requirements Phase:** The requirement phase is the first phase of SDLC, where requirements were underlined after taking customer input. A report is generated regarding the viability and feasibility of the product after having lengthy, and many rounds of discussions with customers, subject

experts, and sales heads concerning their needs, problems, scope, and economic constraints. Their requirements are narrowed down to match their specific domain and mapped to business requirements.

- **Design Phase:** The software requirement specification document prepared in the requirements phase is converted into a design plan to implement the requirements. All concerned parties review the generated document called design specifications for anomalies and give their suggestions and feedback to derive the system architecture. After minutely analysing the requirements, a design document specification (DDS) including all related technical and non-technical details is prepared, including various processes required to meet the desired software requirements. The whole software system architecture is designed to be comprised of multiple modules while their modular structure, internal design, data flow, and interactions within and outside the system were clearly defined during this phase. All architectural modules must be defined and clearly described using a design approach and the best possible DDS.
- **Implementation Phase:** All modules were coded based on design specifications finalized during the design phase. During this stage, DDS is used to create the product or put it into action. A modular approach to coding is formulated structurally suited for the developer, consumer, and application environment. A structural formulation of the problem is not only comfortable for developers but also easy to code and debug.
- **Testing Phase:** Testing is employed using varying techniques suitable for the stage in which it's incorporated and is generally done in all phases of SDLC. Testing is a must to determine the product's viability in solving the problem and issues stated during the requirement phase. Various types of testing were carried out to determine defects or faults to fix them. Unit testing determines the functionality of an individual module by giving

required input and determining the possible outcome. At the same time, Integral testing determines the system's functionality after combining various modules and checking the outflow and inflow of the composite unit as a whole. Faults were reported, located, fixed, repeated, and rested in this phase until the software product reached the desired quality.

- **Validation Phase:** After testing, various methodologies were used to validate the product. Different methodologies required the feedback of various levels of users and stakeholders in discrete or continuous modes for external validation. According to deployment methodologies, the software is released or deployed for internal validation by a limited section of users in the internal environment. Based on stage 1 feedback, external validation involves all the stakeholders and asks their opinion regarding product satisfaction. Based on internal and external validation feedback, the software product is either released or modified for further enhancement.

Due to the requirement for a high-quality product, measurement and reliability prediction is in great demand before release. To guarantee better quality software should be documented, modularly designed, well-tested, and validated before release. Recently, SDLC has made use of existing software reliability growth models or the creation of new models appropriate for specific applications in order to increase quality. Software reliability does not guarantee fault-free software; it just determines the quality of the current version of the system.

2.3. IMPORTANCE OF MODELLING

Modeling is required for the SRA of a system. Fault data collected during testing is used to assess the system's reliability. To be regarded as good, a model must be straightforward, generally applicable, and capable of making accurate predictions of failure in the future. These models were used to determine the phase development state and to monitor and control the enhancement of the product due to new features or

changes in design. Developing a new model required the formulation of a mathematical expression that satisfies the basic assumptions and the formulation of a failure function in terms of factors affecting the product's reliability. A model is nothing but a mathematical version or simulation of an actual situation with certain assumptions. While developing models, the operating environment and testing environment conditions are generally assumed to be the same. The removed fault is typically mathematically formulated in most models under the premise that no new problems are introduced during fault removal.

In most models, the fault removal process is based on a stable operational and software profile, while in others, it may introduce additional faults. The parameters required for predicting the reliability must be evaluated correctly, using enough failure data to estimate future failure phenomena accurately. Incorporating a change in a model for new faults is generally impractical due to its complexity. It requires a reformulation of function, recalculation of parameters, and fine new fault data collection.

Design and validation of a model require a fault dataset, parameter estimation, and performance criteria. The input data, mainly called fault data, is needed to determine the reliability and compare the prediction capability of the developed model.

Models can be thought of as mathematical explanations of the problem of a system's unreliability. The performance of a constructed model is evaluated based on a number of different parameters, including the MVF, the InF, the reliability factor, and the cumulative faults. For quantitative measurement of product quality, the value of the parameter in the mathematical function needs to be evaluated. A confidence interval should be used to represent the range of parameters value.

Software reliability modeling (SRM) provides the undermentioned information regarding the quality management of a product.

- Product reliability when testing ends.
- The time needed to reach the stated product reliability
- Resultant reliability growth because of testing.
- Field reliability of the product

Error seeding, time domain, and data domain are three fundamental approaches to SRM.

2.4. ELEMENTS OF RELIABILITY MODELLING

System reliability is composed of two factors hardware reliability and software reliability. A hardware failure occurs due to physical deterioration with time, wear and tear, and improper maintenance, whereas software failure is mainly because of a coding error. Software reliability, an essential element in customer satisfaction, is used as an important factor in quantifying error occurrences in a system responsible for system failure. However, data reliability is also arising as an essential factor because of the extensive use of big data. When evaluating the quality of the system, we need to take into account both the dependability of the hardware and the reliability of the software. Only then can we establish the overall reliability of the system. In spite of the fact that the nature of both software and hardware dependability is distinct, it is possible for us to build theories of reliability that are compatible with both types.

2.4.1. Failures, Outages, and Faults

Failure is defined as the non-functioning of a system because of an undetected error that causes the entire system to cease functioning. The services delivered by a software system are nothing but a sequence of time-dependent outputs based on the initial specification from which the system was derived. The user can choose various levels of failures and label them as minor, major, or catastrophic depending upon the severity of the impact on rendered services by the system.

An outage can be caused due to external factors like power failure, lightning, or fire, and internal factors like human error, software error, or hardware malfunctioning. The outage also comes under the category of failure. It is defined as a particular failure caused due to the degradation of services provided to the customer over a period called outage duration.

A fault is defined as defective programs (because of missing, wrong, or additional instruction or instruction set) responsible for system-wide failure upon execution. Faults are generally referred to as "bug" results because of incorrect code due to missing or inaccurate instruction that causes system failure. There can be a single

fault due to an untrue statement, or a fault can trigger a sequence of incorrect information. Whatever may be the case, a fault always results due to incorrect coding of a program.

2.4.2. Calendar, Execution, and Clock Time

Calendar time is a time sequence, including when the computer is idle. In contrast, execution time refers to the system's running time while processing tasks during which instructions are executed.

A clock time is the total time elapsed in a program execution from the start till the end, including wait time. Calendar time was required by the models, which used calendar time instead of execution time by converting execution time prediction to dates. It was predicated on the hypothesis that the amount of time needed to complete a task is proportional to the number of available resources.

2.5. MEASURING SOFTWARE RELIABILITY

Undermentioned are some methods used to measure software reliability.

2.5.1. Reliability Function (RF)

The probability that a piece of software will continue to function correctly despite the presence of certain external factors and for a predetermined period of time is what we mean when we talk about software reliability. In the event that the time interval ranges from 0 to t and the probability of failure is given by P(f(t)), then

$$\mathbf{RF(t) = 1 - P(f(t))} \quad \mathbf{(2.1)}$$

2.5.2. Mean Time to Failure (MTTF)

If a fault data set exists consisting of n values at times $t_1, t_2, t_3, \dots, t_n$, then MTTF, the mean of the next failure interval is defined as

$$\text{MTTF} = \frac{1}{n} \sum_{i=1}^n t_i \quad \text{----(2.2)}$$

2.5.3. Mean Time to Repair (MTTR)

Indicated as the interim for the system's successful recovery. Repair time includes tracking and reporting errors responsible for failure.

2.5.4. Mean Time Between Failure (MTBF)

In most cases, the term "hardware reliability" refers to the process of repairing or replacing a hardware component because it has stopped performing properly or has been worn out. Mathematically we can represent it as

$$\text{MTBF} = \text{MTTF} + \text{MTTR} \quad \text{----(2.3)}$$

Availability is the time fraction during which software modules or systems are functionally acceptable. Mathematically in terms of MTTF and MTTR, we can represent them as

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \quad \text{---(2.4)}$$

2.5.5. Failure Rate (FR)

Usually, in reliability modelling, the system failure is described by FR. It is defined as the rate at which failures occur in an interval that spans t and t plus δt .

$$\text{Failure rate} = \frac{F(t+\delta t) - F(t)}{\delta t.R(t)} \quad \text{----(2.5)}$$

2.5.6. Hazard Rate

The hazard rate is the instantaneous FR at a specified time as

$$z(t) = \frac{dF(t)}{dt} \frac{1}{R(t)} \quad \text{----(2.6)}$$

2.6. MODEL CLASSIFICATION

There are several classification schemes available in history, undermentioned is a classification scheme based on different attributes.

- **Time Domain:** classification in accordance with the kind of time, for example, calendar time as opposed to execution time.
- **Category:** Classification based on time duration taken for failures, i.e., finite or infinite.
- Classification based on the functional form
 - **Class:** A finite category failure classification where grouping is according to Failure Intensity Function (FInF) based on time.
 - **Family:** An infinite category classification where the FInF is represented as the anticipated total amount of failures.
- **Type:** classification according to the total number of setbacks encountered up until a certain point in time.
- **Class:** Classification according to the functional form of, i.e., time.

Given below is a brief description of the model's characteristics according to the SDLC classification scheme.

2.6.1. Early Prediction Models (EPM)

These models start addressing the reliability issue either in the requirement phase or the early design phase of the SDLC. The reliability issue can also be handled in the waterfall life cycle model during the detailed design. These models are characterized as being formal, infrequently executable, too insubstantial, and inadequately proper to

be analyzed. There are significantly fewer models in this category, resulting in a modest impact on reliability.

2.6.2. Architecture-based Models (ABM)

This class consists of models that show reliability depending on the architecture of the software. The entirety of the software system is organized into modules, and the reliability is evaluated at each module before being combined into a single score to assess the overall dependability of the system. Architecture-type models are classified as an additive, path-based, and State-based.

- **Additive Models:** Additive Models estimate the system's reliability by considering individual system components' fault data rather than explicitly considering the system's architecture. These models are called additive as they express the systems failure intensity as a sum of individual components' failure intensity. Additive models act like NHPP models while calculating component reliability; thus, system failure must also be NHPP having failure intensity, which is the total intensity of the risk of failure across all components.
- **Path-based Models:** In these models, the failure phenomenon is path-based. Reliability estimation is done using the program's execution path either experimentally or algorithmically. Different execution paths involving components and interfaces are obtained. Multiplying all component reliabilities along the dying path gives us path reliability. The overall system reliability is determined by arithmetically averaging the individual path reliability values.
- **State-based Models:** A control flow diagram is used by these models to represent system architecture and is developed using Markov models. These models are divided into composite or hierarchical classes. To

predict system dependability, the earlier model takes into account both software architecture and failure phenomena, while the latter approach solves the architectural model first and then superimposes the answer with failure behavior.

2.6.3. Software Reliability Growth Models (SRGMs)

SRGM calculates SR by analyzing the system's history of failures. Reliability estimation is done using these failure patterns and data trends. SRGMs are classified into NHPP models and FR models. Many SRGM models were developed and available in the literature, providing a simple and best way to predict software reliability.

NHPP models can be stochastically modelled using cumulative failure time or between models. NHPP models are black-box models, which treat the software as an internal structure with external environment interactions. The fault data collected during testing is utilized to evaluate the reliability factors and parameters of the model. Model parameters are evaluated using inter-failure times. As failure time increases more faults are being captured and amended, but this may not always be since the time between faults is statistically changing and fluctuating. This class of models looks at the time between failures to see if the failure rate has changed as a result of the different error counts. The estimated function dependability is a mission time function that does not decrease with time, where time is the discrete function reflecting program flaws.

2.6.4. Input Domain-Based Models (IDBM)

IDBM evaluates reliability based on the inputs provided to different modules of the software. Various test cases were created to be utilized in the operation phase using the input distribution. It is hard to obtain the Input distribution divided into different equivalence classes that might or might not be linked to the program path. Reliability assessment is done using test cases of an operational program. Program reliability is evaluated when sampled test cases fail during physical or symbolic simulation of the induction field.

2.6.5. Hybrid Black Box Models (HBBM)

Black Box Model (BBM) considers the internal structure as a black box and only considers fault data in reliability estimation. All SRGMs are black-box types, whereas the HBBM combines IDBM and SRGMs.

2.6.6. Hybrid White Box Model (HWBM)

The White Box Model (WBM) considers the internal software architecture while estimating reliability. The hybrid white-box model combines specific features of WBM and BBM.

2.6.7. Non-Homogeneous Poisson Process Models (NHPP)

Stochastic processes, of which NHPP models were a part, were widely employed in establishing hardware dependability. NHPP models were used extensively to measure reliability growth in literature and can be successfully modelled and implemented easily. The mean value function (MVF) is distributed according to the Poisson model for the accumulated number of failures, $m(t)$. Numerous NHPP models can be obtained either by using different MVFs or by modifying existing MVFs of existing NHPP models. Parameter values of models were assessed using statistical techniques including Maximum Likelihood Estimation (MLE) and Least Squares Estimation (LSE). A software failure process can be modelled successfully only after studying the testing process's factors. NHPP models are helpful to determine the software and hardware reliability combined in a model. By summing the MVFs or InFs of many NHPP models, one can generate an NHPP hybrid model. The resultant combined MVF (or InF) represents the MVF of the developed hybrid model. So, we can obtain a hybrid model by combining various NHPP models having significant parameters set, which

gives a good fit but with increased computations and weaker confidence of reliability.

Tabulated below in Table 2.1 are some important NHPP models' MVF and InF.

Table 2.1. MVF and InF of existing NHPP models

Models	MVF	InF
Goel-Okumoto Model (GO)	$m(t) = (1 - e^{-bt})$ $a>0, b>0$	$\lambda(t) = abe^{-bt}$
Generalized Goel Model (GG)	$m(t) = a(1 - e^{-bt^c})$ $a > 0, b > 0, c > 0$	$\lambda(t) = abct^{c-1}e^{-bt^c}$
Musa-Okumoto Model (MO)	$m(t) = a \log(1 + bt)$	$\lambda(t) = \frac{xy}{1 + yt}$
Gompertz Model (GMPZ)	$m(t) = ake^{-bt}$	$\lambda(t) = ab \ln(k) k^{e^{-bt}} e^{-bt}$
Duane Model (DM)	$m(t) = at^b \quad a > 0, b > 0$	$\lambda(t) = abt^{(b-1)}$
Modified Duane Model (MD)	$m(t) = a(1 - (b/b + t)^c)$ $a > 0, b > 0, c > 0$	$\lambda(t) = acb^c(b + t)^{(1-c)}$
Yamada Delayed S-Shaped Model (YDM)	$m(t) = a(1 - (1 + bt)e^{-bt})$ $a \geq 0, b > 0$	$\lambda(t) = a b^2 t e^{-bt}$
Yamada Inflection S-Shaped Model (YIM)	$m(t) = \frac{a(1 - e^{-bt})}{1 + \beta e^{-bt}}$ $a > 0, b > 0, \beta > 0$	$\lambda(t) = \frac{abe^{-bt}(1 + \beta)}{(1 + \beta e^{-bt})^2}$
Logistic Growth Curve Model (LGC)	$m(t) = a/(1 + ke^{-bt})$ $a > 0, b > 0, k > 0$	$\lambda(t) = abke^{-bt}/(1 + ke^{-bt})^2$
Yamada Imperfect Debugging Model-1 (YIDM1)	$m(t) = \frac{ab(e^{pt} - e^{-bt})}{p + b}$	$\lambda(t) = \frac{ab(pe^{pt} + e^{-bt})}{p + b}$

Yamada Debugging Model (YIDM2)	Imperfect Model-2	$m(t) = a(1 - e^{-bt}) \left(1 - \frac{p}{b}\right) + pat$	$\lambda(t) = abe^{-bt} \left(1 - \frac{p}{b}\right) + pa$
Yamada Model (YR)	Rayleigh	$m(t) = a(1 - e^{-r\alpha(1-e^{-\beta t^2})})$	$\lambda(t) = ar\alpha\beta te^{-r\alpha(1-e^{-\beta t^2}) - \beta t^2}$
Yamada Model (YE)	Exponential	$m(t) = a(1 - e^{-r\alpha(1-e^{-\beta t})})$	$\lambda(t) = ar\alpha\beta te^{-r\alpha(1-e^{-\beta t}) - \beta t}$
Pham-Nordmann-Zhang Model (PNZ)		$m(t) = \frac{a(1 - e^{-bt}) \left(1 - \frac{\alpha}{b}\right) + \alpha at}{1 + \beta e^{-bt}}$	$\lambda(t) = \frac{abe^{-bt} \left(1 - \frac{\alpha}{b}\right)}{1 + \beta e^{-bt}} + \frac{ab\beta e^{-bt} \left(1 - \frac{\alpha}{b}\right) ((1 - e^{-bt}) + \alpha t)}{(1 + \beta e^{-bt})^2}$
Pham Zhang Model (PZ)		$m(t) = \frac{1}{1 + \beta e^{-bt}} \left((c + a)(1 - e^{-bt}) - \frac{ab}{b - \alpha} (e^{-\alpha t} - e^{-bt}) \right)$	$\lambda(t) = \frac{b(c + a)(1 + \beta) - \left[\frac{be^{-bt} (1 + \beta e^{-bt})}{\alpha e^{-\alpha t} (1 + \beta e^{-bt})} \right]}{1 + \beta e^{-bt}}$
Pham-Zhang Imperfect Fault Detection Model (PZIFD)		$m(t) = a - a e^{-bt} (1 + (b + d)t + bdt^2)$	$\lambda(t) = ae^{-bt} [bt(b - d) + d(b^2 t^2 - 1)]$
Zhang-Teng-Pham Model (ZTP)		$m(t) = \frac{\alpha}{p - \beta} \left[\left(1 - \frac{(1 + \alpha) e^{-bt}}{1 + \alpha e^{-bt}} \right)^{\frac{c}{b}(p - \beta)} \right]$	$\lambda(t) = \frac{ac}{1 + \alpha e^{-bt}} \left[\left(\frac{(1 + \alpha) e^{-bt}}{1 + \alpha e^{-bt}} \right)^{\frac{c}{b}(p - \beta)} \right]$

2.6.8. Big Data and Reliability

Big data has unique properties that necessitate new forms of specialized gear and software for processing. A clerical error in the software that processes data hardware

subpar performance is the direct outcome of results in findings that are erroneous and compromised, as well as lacking performance (Sharma, Kumar & Kaswan, 2021).

Big data reliability can also be classified as hardware reliability, software reliability, and data reliability. Data reliability is data quality depicted as correctness and completeness when inflows from heterogeneous sources. Software errors can result from misinterpretation of specifications, inadequate testing, a mistake in code, or incorrect usage of the software. The likelihood of mistakes skyrockets in tandem with the exponential expansion in data volume. Therefore, evaluating the data's dependability is essential for producing accurate outcomes. Since the purpose of the data analysis is to help with decision-making, a reliability model should be used to double-check for any errors that might have caused the system to provide inaccurate results.

There are two components of hardware reliability operational and architectural. Modern reliability large data acquired via IoT or sensor devices linked to the system is used in operational reliability to reliably anticipate system performance and failure of equipment in the field. The ability to quickly retrieve data from a large capacity storage media is crucial for big data projects, which determines its architectural reliability and is a crucial factor in the implementation of big data projects. In contrast, hardware reliability is due to a poorly designed system, leading to performance degradation and eventually failing to meet the specified requirements.

2.7. PARAMETER EVALUATION OF RELIABILITY MODELS

In reality, the reliability model is a mathematical statement of the problem with unspecified values for key variables. To determine the system's reliability, we need to find out the parameter's value as a linear or non-linear function of an optimization problem. LSE and MLE are two widely used statistical techniques to determine parameters in reliability modelling. LSE determines the minimum sum of squared deviation. MLE, developed by R. A. Fisher (1920), on the other hand, calculates the parameter value which maximizes the function. MLE is sufficient, consistent, efficient, and parameter invariant, making it a must for inference with missing data. It isn't easy

to obtain parameter values when the density function is complex and non-linear, involving too many parameters. To find the parameter value that minimizes LSE or maximizes MLE, we need to employ numerical optimization techniques such as Newton, quasi-Newton, Gauss-newton, and Levenberg-Marquardt. The nonlinearity and complexity of the model function made it difficult to estimate the parameter's value since the model returned various values for the same set of parameters depending on the value of the guesses. With a larger number of parameters, it also got tedious to test every possible combination of guess values to find the best one.

Optimization issues in software reliability analysis (SRA) are increasingly being tackled with the use of soft computing methods, either alone or in tandem with other approaches.

2.8. CRITERIA FOR COMPARING RELIABILITY MODELS

Specific techniques are required to access the predictions made by developed models. Only calculating the difference or the summative deviation between the observed and experimented value is not enough to determine the model's capability; therefore, a set of values were used by many researchers to determine the model's reliability. Tabulated below in table 2.2 are some criteria based on which models were compared in the literature.

Table 2.2. Criteria used by researchers for the reliability model's comparison

Researcher (Year)	Comparison Criteria
Lyu and Nikora (1992)	BIAS, NOISE
Zhao and Xie (1992)	MEOP
Pillai and Nair (1997)	BIAS, VARIANCE, RMSE
Huang and Kuo (2002)	BIAS, AE, NOISE, VARIANCE, RMSE

Zhang et al. (2003)	PRR, SSE
Li et al. (2005)	TS
Zhao et al. (2006)	AE, R ² , SSE
Yang and Li (2007)	MSE
Chiu et al. (2008)	MSE, AE, R ²
Zheng and Xu (2008)	MSE
Huang and Huang (2008)	MSE, AE, VARIANCE, RMSE
Dohi and Ishii (2008)	MSE
Lin and Huang (2008)	BIAS, MSE, MEOP, VARIANCE, TS, RMSE
Caiuta et al. (2008)	AE, NOISE, VARIANCE
Hwang and Pham (2009)	MSE
Huang et al. (2009)	BIAS, MSE, VARIANCE, TS, RMSE
Kumar and Kapur (2009)	BIAS, MSE, VARIANCE, R ² , RMSE
Liu and Gao (2009)	BIAS, NOISE
Rafi and Akthar (2010)	BIAS, VARIANCE, R ² , SSE
Garg et al. (2010)	BIAS, MSE, VARIANCE, TS, RSE, MEOP, AE, NOISE, PRR
Sharma et al. (2010)	BIAS, MSE, VARIANCE, TS, RSE, MEOP, AE, NOISE, PRR
Miglani and Rana (2011)	BIAS, MSE
Aljahdali (2011)	VARIANCE
Anjum et al. (2013)	MSE, MEOP, AE, NOISE, TS

Khalid and Sharma (2015)	BIAS, MEOP, AE, PRR, RSE, TS
Lee et al. (2018)	MSE
P. Govindasamy and R. Dillibabu (2019)	MAD, RMSE, RAE, RRSE, MSE, MMRE, PRR, PP, R ²
Garg (2019)	BIAS, MSE, VARIANCE, TS, RSE, MEOP, AE, NOISE, PRR
Da Hye Lee et al. (2020)	MSE, PRR, PP, R ² , SAE, AIC, RMSE
Kamlesh Kumar Raghuvanshi et al. (2021)	MSE, RMSE, R ²

2.9. LITERATURE REVIEW

To grasp the overall concept and the many ways in which different studies have contributed to the subject, a comprehensive literature review was conducted. To fully understand the work done related to a different aspect of the research problem, the literature available in the undermentioned fields related to our research work was explored and considered. We can broadly classify it into the following main categories based on our literature study.

1. Big Data Applications (BDA)
2. Big Data and Reliability (BDR)
3. Software Reliability Hybrid Models (SRHM)
4. Soft Computing Techniques (SCT) and Reliability

The literature review format only discusses the representative work in a particular area based on existing trends, technology, and future research potential. The literature represents an indicative sample from the research pool due to the non-availability of all the research material about the research problem. Still, it supports the methodology development carried out in our work. The following sections

discuss the literature reviewed concerning major influential articles in various categories necessary for formulating research methodology.

2.9.1. Big Data Applications

In today's world, with so much advancement in technology, everyone is dependent on the internet for social networking, e-commerce, and various utility apps. They generate a vast amount of data regarding people's preferences, choices, likes, and dislikes. This bulk data, which carries information regarding customers, can significantly help decision-making for certain corporate houses related to a new product. A competitive edge and improved performance in a variety of fields are possible through the analysis of big data in the areas of management, security, sales, and customer happiness. (Sharma, Kumar & Kaswan 2019)

Big data is characterized by various dimensions ranging from 3V's to 7V's, consisting of volume, value, velocity, veracity, variability, etc., depending upon the usage in application and collection point. Nowadays, we feel the presence of big data in various fields like agriculture, education, healthcare, banking, business, transportation, etc. It is growing at a tremendous pace.

Gang Zeng (2015) resolves the problem of traffic management by proposing an architecture of intelligent transportation using traffic flow, the average speed of a road, the travel path taken by the vehicle, and controlling fake vehicles.

According to Protopopta and Shanoyanb (2016), the prospective expansion of agriculture in developing nations may be facilitated by the use of information and communication technologies (ICTs) and global positioning systems (GPS).

Applications of big data analytics were covered by Liu, Chong, Man, K. L., and Chan (2016). Companies that are making the most of Big data are setting the industry standard by analyzing the vast amounts of customer feedback they receive via social media, surveys, and reviews on their mobile apps, smartphones, and e-commerce sites to determine what is working and why.

Li, Zhang & Tian (2016) study shows that healthcare is another important sector that utilizes data generated from accurate diagnoses, successful treatment of patients, workflow management, education, and research.

Vaitsis, Hervatis, and Zary (2016) provide the user with a formalized understanding of education data and how to leverage big data using analytical and scientific methods. Information gathered from students' behaviors and interactions can help educators refine their practices and better define what constitutes high-quality education.

Using a cloud-based service-oriented architecture to predict the likelihood of congestion, blockages, traffic violence, and accidents, Kemp, Vargas-Solar, and colleagues (2016) detailed how big data analysis may be utilized to govern traffic lights and other factors.

Majumdar, Naraseeyappa & Ankalaki (2017) explained that the main occupation of our country is agriculture, and crop production depends on factors like climate, soil conditions, fertilizers used, and the type of seed used. By using CLARA, PAM, and BDSCAN various data mining techniques that can obtain the best yield of wheat.

Gill, Chana & Buyya (2017) developed a cloud-based automatic information system, that collects and manages agricultural data using K-NN for various agricultural-related queries. Smart farming using IoT is the need of the hour to increase crop production.

Kamilaris, Kartakoullis & Penafeta-Boldú (2017) give brief information regarding the source, characteristics, application, and techniques employed for big data in agriculture.

Radmehr and Bazmara (2017) explain that for better decision-making, a bank utilizes tools dependent on the volume of available data through a financial institution's social or economic aspect. Financial institutions stop fraudulent attempts and unusual activities by analysing real-time data. Robust fraud detection uses data analytics to

determine fraud patterns by comparing them to customers' usual patterns of cash deposits, withdrawals, and loans.

Self-analyzing many production factors involving massive amounts of data, Yao, Han, Yang, and Zhang (2018) created a distributed processing system based on cloud computing and described a cloud-based storage strategy.

Rao, Kishore, and Baglodi (2018) identify the three main domains in education, mainly students, teachers, and institute, which can be harnessed using suitable data mining techniques.

Rawat and Yadav (2021) discussed various analytics techniques, processes, and prominent challenges regarding big data. Their study emphasizes the need of updating current technology and tools with time.

Kushwaha, Kara & Dwivedi (2021) analyzed some business value frameworks in their study that can measure unbiased BDA values and crisis management. Their study identifies emerging management areas that are yet to get attention and are supported by big data.

2.9.2. Big Data Reliability

We studied the research papers concerning big data reliability from 2012 to 2019 in various reputed journals. We classified them according to the type of reliability they address-Data, Hardware, or Software. We further compared the software reliability models concerning the methods and techniques they use to analyze reliability.

After analyzing massive amounts of data from social networks, Han, Tian, Yoon, and Lee (2012) presented a big data model based on Map Reduce. The reliability of the recommendations concerning social behavior can be increased by using the model's parameters. The model can be modified to grow and expand by including extra parameters to reflect external factors if required.

Meeker and Hong (2013) recognized several uses for field reliability data such as warranty, degradation, lifespan, and recurrence field data, and they investigated

numerous possibilities for evaluating robust statistical approaches in forecasting the performance of the system in the field.

A public auditing technique that uses the Boneh-Lynn-Shacham (BLS) signature and the Multiple Huffman Table (MHT) has been suggested by Chang Liu et al. (2014). This scheme is for approved auditing. The method offers enhanced security by incorporating an additional authorization process to eliminate threats due to malicious auditors and show promising results for a vast number of small updates. The updates are not constrained by the size of the file block, thus providing better scalability and flexibility than current schemes.

Using a three-dimensional stochastic differential equation (3D-SDE) and assuming irregular and time-dependent fault reporting throughout the operation phase, Tamura, Miyaoka, and Yamada (2014) suggested a reliability model which estimates the reliability of open software systems running on the cloud.

Yoshinobu, Tamura & Yamada (2014) proposed a 3-D stochastic differential reliability model. They also explain the indirect effect on integrated reliability by considering the external interactions.

Bail (2014) synthesizes cultural sociology with big data by combining traditional qualitative research techniques with modern technological advancements. This allows for automated text extraction methods to be applied, which can then be used to follow the growth of the cultural environment. This synthesis will utilize automated text analysis techniques and political and computer scientists along with linguists to answer the unmeasurable questions related to theoretical progress by classifying cultures in schemas, frames, and different boundaries corresponding to different types of cultures.

According to Kwon, Lee, and Shin (2014), the desire to acquire big data analytics is significantly influenced by IT skills such as management of data quality and expertise in data utilization and better-quality management promotes data usage regardless of its source.

In order to settle disagreements across dissimilar large data sources, Li et al. (2014) suggested a methodology based on an optimal framework consisting of two variables: truths and source dependability. where the weights denote the degree of trustworthiness and truth is defined as the value accountable for the least feasible variation from multiple source inputs. They conducted various experiments to validate the developed two-step iterative process and demonstrate the benefits of using the CRH framework over existing conflict resolution techniques used in finding truth from heterogeneous data.

Tamura and Yamada (2015) proposed an SRM-based K-means clustering and hazard rate for big data environments with cloud computing. In order to assess the dependability of cloud computing throughout its operational period, they used cluster analysis of fault data to create an application called Application for Reliability Assessment (AIR).

Tamura and Yamada (2015) Proposed another model to evaluate the reliability of cloud computing. Using stochastic differential equations including two-dimensional Weiner processes, they calculated the predicted price of the program using a jump-diffusion model. They also describe an optimum maintenance issue in terms of a sample path, taking into account the amount of noise present.

Tamura and Yamada (2015) developed two reliability assessment (RA) methods for OSS based on determining component reliability and system-wide reliability using big data. A hazard rate model based on stochastic equations was used for a data set of cumulative faults and time intervals between failures in order to detect SRA. They verified their model's ability to predict cumulative failures in Hadoop and OpenStack database systems. Their research showed that the model they created could accurately predict the total number of bugs in Hadoop and OpenStack.

Tamura, Nobukawa & Yamada (2015) used NN and clustering (K-means) to develop a reliability model. NN was employed for cumulative faults estimating techniques that utilized the findings of cluster analysis based on fault datasets obtained from databases such as Hadoop and NoSQL as well as cloud technologies such as Eucalyptus and OpenStack.

The Socio-technical Risk Analysis "SoTeRia" framework was introduced by Pence et al. (2016). This method combines human error effects with traditional PRA approaches and managerial considerations and quantifies the organizational mechanism for performance shaping factors (PSF) in human resources. They applied the developed method to quantify the training quality by using it as a surrogate node for PSF in HRA and identified essential factors influencing the training quality.

Cai and Zhu (2015) developed a quality assessment framework for big data utilizing a feedback mechanism that lists common quality elements along with associated indicators.

Li, He & Ma (2016) developed a model for reliable network data mining using an improved version of the PageRank algorithm. The developed model used Matrix operations based on MapReduce to reduce time and space complexity, ensuring that the acquired dataset of webpages was relevant to the study subject. The developed model consists of three parts where the first module eliminates cheating webpages by utilizing Trust-Rank and PageRank. Module two refined the webpage search related to a topic using topic relevance combined with TC-Page-Rank. The third module determines the relevant web pages using improved HITS, considering the similarity index and webpage link amplification.

Hu, Liu, Diao, Meng & Sheng (2016) suggested a big data model for power distribution system operational reliability by applying parallel index rule mining to conduct an analysis of the influential aspects associated with the reliability index and by employing neural networks to evaluate the model. The proposed methodology can effectively assess and analyse the reliability indexes and operation state in real-time.

Spichkova et al. (2016) formulated a research-oriented framework and a model using features such as usability and reliability on the cloud computing platform. The model can be used by researchers involved in experimenting using large computations involving big data. The chimney platform has been tested using physics and structural biology research disciplines.

Tamura and Yamada (2016) proposed a maintenance problem and method to evaluate component reliability on the cloud platform. Both GA and NN were utilized to evaluate the parameters of the model.

Yan, Meng, Lu, and Li (2017) suggested a spatiotemporal framework for organizing heterogeneous large data. They simulate unseen aspects like predictive maintenance and energy savings to make manufacturing transparent. Semi-structured and unstructured big industrial data are both categorized using semantic web and target recognition, and then further sub-categorized using envelope analysis, time-frequency analysis, and signal decomposition to improve manufacturing transparency.

Wang, Wu, and Wang (2017) developed a reliability model to analyze the likelihood of data loss and look into the best parallel recovery strategies for replication and random shifting using 45 multi-way de-clustering data layouts. The simulation was done using MATLAB and SHARPE.

Nachiappan, Javadi, Calherios & Matawie (2017) explored the difficulties associated with cloud storage for large amounts of data and how replication and erasure may help to overcome them. They also discussed conceptual architecture and proposed a hybrid approach to handle reconstruction issues related to erasure coding utilizing less storage and reliability improvement.

Xiang, Du, Ma & Fan (2017) developed a text classifier to determine the reliability of online hotel reviews. To evaluate the classifier sensitivity and predict the purpose of travel they utilised datasets of reviews on Trip Advisor. They improved the classifier's performance using a unique solution to identify misclassification due to noise in the data set. The study reveals that issues related to data quality arise because of the innate nature and inconsistencies in the behavior of social media users, and collecting data from highly reputed websites might also yield unreliable results.

For free and open-source software, Tamura and Yamada (2017) created a deep learning-based RA model. They developed an application to access the OSS reliability using fault datasets. The proposed tool performs reliability estimation using a hazard rate model and deep learning.

Hong, Zhang & Meeker (2018) discussed and provided a review concerning the recent modeling and reliability analysis development regarding complex dimension structures. They emphasized the use of operating data and environment for large-scale data systems. They illustrated linking complex data as covariates to reliability response factors like recurrence time of events, time of failure, and degradation measurements.

Cao and Gao (2018) developed an SRM using fault tree analytics (FTA) for big data systems. FTA evaluates the big system reliability, acts as a reference for quality assurance, and performs a qualitative assessment of a module to determine the probability of failure.

The similarity model, as well as other approaches and tools to detect plagiarism and clones in software, were created by Yaremchuk and Kharchenk (2018). Researchers developed software agents to search global networks and storage regarding the actual reliability of pre-existed similar software.

Govindasamy and Dillibabu (2018) developed three hybrid reliability models combining pre-existed NHPP models. Models were validated using comparison criteria. MLE and GA were used for parameter evaluation. The expected number and cumulative failures were calculated and compared with experimental data to show the model's performance. To show the superiority of the developed models over other existing models a t-test was also conducted.

On the premise that OSS fault introduction rates are reducing with time, Wang, Zhang, and Yang (2022) proposed a Random Access Memory (RAM). The model proposed by them is a good fit and shows improved performance than other models. Their model can be utilized for fault prediction RA over OSS.

A hazard rate model for OSS with fault severity levels was presented by Yanagisawa, Tamura, Anand, and Yamada (2022). The researchers set out to create an adaptive baseline hazard function for two distinct types of fault data in the Bug Tracking System (BTS) by use of a Hazard rate model based on covariate vectors with CFSL. They evaluated their model's performance to that of the Hazard rate model CFSL using a number of cases, demonstrating that their suggested model is a superior match.

2.9.3. Software Reliability Hybrid Model

Pietrantuono, Russo & Trivedi (2003) proposed a prototype implementation of an online reliability monitoring approach by combining the dynamic analysis with an architectural-based reliability model.

Bustamante and Bustamante (2003) developed a hybrid NHPP model using the multinomial-exponential and exponential functions to determine the SRM. The repair process is a simple multinomial distribution.

Two hybrid neural fuzzy systems were created by Pai and Lin (2005) to help with reliability prediction issues. In the supervised learning phase, we quickly train using a scaled conjugate gradient learning approach. Different models, such as the Radial Bias Function (RBF), the feed-forward multi-layer perceptron (FFMLP), and the Autoregressive Integrated Moving Average (ARIMA), were used to evaluate the model's performance. Experiment results confirm the higher prediction capability of the hybrid model compared to others.

To mitigate the negative consequences of defects, Reis et al. (2005) presented a hybrid system that combines software and hardware fault detection techniques. They developed three hybrid techniques CRAFT, SWIFT (software technique), and RMT (hardware technique). Results indicate that adding software and hardware techniques enhances the system's reliability.

Pai (2005) developed an SRM using a Support Vector Machine (SVM) for lower forecasting errors. The SVM parameters were evaluated using GA. The non-linear property of the SVM model makes it a suitable candidate for capturing reliability than other models. Improper selection of SVM parameters may result in overfitting or underfitting but using GA ensures the selection of proper parameters in the SVMG model for reliability predictions.

Pai and Hong (2005) proposed a hybrid model for software reliability measurement by combining SVM and SA methods. SA is used for parameter estimation of the SVM model.

Kiran and Ravi (2007) proposed a hybrid model that uses various linear and non-linear ensembles and uses multiple linear regression and intelligent techniques like neuro-fuzzy inference, BPNN, and Tree Net to predict SR. The results demonstrated that the non-linear combination of methods was superior to the linear one.

Yu-dong, Ning, Ying & Xiao-fang (2010) proposed a hybrid non-linear model using Backpropagation NN to improve prediction accuracy. Results showed that retained traditional models experience while incorporating the non-linear mapping of BPNN.

Cao and Zhu (2010) developed a hybrid model combining ARIMA and Fractal models, which incorporates the strength of both linear as well as non-linear modeling.

Jin (2010) developed a software reliability model using SVM for reliability prediction and a combination of GA and SA methods for parameter estimation of SVM.

Blackburn and Huddell (2012) developed a hybrid Bayesian network to predict software reliability.

Lo (2012) created a hybrid model that combines ARIMA and SVM to assess trustworthiness by taking into account both linear and non-linear trends. The developed method incorporates the unique strength of both ARIMA and SVM to improve predictions.

Vamsidhar, Raju & Kumar (2012) combined various SRGM using an Ada-boosting-based combinational model (ACM) to obtain a linear combinatorial model. Results depicted that the fitness and prediction of ACM are better than the individual SRGM.

To ascertain the dependability of composite web services, Prakash, Maruthurkarasi, Ganesh, and Maheswari (2013) created a hybrid model by fusing path and state based models. The model assumes that the web service's reliability rate is proportional to the server workload. The services were dispatched as being in an ideal or active state depending upon the server workload using a stochastic model. An architecture was implemented using bounded set techniques to infer the errors by calculating error rate, accessibility, and active state availability.

Using time series data, Chang, Lin, and Pai (2014) built a neuro-fuzzy hybrid model to test the viability of dependability prediction. Generalized Neural Networks (GRNN) and Bix-Jenkins ARIMA were used for prediction accuracy.

Kalaivani and Somsundaram (2014) proposed a framework for accessing reliability, quality risk, and fault detection using GA and PSO methods for type II censored data. Reliability estimation is done using hazard rate corresponding to PSO-GA algorithms.

Pushphavathi, Suma & Ramaswamy (2014) developed a defect prediction hybrid model using fuzzy c means (FCM) and random forest (RF). RF method is used to screen variables and gain actual ranks. FCM then builds a fault prediction model using the modified data set. Good classification accuracy is shown by testing the suggested model on the empirical data set.

Mohanty, Ravi & Patra (2015) proposed a new architecture utilizing genetic programming (GP). For the purpose of dependability estimation, they devised a number of different hybrid models and group methods of data handling (GMDH).

Pati and Shukla (2015) developed an ARIMA + NN-based software reliability hybrid model for failure interval series. A good deal of time, cost, and statistical testing is required by models using only ARIMA. In contrast, the hybrid model is completely data-oriented and provides an excellent alternative to ARIMA. The hybrid ARIMA + NN model shows better performance than only ARIMA models in reliability prediction.

Cao, Yue, Xiong & Zhao (2015) developed a hybrid model combining neural networks with back-propagation (BPNN) and fractal models to determine the next failure time.

In order to measure the dependability of cloud computing, Xuejie, Zhijian, and Feng (2015) integrate the MTTF/MTTR model with the CTMC model.

Roy, Mahapatra & Dey (2015) proposed a feed-forward multi-layer NN-based model for predicting reliability. The neural network was trained using NN, and GA was used to optimize the parameter's value.

Jin and Jin (2015) proposed an SRGM with an S-shaped testing effort function (TEF) to determine the reliability and then utilized a modified version of PSO to optimize the value of the parameter thus obtained.

Jabeen et al. (2017) develop a hybrid model by combining Jelinski-Moranda (JM) and Gompertz-Makeham (GM) models. JM model is best suited for stochastic data sequences requiring large data sets for vibrating and changing data, whereas GM handles forecasting well when the data set is small. The developed hybrid model utilizes the property of both models in giving better forecasting results.

Manjula and Florence (2017) developed a (GA and NN)-based model for feature optimization and classification for the early estimation of faults.

Wang, Jin, Yang & Han (2018) developed a hybrid model using BPNN and GA. BPNN model takes three traditional reliability models as input and GA to optimize the weights.

Sahu and Srivastava (2018) developed a hybrid model for SRA using fuzzy logic (FL) and NN. Compared with other models like a fuzzy neural network and neural-fuzzy, the model showed better performance than others.

By fusing the Flower Pollination and GA(real coded) algorithms, Alneamy and Dabdoob (2019) created a novel hybrid software reliability model. Models are validated using PSO, Artificial Bee Colony (ABC), Dichotomous ABC, Classic, and Modified GA. Results show that the developed hybrid model outperforms other models.

Sun, Wu, Wu, and Yang (2019) developed a hybrid model using SVM and ARIMA to determine the prediction of failure time series.

Shakya and Smys (2020) proposed a hybrid model of ant colony optimization (ACO) and differential evaluation algorithm (DEA) to measure software faults. The model is tested against artificial neural networks and PSO methods. The hybrid model achieves an impressive 96.2% precision.

Sudharson and Prabha (2020) developed a hybrid model using Pareto distribution (PD) with ACO and neural networks for enhancing classification.

Meng Li, Sadoughia, Hub & Hua (2020) proposed a hybrid Gaussian model (HGP_SRA) consisting of multivariate and univariate Gaussian process models (MGP+UGP).

Gandhi et al. (2021) developed a Neuro-Fuzzy hybrid model for reliability prediction. The developed model was compared with others for performance evaluation using MRE and MARE criteria. Results confirm the better accuracy of the hybrid model as compared to NN and Fuzzy Inference.

Milovancevic et al. (2021) determine the best reliability model for a particular application using a neuro-fuzzy inference system based on root mean square evaluation. Results showed that NHPP models combined with Littlewood- Verrall (LV) model give optimum performance concerning software quality.

2.9.4. Parameter Estimation Techniques

Sinha Kalra & Kumar (2000) proposed a neuron model which outperforms the Kalman filter (KF) and feed-forward multi-layer neural network (FMNN) using lambda -gamma learning by reducing the neuron interconnections and hence the computing time. The suggested model possessed the characteristics of both the basic neuron model and higher neuron model and adapted well not only to standard or higher neurons but also to their combination. Researchers also proposed to use the developed model with II order optimization with back-propagation of errors for further reduction in computations.

Sharma, Pant & Abraham (2011) proposed a parameter optimization method using a modified ABC algorithm. The revised version is called Dichotomous ABC (DABC) to create a new trial, and it moves bidirectionally. The parameters of models like exponential, power, and S-shaped were evaluated using a developed model and showed better performance than statistical methods.

Diwaker and Goyat (2014) optimized the Goel-Okamoto reliability model parameters using simulated annealing. The performance of SA is superior compared to PSO, ACO, GA, and NN optimizers.

Parameter optimization utilizing a variant of genetic swarm optimization (MGSO) and the logistic-exponential testing effort function was suggested by Rao and Anuradha (2016).

AL-Saati and Abd-AIKareem (2016) estimated the parameters of SRGM using the cuckoo search (CS) optimization method. Comparisons were made between the performance of CS and that of other optimizers, such as PSO, ACO, and extended ACO (EACO). The performance of CS was better than PSO and ACO, but in some cases, EACO gave better results.

Choudhary, Baghel & Sangwan (2016) proposed an effective parameter optimization approach using the gravitational search (GSA) algorithm. They carried out the experiments with nine datasets, and the results showed a better performance of GSA than GA and CS.

Alneamy and Dabdoob (2017) proposed a hybrid model HGWO for parameter estimation using grey wolf optimizer (GWO) and GA of a reliability growth model.

Kumar, Tripathi, Saraswat & Gupta (2017) developed a hybrid SRM by combining GA and PSO for parameter optimization.

Lohmor and Sagar (2017) developed a parameter estimation technique using hybrid Dolphin Echolocation Optimization (DEO) combined with ANN.

Latha and Premchand (2018) proposed a hybrid scheme to optimize the parameters of SRGM by combining ACO with the Traveling Salesman Problem (TSP) algorithm. Model validation is done using the criteria involving the Meantime to Failure (MTTF) measure along with Meantime Between Failure (MTBF).

Sharma (2018) used a Shuffled Frog leaping algorithm (SFLA), for parameter estimation of SRGMs. The researcher also developed a modified version of SFLA called the opposition-based SFLA. The developed model was verified using six benchmark functions compared with other algorithms to confirm better efficiency.

Sangeeta, Sharma & Bala (2019) developed a hybrid model for parameter optimization using the ABC model with a differential evolution (DE) model based on the ecological space concept.

Li, Yu, Wang & Wei (2019) proposed a hybrid parameter optimization algorithm using hybrid ABC and PSO models.

Banga, Bansal & Singh (2019) proposed a hybrid parameter optimization algorithm using PSO and modified GA.

For the purpose of parameter optimization, Gao and Zhao (2019) suggested an enhanced version of grey wolf optimization that they referred to as variable weight GWO. Comparing the created model to ant lion optimization (ALO), particle swarm optimization (PSO), and the bat algorithm (BA), the results demonstrated the superior performance of the latter.

Zhen, Liu, Dongsheng & Wei (2020) developed a hybrid optimization algorithm using PSO and wolf pack algorithm (WPA) to optimize the parameters of GO SRM accurately.

Sangeeta and Sitender (2020) developed hybrid meta-heuristic nature-inspired algorithms for parameter optimization using both interval domain and time domain data sets.

Yang, Li, Wang, Miao & Wang (2021) developed a hybrid model for parameter estimation using PSO and sparrow search algorithm (SSA).

2.10. FINDINGS OF CRITICAL REVIEW

- Hundreds of existing reliability models add more to the list every year, but not one can be used universally. To account for the parameters representing new interactions, model complexity continues rising with the arrival of big data applications and changes in the architecture and technologies employed, making their implementation more challenging.

- We live in an age when big data applications are ubiquitous. There are various models for predicting dependability, but whether or not they can be used with large-scale failure data is an open subject. We need a model which can address all the issues arising due to big data.
- Different models have different properties; it's better to combine two or more models to get the desired result to address all the issues. NHPP models are simple and effective and are primarily used for SRA. We may also add the mean value functions of many NHPP models and evaluate their parameters to get a single hybrid NHPP model.
- Parameter evaluation is a critical task in complex and non-linear hybrid models. Moreover, the fault data required to evaluate and test the models are scarcely available, making it difficult to validate the parameter's value in the context of real applications. This deficiency of experimental data acts as a roadblock to successfully implementing reliability models.
- Parameters can be evaluated statically using LSE or MSE techniques. In the case of non-linear and complex distribution functions, various soft computing techniques were utilised to get results.
- Parameter evaluation using MLE or LSE techniques requires various optimizers like Newton-Raphson, Nelder-Mead, Trust-region, Trust-region-dogleg, and Lavenberg-Marquardt. These methods were used to solve the linear equations and determine the parameter's value. Each method has its merits and demerits, and not one can be used universally for all types of applications. Newton-Rapson can be used for non-linear functions, but if the initial guess is not correct, it might not converge, whereas the trust-region algorithm is more effective on sparse problems, and trust-region-dodge is specifically designed to handle non-linear equations. Nelder-Mead is efficient, robust, and always

converges but can work only with a specific number of parameters and is slow in the neighborhoods of minima as compared to other methods.

- There is no guideline that we can accommodate to select a particular model. Researchers have been using various comparison criteria to choose a specific model, but no one can ensure the suitability of a specific model for reliability prediction giving rise to the need to develop new models for newer and technically advanced applications.

CONCLUSION

There is an abundance of models to choose from for the reliability prediction of a specific application but the model's classification according to their applicability is not present. Although literature discusses the MLE and LSE statistical techniques, how to use them in reliability modelling to evaluate parameters effectively is missing. Researchers did not test various combinations of model selection criteria set for any specific model to obtain the pros and cons of a grouping of multiple selection methods. Everyone includes these methods without any explanation. There was no suitable material in the literature regarding the acceptable range for guess values for obtaining parameters. There is no explanation in the literature regarding which in-built functions of particular applications they used and in what way we can use them for parameter evaluation. The discussion regarding specific queries like: Is it better to develop programs for parameter evaluation or to use in-built functions of a particular application, which platform is best suited for, which type of reliability modelling is entirely missing, and which optimization algorithm to choose?

CHAPTER 3

MODEL DEVELOPMENT, SELECTION FUNCTION, COMPARISON CRITERIA, AND RANKING METHODOLOGY

Traditional reliability models attributed program failures to internal causes, such as faulty specifications or coding, rather than external causes. Technological advancement and the widespread use of big data have led to the creation of several types of specialist software. It was observed that any software might contain many errors and each fault require a different mechanism and method to be detected and corrected. In this chapter, we proposed some hybrid models that, apart from pure software errors, also consider induced errors in software due to environmental factors into consideration. Additionally, a ranking approach to rank models is presented, along with a selection function and comparison criteria to choose the best model among the candidate models.

3.1. INDUCED ERRORS IN SOFTWARE

A system interacts with its environment to get the intended work done. As big data is increasingly used, specialized hardware is needed to store and retrieve the massive amounts of information involved. Big data needs particular treatment because of its volume, velocity, and veracity, which may generate the following sorts of errors

3.1.1. Volume

Due to the high volume, a large enough memory space is required with an advanced access mechanism. Any fault due to the current load while fetching data results in software failure due to hardware-induced error.

3.1.2. Velocity

Data is streaming in the system in continuous mode and in real-time, requiring fast collection and processing. High velocity requires new cache architecture for the primary access mechanism and advanced techniques for processing and analysing massive data rapidly. Malfunctioning in any can also affect the reliability of software.

3.1.3. Variety

Since the database of Big Data is not usually structured, the faults resulting from the employment of techniques for keeping unstructured data in the memory come under this category. When a model accounts for every possible failure scenario, it is considered to be more flexible. To make advantage of faulty data, the hybrid system design must account for the flaws arising from these features of big data. A developed model must catch all pure errors as well hardware and user-induced errors.

3.2. HYBRID MODEL DEVELOPMENT

To develop a model, we consider that apart from pure software faults, mainly resulting from a coding error; software failure may occur due to induced errors in software. Errors can be caused either by malfunctioning hardware or by a user's error while handling data. A direct modification in an NHPP model that can successfully handle software errors is to combine it with other NHPP models to tackle induced errors resulting from hardware and user. The hybrid model is then developed by adding three NHPP models to capture three types of errors: pure software, hardware, and user.

3.2.1. Pure Software Errors

Software errors are generated because of a bug in code or incorrect design specifications and remain constant throughout the life of software until removed. Pure software errors can be successfully modelled using well-known existing NHPP models.

3.2.2. Hardware-Generated Errors

Hardware reliability depends on time. Suppose there is an error due to hardware malfunctioning. In that case, the hardware will degrade with time, and the error will amplify and be represented by a growing curve of cumulated errors concerning the time until the problem is solved. When there is no hardware malfunctioning, then the error is eliminated, giving us a curve similar to the Weibull distribution.

We used Duane and modified Duane NHPP reliability models to capture fault distribution related to hardware.

3.2.3. Human Errors

Human errors involve faults due to the mishandling of software. Initially, when personnel are less skilled in handling software, the errors generated will be increased. As time passes, their expertise increases, and they become familiar with the software; the errors reduce. The user-induced errors take on the form of an exponential distribution or can be described by the Yamada S-shaped curve.

3.3. MATHEMATICAL FORMULATION OF HYBRID MODELS

To develop a mathematical formulation and determine the MVF of a hybrid model, we combined the MVF of three NHPP models:

$$\mathbf{m}(t) = \mathbf{m1}(t) + \mathbf{m2}(t) + \mathbf{m3}(t)$$

Where,

$\mathbf{m}(t)$: MVF of hybrid model.

$\mathbf{m1}(t)$: MVF of models for capturing pure software errors.

$\mathbf{m2}(t)$: MVF of models for capturing hardware-induced errors in software.

$m_3(t)$: MVF of models for induced errors in software due to human negligence.

We used $m_1(t)$ values of Generalized Goel(GG), Gompertz(GMPZ), Goel-Okumoto(GO), Yamada Delayed(YDM), and Inflection S-shaped(YIM), and logistic growth curve (LGC) models in combination with $m_2(t)$ values of the Duane model(DM) and modified Duane(MD) model and $m_3(t)$ values of the Exponential model(ExpM), Yamada's Delayed and exponential(YEM) models to obtain 33 various hybrid models and name them Sharma Kumar Kaswan-1(SKK-1) up to Sharma Kumar Kaswan-33 (SKK-33) as shown in Table 3.1.

Table 3.1. Proposed hybrid software reliability models

Model Name	Models used for Pure Software error	Models used for hardware-induced errors	Models used for user-induced errors
SKK-1	Goel-Okumoto	Modified Duane	Exponential
SKK-2	Yamada Delayed S-Shaped	Modified Duane	Exponential
SKK-3	Yamada Inflection S-Shaped	Modified Duane	Exponential
SKK-4	Generalized Goel	Duane	Exponential
SKK-5	Generalized Goel	Modified Duane	Exponential
SKK-6	Gompertz	Duane	Exponential
SKK-7	Gompertz	Modified Duane	Exponential
SKK-8	Logistic Growth Curve	Duane	Exponential
SKK-9	Logistic Growth Curve	Modified Duane	Exponential
SKK-10	Yamada Exponential	Duane	Exponential
SKK-11	Yamada Exponential	Modified Duane	Exponential
SKK-12	Generalized Goel	Duane	Yamada Delayed S-Shaped
SKK-13	Gompertz	Duane	Yamada Delayed S-Shaped
SKK-14	Goel-Okumoto	Duane	Yamada Delayed S-Shaped
SKK-15	Yamada Inflection S-Shaped	Duane	Yamada Delayed S-Shaped
SKK-16	Logistic Growth Curve	Duane	Yamada Delayed S-Shaped
SKK-17	Generalized Goel	Modified Duane	Yamada Delayed S-Shaped

SKK-18	Gompertz	Modified Duane	Yamada Delayed S-Shaped
SKK-19	Goel-Okumoto	Modified Duane	Yamada Delayed S-Shaped
SKK-20	Yamada Inflection S-Shaped	Modified Duane	Yamada Delayed S-Shaped
SKK-21	Logistic Growth Curve	Modified Duane	Yamada Delayed S-Shaped
SKK-22	Generalized Goel	Duane	Yamada Exponential
SKK-23	Gompertz	Duane	Yamada Exponential
SKK-24	Goel-Okumoto	Duane	Yamada Exponential
SKK-25	Yamada Inflection S-Shaped	Duane	Yamada Exponential
SKK-26	Logistic Growth Curve	Duane	Yamada Exponential
SKK-27	Yamada Delayed S-Shaped	Duane	Yamada Exponential
SKK-28	Generalized Goel	Modified Duane	Yamada Exponential
SKK-29	Gompertz	Modified Duane	Yamada Exponential
SKK-30	Goel-Okumoto	Modified Duane	Yamada Exponential
SKK-31	Yamada Inflection S-Shaped	Modified Duane	Yamada Exponential
SKK-32	Logistic Growth Curve	Modified Duane	Yamada Exponential
SKK-33	Yamada Delayed S-Shaped	Modified Duane	Yamada Exponential

3.3.1. Sharma, Kumar & Kaswan-1 Hybrid Model

We derived the MVF of the SKK-1 hybrid model by combining:

- The MVF $m_1(t)$ of Goel-Okumoto Model for pure software errors.
- The MVF $m_2(t)$ of modified Duane for hardware-induced errors.
- The MVF $m_3(t)$ of exponential model for user-induced errors.

$$m_{h1}(t) = m_1(t) + m_2(t) + m_3(t) \quad (3.1)$$

$$\lambda_{h1}(t) = \lambda_1(t) + \lambda_2(t) + \lambda_3(t) \quad (3.2)$$

Where,

$\lambda_1(t)$, $\lambda_2(t)$ and $\lambda_3(t)$ represents the InF of Goel-Okumoto, modified Duane, and exponential models. The MVF $m_{h1}(t)$ and InF $\lambda_{h1}(t)$ of SKK-1 hybrid model can be derived as

Using Goel-Okumoto model: -

$$m_1(t) = x(1 - e^{-yt}) \quad (3.3)$$

$$\lambda_1(t) = xye^{-yt} \quad (3.4)$$

Using the modified Duane model: -

$$m_2(t) = a(1 - (b/(b+t))^c) \quad (3.5)$$

$$\lambda_2(t) = acb^c (b+t)^{(1-c)} \quad (3.6)$$

Using the Exponential model: -

$$m_3(t) = \alpha t \quad (3.7)$$

$$\lambda_3(t) = \alpha \quad (3.8)$$

Substituting equations (3), (5), and (7) in equation (1), we get

$$m_{h1}(t) = x(1 - e^{-yt}) + a(1 - (b/(b+t))^c) + \alpha t \quad (3.9)$$

Combining equations (4), (6), and (8) the SKK-1 Intensity function $\lambda_{h1}(t)$ is derived as

$$\lambda_{h1}(t) = xye^{-yt} + acb^c (b+t)^{(1-c)} + \alpha \quad (3.10)$$

3.3.2. Sharma, Kumar & Kaswan-2 Hybrid Model

SKK-2 model was developed by combining MVF of YDM, MD, and ExpM.

we derived the MVF $m_{h2}(t)$ of SKK-2 model as:

$$m_{h2}(t) = x(1 - (1 + yt)e^{-yt}) + a(1 - (b/(b+t))^c) + \alpha t \quad (3.11)$$

Similarly combining intensity functions, we derived the InF $\lambda_{h2}(t)$ of the hybrid model as

$$\lambda_{h2}(t) = xy^2te^{-yt} + acb^c (b+t)^{(1-c)} + \alpha \quad (3.12)$$

3.3.3. Sharma, Kumar & Kaswan-3 Hybrid Model

The MVF and InF of SKK-3 can be derived by combining Yamada Inflection S-Shaped Model with Modified Duane and Exponential model as:

$$m_{h3}(t) = \frac{x(1-e^{-yt})}{1+ze^{-yt}} + a(1 - (b/(b+t))^c) + \alpha t \quad (3.13)$$

$$\lambda_{h3}(t) = \frac{xye^{-yt}(1+z)}{(1+ze^{-yt})^2} + acb^c (b+t)^{(1-c)} + \alpha \quad (3.14)$$

3.3.4. Sharma, Kumar & Kaswan-4 Hybrid Model

The MVF and Inf of SKK-4 were derived by combining GG Model, Duane, and Exponential model as

$$m_{h4}(t) = x(1 - e^{-yt^z}) + at^b + \alpha t \quad (3.15)$$

$$\lambda_{h4}(t) = xyz t^{z-1} e^{-yt^z} + abt^{(b-1)} + \alpha \quad (3.16)$$

3.3.5. Sharma, Kumar & Kaswan-5 Hybrid Model

The MVF and Inf of SKK-5 can be derived by combining GG Model with the modified Duane and Exponential model as:

$$m_{h5}(t) = x(1 - e^{-yt^z}) + a(1 - (b/(b+t))^c) + \alpha t \quad (3.17)$$

$$\lambda_{h5}(t) = xyz t^{z-1} e^{-yt^z} + acb^c (b+t)^{(1-c)} + \alpha \quad (3.18)$$

3.3.6. Sharma, Kumar & Kaswan-6 Hybrid Model

The MVF and Inf of SKK-6, derived after combining the GMPZ model, Duane, and Exponential model as:

$$m_{h6}(t) = xze^{-yt} + at^b + \alpha t \quad (3.19)$$

$$\lambda_{h6}(t) = xy \ln(z) z e^{-yt} e^{-yt} + abt^{(b-1)} + \alpha \quad (3.20)$$

3.3.7. Sharma, Kumar & Kaswan-7 Hybrid Model

Combining Gompertz Growth Curve (GMPZ) Model, Modified Duane, and Exponential model, we got the MVF and InF of the SKK-7 model as:

$$m_{h7}(t) = xze^{-yt} + a(1 - (b/(b+t))^c) + \alpha t \quad (3.21)$$

$$\lambda_{h7}(t) = xy \ln(z) z e^{-yt} e^{-yt} + acb^c (b+t)^{(1-c)} + \alpha \quad (3.22)$$

3.3.8. Sharma, Kumar & Kaswan-8 Hybrid Model

The MVF and Inf of SKK-8, derived after combining the LGC model, Duane, and Exponential model as:

$$m_{h8}(t) = \frac{x}{(1+ze^{-yt})} + at^b + \alpha t \quad (3.23)$$

$$\lambda_{h8}(t) = \frac{xyze^{-yt}}{(1+ze^{-yt})^2} + abt^{(b-1)} + \alpha \quad (3.24)$$

3.3.9. Sharma, Kumar & Kaswan-9 Hybrid Model

Combining the LGC-modified Duane and Exponential model, we got MVF and InF of the SKK-9 model as:

$$m_{h9}(t) = \frac{x}{(1+ze^{-yt})} + a(1 - (b/(b+t))^c) + \alpha t \quad (3.25)$$

$$\lambda_{h9}(t) = \frac{xyze^{-yt}}{(1+ze^{-yt})^2} + acb^c (b+t)^{(1-c)} + \alpha \quad (3.26)$$

3.3.10. Sharma, Kumar & Kaswan-10 Hybrid Model

The MVF and Inf of SKK-10, derived after combining the Yamada exponential curve model, Duane, and Exponential model as:

$$m_{h10}(t) = x\{1 - e^{-zw(1-e^{yt})}\} + at^b + \alpha t \quad (3.27)$$

$$\lambda_{h10}(t) = xwzye^{-zw(1-e^{-yt})-yt} + abt^{(b-1)} + \alpha \quad (3.28)$$

3.3.11. Sharma, Kumar & Kaswan-11 Hybrid Model

Combining YEM with the Modified Duane and Exponential Model, we derived the MVF and InF of the SKK-11 hybrid model as:

$$m_{h11}(t) = x\{1 - e^{-zw(1-e^{yt})}\} + a(1 - (b/(b+t))^c) + \alpha t \quad (3.29)$$

$$\lambda_{h11}(t) = xwzye^{-zw(1-e^{-yt})-yt} + acb^c (b+t)^{(1-c)} + \alpha \quad (3.30)$$

3.3.12. Sharma, Kumar & Kaswan-12 Hybrid Model

The MVF and Inf of SKK- 12, derived after combining the GG model, Duane model, and YDM as:

$$m_{h12}(t) = x(1 - e^{-yt^z}) + at^b + \alpha\{1 - (1 + \beta t)e^{-\beta t}\} \quad (3.31)$$

$$\lambda_{h12}(t) = xyz t^{z-1} e^{-yt^z} + abt^{(b-1)} + \alpha\beta^2 t e^{-\beta t} \quad (3.32)$$

3.3.13. Sharma, Kumar & Kaswan-13 Hybrid Model

Combining GMPZ, DM, and YDM, we got the MVF and InF of the SKK-13 model as:

$$m_{h13}(t) = xze^{-yt} + at^b + \alpha(1 - (1 + \beta t)e^{-\beta t}) \quad (3.33)$$

$$\lambda_{h13}(t) = xy \ln(z) z e^{-yt} e^{-yt} + abt^{(b-1)} + \alpha\beta^2 t e^{-\beta t} \quad (3.34)$$

3.3.14. Sharma, Kumar & Kaswan-14 Hybrid Model

The MVF and Inf of SKK-14, derived after combining the GO model, DM, and YDM as:

$$m_{h14}(t) = x(1 - e^{-yt}) + at^b + \alpha(1 - (1 + \beta t)e^{-\beta t}) \quad (3.35)$$

$$\lambda_{h14}(t) = xye^{-yt} + abt^{(b-1)} + \alpha\beta^2 te^{-\beta t} \quad (3.36)$$

3.3.15. Sharma, Kumar & Kaswan-15 hybrid model

Combining the YIM, DM, and YDM, we got the MVF and InF of the SKK-15 model as:

$$m_{h15}(t) = \frac{x(1-e^{-yt})}{1+ze^{-yt}} + at^b + \alpha(1 - (1 + \beta t)e^{-\beta t}) \quad (3.37)$$

$$\lambda_{h15}(t) = \frac{xye^{-yt}(1+z)}{(1+ze^{-yt})^2} + abt^{(b-1)} + \alpha\beta^2 te^{-\beta t} + \alpha\beta^2 te^{-\beta t} \quad (3.38)$$

3.3.16. Sharma, Kumar & Kaswan-16 Hybrid Model

The MVF and Inf of SKK- 16, derived after combining the LGC model, Duane, and YDM as:

$$m_{h16}(t) = \frac{x}{(1+ze^{-yt})} + at^b + \alpha(1 - (1 + \beta t)e^{-\beta t}) \quad (3.39)$$

$$\lambda_{h16}(t) = \frac{xyze^{-yt}}{(1+ze^{-yt})^2} + abt^{(b-1)} + \alpha\beta^2 te^{-\beta t} \quad (3.40)$$

3.3.17. Sharma, Kumar & Kaswan-17 Hybrid Model

Combining the GG, MDM, and YDM, we got the MVF and InF of the SKK-17 model as:

$$m_{h17}(t) = x(1 - e^{-yt^z}) + a(1 - (b/(b + t))^c) + \alpha(1 - (1 + \beta t)e^{-\beta t}) \quad (3.41)$$

$$\lambda_{h17}(t) = xyz t^{z-1} e^{-yt^z} + acb^c (b + t)^{(1-c)} + \alpha\beta^2 te^{-\beta t} \quad (3.42)$$

3.3.18. Sharma, Kumar & Kaswan-18 Hybrid Model

The MVF and Inf of SKK-18, derived after combining the GMPZ model, Duane, and YDM as:

$$m_{h18}(t) = xze^{-yt} + a(1 - (b/(b+t))^c) + \alpha(1 - (1 + \beta t)e^{-\beta t}) \quad (3.43)$$

$$\lambda_{h18}(t) = xy \ln(z) z e^{-yt} e^{-yt} + acb^c (b+t)^{(1-c)} + \alpha\beta^2 t e^{-\beta t} \quad (3.44)$$

3.3.19. Sharma, Kumar & Kaswan-19 Hybrid Model

Combining the GO model, MDM, and YDM, we got the MVF and InF of the SKK-19 model as:

$$m_{h19}(t) = x(1 - e^{-yt}) + a(1 - (b/(b+t))^c) + \alpha(1 - (1 + \beta t)e^{-\beta t}) \quad (3.45)$$

$$\lambda_{h19}(t) = xye^{-yt} + acb^c (b+t)^{(1-c)} + \alpha\beta^2 t e^{-\beta t} \quad (3.46)$$

3.3.20. Sharma, Kumar & Kaswan-20 Hybrid Model

The MVF and InF of SKK-20, derived after combining YIM, MDM, and YDM as

$$m_{h20}(t) = \frac{x(1-e^{-yt})}{1+ze^{-yt}} + a(1 - (b/(b+t))^c) + \alpha(1 - (1 + \beta t)e^{-\beta t}) \quad (3.47)$$

$$\lambda_{h20}(t) = \frac{xye^{-yt}(1+z)}{(1+ze^{-yt})^2} + acb^c (b+t)^{(1-c)} + \alpha\beta^2 t e^{-\beta t} \quad (3.48)$$

3.3.21. Sharma, Kumar & Kaswan-21 Hybrid Model

Combining LGC, Modified Duane, and Yamada Exponential model, we got MVF and InF of the SKK-21 model as:

$$m_{h21}(t) = \frac{x}{(1+ze^{-yt})} + a(1 - (b/(b+t))^c) + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.49)$$

$$\lambda_{h21}(t) = \frac{xyze^{-yt}}{(1+ze^{-yt})^2} + acb^c (b+t)^{(1-c)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.50)$$

3.3.22. Sharma, Kumar & Kaswan-22 Hybrid Model

The MVF and InF of SKK-22, derived after combining the GG model, Duane, and Yamada Exponential model as:

$$m_{h22}(t) = x(1 - e^{-yt^z}) + at^b + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.51)$$

$$\lambda_{h22}(t) = xyz t^{z-1} e^{-yt^z} + abt^{(b-1)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.52)$$

3.3.23. Sharma, Kumar & Kaswan-23 Hybrid Model

Combining the GMPZ, DM, and YEM, we got MVF and InF of the SKK-23 model as

$$m_{h23}(t) = xze^{-yt} + at^b + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.53)$$

$$\lambda_{h23}(t) = xy \ln(z) z e^{-yt} e^{-yt} + abt^{(b-1)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.54)$$

3.3.24. Sharma, Kumar & Kaswan-24 Hybrid Model

The MVF and InF of the SKK-24, derived after combining the GO model, DM, and YEM

$$m_{h24}(t) = x(1 - e^{-yt}) + at^b + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.55)$$

$$\lambda_{h24}(t) = xye^{-yt} + abt^{(b-1)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.56)$$

3.3.25. Sharma, Kumar & Kaswan-25 Hybrid Model

Combining the YIM, the DM, and the YE model, The MVF and InF are given as

$$m_{h25}(t) = \frac{x(1-e^{-yt})}{1+ze^{-yt}} + at^b + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.57)$$

$$\lambda_{h25}(t) = \frac{xye^{-yt}(1+z)}{(1+ze^{-yt})^2} + abt^{(b-1)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.58)$$

3.3.26. Sharma, Kumar & Kaswan-26 Hybrid Model

Combining the LGC model with Duane and YE to derive MVF and InF of the SKK-26 hybrid model as:

$$m_{h26}(t) = \frac{x}{(1+ze^{-yt})} + at^b + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.59)$$

$$\lambda_{h26}(t) = \frac{xyze^{-yt}}{(1+ze^{-yt})^2} + abt^{(b-1)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.60)$$

3.3.27. Sharma, Kumar & Kaswan-27 Hybrid Model

Combining the YDM, MDM, and YEM, we got the MVF and InF of the SKK-27 model as:

$$m_{h27}(t) = x(1 - (1 + yt)e^{-yt}) + at^b + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.61)$$

$$\lambda_{h27}(t) = xy^2te^{-yt} + abt^{(b-1)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.62)$$

3.3.28. Sharma, Kumar & Kaswan-28 Hybrid Model

Combining the GG model, MDM, and YEM, we derived the MVF and InF of the SKK-28 hybrid model as:

$$m_{h28}(t) = x(1 - e^{-yt^z}) + a(1 - (b/(b+t))^c) + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.63)$$

$$\lambda_{h28}(t) = xyz t^{z-1} e^{-yt^z} + acb^c (b+t)^{(1-c)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.64)$$

3.3.29. Sharma, Kumar & Kaswan-29 Hybrid Model

Combining GMPZ, MDM, and YEM, we got MVF and InF of the SKK-29 model as

$$m_{h29}(t) = xze^{-yt} + a(1 - (b/(b+t))^c) + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.65)$$

$$\lambda_{h29}(t) = xy \ln(z) z e^{-yt} e^{-yt} + acb^c (b+t)^{(1-c)} + rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.66)$$

3.3.30. Sharma, Kumar & Kaswan-30 Hybrid Model

Combining the GO model with MDM and YIM, we derived MVF and InF of the SKK-30 hybrid model as:

$$m_{h30}(t) = x(1 - e^{-yt}) + a(1 - (b/(b+t))^c) + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.67)$$

$$\lambda_{h30}(t) = xye^{-yt} + acb^c (b+t)^{(1-c)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.68)$$

3.3.31. Sharma, Kumar & Kaswan-31 Hybrid Model

Combining the YIM with MD and YE models, we derived the MVF and InF of the SKK-31 hybrid model as:

$$m_{h31}(t) = \frac{x(1-e^{-yt})}{1+ze^{-yt}} + a(1 - (b/(b+t))^c) + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.69)$$

$$\lambda_{h31}(t) = \frac{xye^{-yt}(1+z)}{(1+ze^{-yt})^2} + acb^c (b+t)^{(1-c)} + \alpha rp\beta e^{-rp(1-e^{-\beta t})-\beta t} \quad (3.70)$$

3.3.32. Sharma, Kumar & Kaswan-32 Hybrid Model

Combining LGC, MDM, and YE models, we derived the MVF and InF of the SKK-32 hybrid model as:

$$m_{h32}(t) = \frac{x}{(1+ze^{-yt})} + a(1 - (b/(b+t))^c) + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.71)$$

$$\lambda_{h32}(t) = \frac{xyze^{-yt}}{(1+ze^{-yt})^2} + acb^c (b+t)^{(1-c)} + \alpha rp\beta e^{-rp(1-e^{\beta t})-\beta t} \quad (3.72)$$

3.3.33. Sharma, Kumar & Kaswan-33 Hybrid Model

Combining the Yamada Delayed model with MDM and YIM, we derived MVF and InF of the SKK-33 hybrid model as:

$$m_{h33}(t) = x(1 - (1+yt)e^{-yt}) + a(1 - (b/(b+t))^c) + \alpha(1 - e^{-rp(1-e^{\beta t})}) \quad (3.73)$$

$$\lambda_{h33}(t) = xy^2te^{-yt} + acb^c (b+t)^{(1-c)} + \alpha rp\beta e^{-rp(1-e^{\beta t})-\beta t} \quad (3.74)$$

3.4. SELECTION FUNCTION

Since it is common to multi-develop models, we developed 33 models and evaluated the value of their parameters. After discarding models having homogeneous failure rates and negative parameter values, we were left with ten suitable candidate models. To select the best estimator out of developed ten we developed an estimation function based on prediction accuracy. The estimation function calculated the error E_i between observed and estimated values up to a five-point dimension for $i=0$ to 5. Five-point dimension specifies six error $E_0, E_1, E_2, E_3, E_4, E_5$ where E_0 specifies no error means when the model's forecast exactly matches the observed data. Errors E_1, E_2, E_3, E_4 & E_5 specify the 1-point deviation till the five-point deviation means when the discrepancy between the two sets of numbers is 1,2,3,4 and 5 respectively. We also used a weighted function W based on the total match till the five-point dimension was calculated using error deviation from zero to five as

$$W = E_{i-1}w_i \quad \text{----(3.75)}$$

Where $w_i, i = 1 \text{ to } 6$ are the weights assigned to errors having values as 10,8,6,4,2,1 corresponding to errors $E_0, E_1, E_2, E_3, E_4, E_5$ respectively. The highest weight 10 is given to no error condition when the model accurately estimates the observed value. We then decrease the weights uniformly by a difference of 2 till we have weight 1 corresponding to a maximum deviation that is 5.

3.5. COMPARISION CRITERIA

How well a model estimates the failure data determines its performance. We developed an estimation function to find the top design from the few that have been perfected based on accurate estimation. Oval is used to represent the observed fault data and Eval to represent the estimated fault values predicted by the model. For comparing the model's capability following are the thirteen measures for its performance criteria utilized in our research work.

3.5.1. Mean Value (bias)

The average cumulative discrepancy between predicted and measured quantities is calculated using bias. For best prediction ability a model must have a small value of bias in comparison to other models. The only drawback of this criteria is that if the dataset includes both positive and negative values then while taking summation, they tend to cancel out each other and give the wrong prediction. When working with data sets of varying polarities, it is preferable to calculate estimates and predictions using the absolute value of the mean.

$$\sum_{i=1}^n (Eval(i) - Oval(i)) / n \quad (3.76)$$

3.5.2. Mean Square Error (MSE)

When comparing observed and estimated values, MSE calculates the sum of the squares of the differences. Small values represent better prediction. It also provides the best-fit line for the data set.

$$\sum_{i=1}^n \frac{(Oval(i)-Eval(i))^2}{n} \quad (3.77)$$

3.5.3. Mean Absolute Deviation (MAD)

Displays the cumulative average absolute difference between the values that were observed and those that were estimated. Small values of MAD indicate lesser deviation and better estimation capability of a model.

$$\sum_{i=1}^n \frac{|Oval(i)-Eval(i)|}{n} \quad (3.78)$$

3.5.4. Predictive Ratio Risk (PRR)

When comparing estimated and observed values, PRR provides a cumulative measure of the discrepancy for estimated values.

$$\sum_{i=1}^n \frac{(Eval(i)-Oval(i))}{Eval(i)} \quad (3.79)$$

3.5.5. Noise

Noise gives us the deviation in the value corresponding to the previous value. It is calculated by adding the variance between the estimated values at times t and t-1 with respect to the estimated value at time t-1.

$$\sum_i \frac{Eval(t_i)-Eval(t_{i-1})}{Eval(t_{i-1})} \quad (3.80)$$

3.5.6. Root Mean Square Error (RMSE)

The Root Mean Squared Error (RMSE) measures the average squared error. The value is calculated by taking the square root of the total of the squared differences between the observed and estimated values. Low values of RMSE indicate a better fit for a model.

$$\sqrt{\sum_{i=1}^n \frac{(Oval(i) - Eval(i))^2}{n}} \quad (3.81)$$

3.5.7. Relative Absolute Error (RAE)

When calculating the deviation of an estimated value from its mean, RAE calculates the total of the absolute differences between the actual and predicted values. A model is said to be a better predictor if this ratio value is near zero.

$$\frac{\sum_{i=1}^n |Oval(i) - Eval(i)|}{\sum_{i=1}^n |Eval(i) - \bar{Eval}|} \quad (3.82)$$

3.5.8. Root Relative Squared Error (RRSE)

Using the sum of the squared deviations of the value that was observed from the mean, RRSE calculates the square root of the difference among the predicted and measured values. By taking into account the dispersion of the estimated values from their mean value, RRSE normalizes the difference between the measured and predicted values. The smaller the RRSE value, the better the fitting model.

$$\sqrt{\frac{\sum_{i=1}^n (Eval(i) - Oval(i))^2}{\sum_{i=1}^n (Oval(i) - \bar{Oval})^2}} \quad (3.83)$$

3.5.9. Mean Magnitude of Relative Error (MMRE)

The MMRE is calculated as the absolute value of the standard deviation of the observable quantities. Calculated as the mean absolute difference between observed and estimated values for the same set of observations. When gauging the accuracy of a model's predictions, the MMRE number should be as near to zero as possible.

$$\frac{1}{n} \sum_{i=1}^n \left| \frac{Oval(i) - Eval(i)}{Oval(i)} \right| \quad (3.84)$$

3.5.10. Predictive Power (PP)

The PP value represents the total discrepancy between the estimated and observed values.

$$\sum_{i=1}^n \left(\frac{Eval(i) - Oval(i)}{Oval(i)} \right)^2 \quad (3.85)$$

3.5.11. Coefficient of Determination (R²)

R² is one minus the ratio of the sum of squared errors (between the actual and predicted values) to the sum of mean deviations (actual value). R² values close to 1 indicate a good match between data and a model.

$$1 - \frac{\sum_{i=1}^n (Oval(i) - Eval(i))^2}{\sum_{i=1}^n (Oval(i) - \bar{Oval})^2} \quad (3.86)$$

3.5.12. Accuracy of Estimation (AE)

AE is calculated by dividing the disparity between the reported and predicted cumulative values by the reported cumulative value [18]. Small values close to zero of AE yield a model that is well-fitting.

$$\frac{O(a)-E(a)}{O(a)} \quad (3.87)$$

3.5.13. Theil's Statistics (TS)

The evaluation of TS is expressed as a percentage. It computes the square root of the squared difference between recorded and predicted values, divided by the squared sum of recorded values [17]. Lower readings of TS indicate a better model fit.

$$\sqrt{\frac{\sum_{i=1}^n (Eval(i)-Oval(i))^2}{\sum_{i=1}^n Oval(i)^2}} \times 100\% \quad (3.88)$$

3.5.14. Comparison Criteria Values for all Proposed Models

The calculated criterion values for ten candidate models are displayed in Table 3.2. Analyzing the table reveals that model SKK-28 has the greatest number of minimum values compared to other candidate models meeting the specified criteria. SKK-31 has a minimum value in criteria PRR and SKK-7 have a minimum value in criteria RAE. The minimum value of MMRE was shown by model SKK-26. Model SKK-28 has minimum values in ten criteria measures as compared to thirteen. So, we can safely say that model SKK-28 is the best model out of the proposed ten candidate models.

Table 3.2. Comparison criteria of proposed candidate models

COMPARISON CRITERIA VALUES													
Models	Bias	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	R ²	AE	TS
SKK-1	5990	28000 0000	5990	353 00	742 00	16700	0.66	1830	1190	1.00	-3330000	621	1260
SKK-2	5910	27400 0000	5920	391 00	736 00	16600	0.66	1810	1170	0.51	-3270000	613	1250
SKK-3	18200	16800 00000	18200	25.7	158 000	41000	0.74	4470	3680	1.00	- 20000000	1890	3080
SKK-5	5940	27400 0000	5950	190 0	733 00	16600	0.66	1810	1180	1.21	-3260000	617	1240
SKK-7	2670	71200 000	2680	118 0	400 00	8440	0.62	921	522	0.81	-848000	277	635
SKK-11	3610	12200 000	3630	770 000	514 00	11000	0.63	1200	710	0.59	-1450000	375	830
SKK-25	-8.93	168	8.93	611 00	8.10	13.00	15.50	1.41	0.86	7780.00	-1.00	0.93	0.97
SKK-26	-8.92	167	8.92	394 00	6.96	12.90	18.10	1.41	0.86	0.97	-0.99	0.93	0.97
SKK-28	2.44	94.7	7.21	16.7	5.30	9.73	1.81	1.06	1.45	0.17	-0.13	0.25	0.73
SKK-31	3.54	103	7.72	14.6	6.16	10.10	1.81	1.11	1.61	4290000 000000	-0.23	0.37	0.76

3.6. RANKING METHODOLOGY

Additionally, we devised a methodology for ranking models based on the estimation precision. We used the estimation function up to ten point error difference. The above-mentioned criteria set was used to determine the thirteen criteria values for several models. The criteria matrix holds the value of all these criteria for all models.

$$\text{Criteria Matrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad (3.89)$$

Where $[a_{ij}]$ represents the j^{th} criteria value of an i^{th} model.

The models under consideration were ranked based on individual criteria. A rank matrix holds the rank of all m models corresponding to each of n criteria as

$$\mathbf{Rank} = \begin{bmatrix} \mathit{rnk}_{11} & \mathit{rnk}_{12} & \dots & \mathit{rnk}_{1n} \\ \mathit{rnk}_{21} & \mathit{rnk}_{22} & \dots & \mathit{rnk}_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \mathit{rnk}_{m1} & \mathit{rnk}_{m2} & \dots & \mathit{rnk}_{mn} \end{bmatrix} \quad (3.90)$$

Where $[r_{ij}]$ specifies the rank of the i^{th} model in the j^{th} criteria.

Weights were assigned to ranks in increasing order of ranks to determine weighted rank values so that models having lower ranks in maximum criteria should give a low value. The highest weight assigned was 11 and the lowest was 0.0001 corresponding to rank 11 and rank 1 respectively as tabulated below in Table 3.3: -

Table 3.3. Weight values assigned to the model's individual criteria rank

R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11
0.0001	0.0005	0.001	0.005	1	2	3	4	5	6	7

For each candidate model, the Weight Matrix was computed by replacing the rank with the associated weight according to the preceding table. To acquire a weighted rank matrix, the model's rank in each criterion was then multiplied by the associated weight.

$$\mathbf{Weight\ matrix} = \begin{bmatrix} \mathit{w}_{11} & \mathit{w}_{12} & \dots & \mathit{w}_{1n} \\ \mathit{w}_{21} & \mathit{w}_{22} & \dots & \mathit{w}_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \mathit{w}_{m1} & \mathit{w}_{m2} & \dots & \mathit{w}_{mn} \end{bmatrix} \quad (3.91)$$

Where $[w_{ij}]$ represents the weight of the i^{th} model rank in the j^{th} criteria.

Even though the rank matrix assigns rank according to the measure's value specifying the performance of a model, it might be the case that the two models still have the same graded weights because of specific rank combinations. In such situations,

it is possible that the model with the highest ranking is not the most accurate estimator. Thus, we need to compute a weighted rank matrix to eliminate any such possibility.

Weighted Rank Matrix = Rank Matrix * Weight Matrix

$$= \begin{bmatrix} \mathit{rnk}_1 & \mathit{rnk}_{12} & \dots & \mathit{rnk}_{1n} \\ \mathit{rnk}_{21} & \mathit{rnk}_{22} & \dots & \mathit{rnk}_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \mathit{rnk}_{m1} & \mathit{rnk}_{m2} & \dots & \mathit{rnk}_{mn} \end{bmatrix} * \begin{bmatrix} \mathit{w}_{11} & \mathit{w}_{12} & \dots & \mathit{w}_{1n} \\ \mathit{w}_{21} & \mathit{w}_{22} & \dots & \mathit{w}_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \mathit{w}_{m1} & \mathit{w}_{m2} & \dots & \mathit{w}_{mn} \end{bmatrix} \quad (3.92)$$

$$= \begin{bmatrix} \mathit{rnkw}_{11} & \mathit{rnkw}_{12} & \dots & \mathit{rnkw}_{1n} \\ \mathit{rnkw}_{21} & \mathit{rnkw}_{22} & \dots & \mathit{rnkw}_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \mathit{rnkw}_{m1} & \mathit{rnkw}_{m2} & \dots & \mathit{rnkw}_{mn} \end{bmatrix} \quad (3.93)$$

Where $[rw_{ij}]$ represents the weighted rank value of the i^{th} model in the j^{th} criteria. We calculated the weighted function using weights from minimum zero error difference E_0 to E_{10} , where E_{10} gives a maximum difference of 10 between observed and estimated values according to the following Table 3.4: -

Table 3.4. Weighted estimation vector

Ei	0	1	2	3	4	5	6	7	8	9	10
Wi	20	18	16	14	12	10	8	6	4	2	1

The order of the models was determined by dividing the sum of rows of the weighted rank matrix by the weighted estimation function. Rank 1 represents the best model having maximum values in criteria corresponding to best fit as compared to other models.

$$R_j = \frac{\sum_{i=1}^n \mathit{rw}_{ji}}{\sum \mathit{WF}_j}, \text{ where } j = 1 \text{ to } m \quad (3.94)$$

The complete ranking methodology based on accurate estimation is represented below in Figure 3.1.

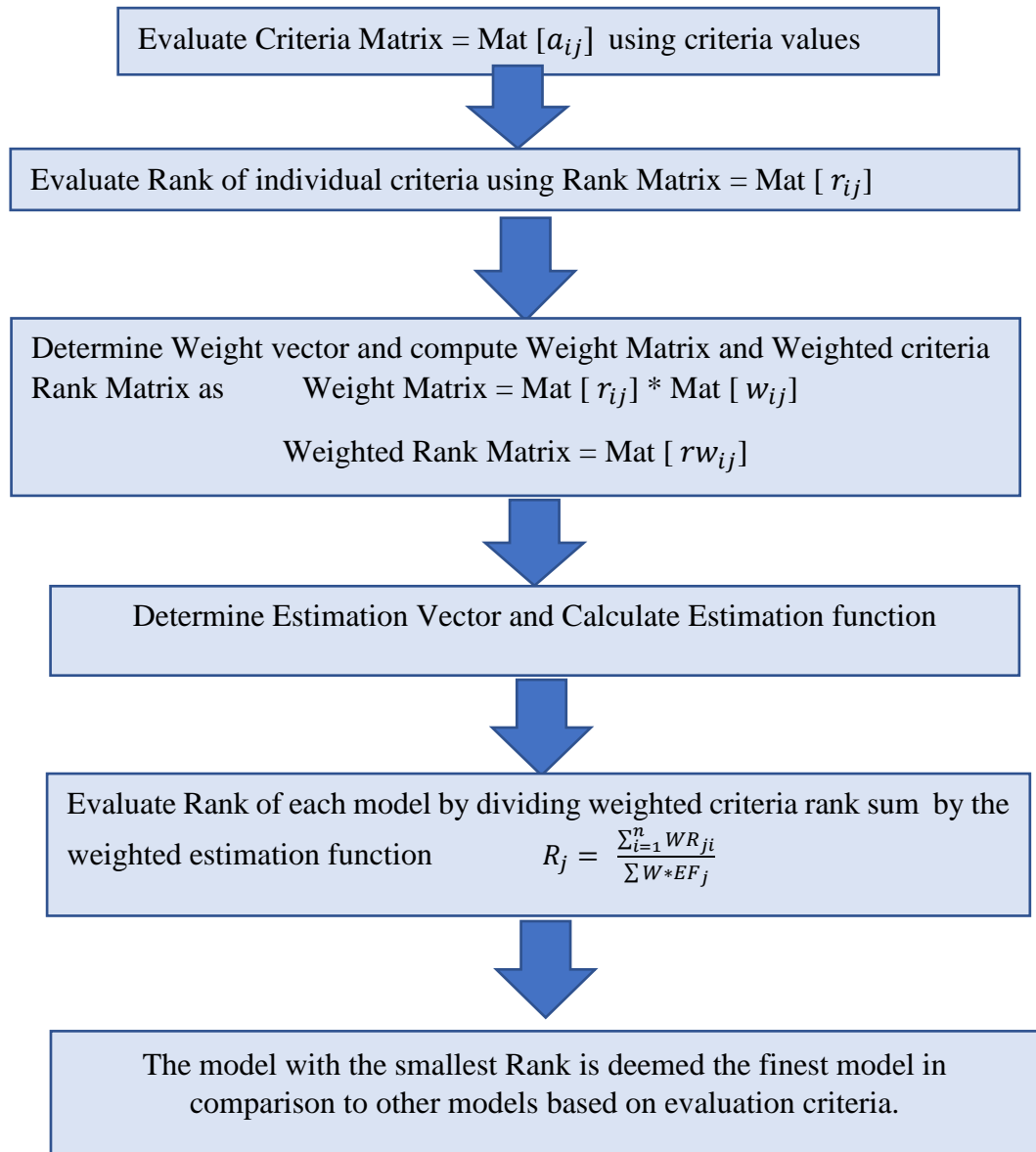


Figure 3.1. Flowchart of ranking methodology

CONCLUSION

Fault-free execution of software for a significant period determines its reliability under given conditions. These faults are generally caused due to incorrect specifications during the design phase or coding errors. Due to technological advancement and big data, the storage capacity and access mechanism undergo tremendous change to incorporate its characteristics. Traditional reliability models mostly consider pure software errors during reliability analysis. To incorporate induced errors in software due to hardware and human error 33 hybrid models were developed. Since a hybrid NHPP model can be obtained by adding the MVF and InF of two or more NHPP models, proposed hybrid models consist of three NHPP models first for handling pure software errors, second for handling hardware-induced errors in software, and third to handle induced errors in software due to the human factor.

CHAPTER 4

EXPERIMENTAL WORK

This chapter describes datasets used in research work for parameter evaluation, parameter optimization, and determining the ranking of reliability models. Different failure datasets taken from the literature represent the observed faults and cumulated faults time of fault (in days/weeks) for a specific period.

4.1. FAILURE DATA SETS

A group of related information sets consisting of various components which the system can manipulate as a unit is called a data set. Four datasets, including one big fault data, are used. Graphical representations of these datasets concerning faults and cumulative faults are shown to understand the failure behavior. All models and parameter evaluation techniques require data regarding the time at which failures occurred. In early-developed models, calendar time was utilized by researchers instead of execution time. The undermentioned Table 4.1 contains datasets used in this research to evaluate parameter values and validate the developed models. The focus of our study is to utilize these datasets in creating a hybrid model for accurate reliability estimation.

Table. 4.1. Fault datasets used by researchers in developed models

S.No	Data Sets	Author
1.	DS#1	Govindasamy and Dillibabu (2018)
2.	DS#2	Peng, Hu and Ng (2008)
3.	DS#3	Minohara and tohma (1995)
4.	DS#4	Lyu (1996)

4.1.1. Data Set #1 (DS1)

Data was collected through big data analysis (Govindasamy and Dillibabau,2018). The data set includes testing duration in days, the number of failures, and the cumulative number of failures for 30 days, as displayed in Table 4.2 shown below. A total of 289 cumulative failures were reported.

Table 4.2. Failure data set collected using big data analysis

Testing time (In days)	Total Failures	Cumulative no. of failure
1	6	6
2	4	10
3	7	17
4	3	20
5	9	29
6	5	34
7	4	38
8	3	41
9	9	50
10	5	55
11	14	69
12	29	98
13	4	102
14	6	108
15	9	117
16	1	118
17	45	163
18	14	177

19	9	186
20	6	192
21	9	201
22	3	204
23	5	209
24	23	232
25	3	235
26	5	240
27	8	248
28	14	262
29	22	284
30	5	289

The graph shown below in Figure 4.1 is plotted to examine the behavior of the failure data set DS1. Y-axis represents the failures while X-axis represents the time of fault measured in days. The orange line shows the pattern of cumulative errors while the blue line depicts the behavior of the number of failures per day.

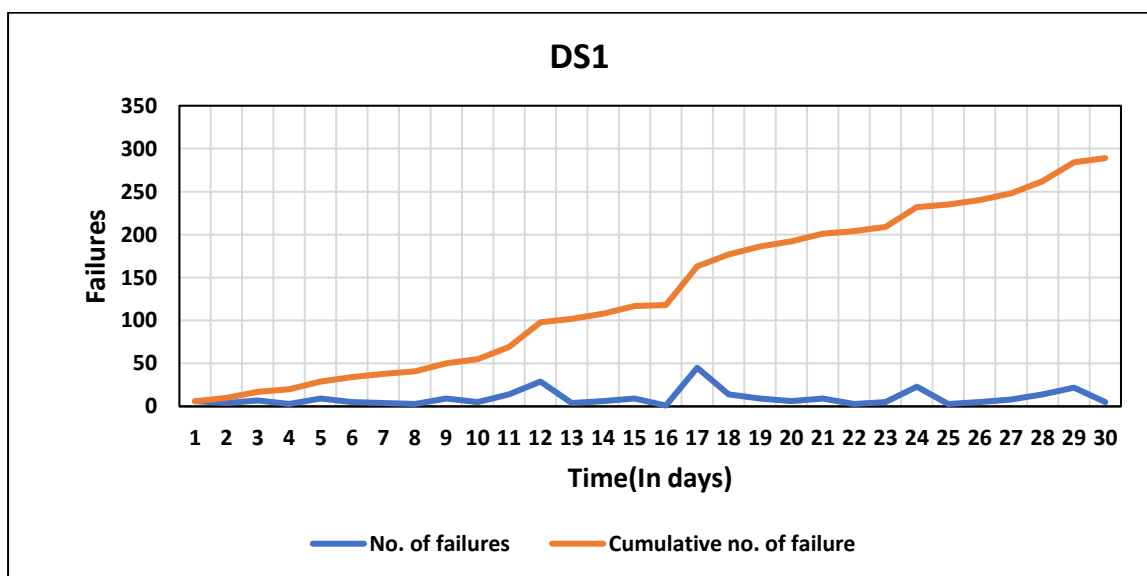


Figure 4.1. Number of failures and cumulative failures with time (days) of DS1

4.1.2. Data Set #2 (DS2)

This data set (2008) was utilized by Peng, Hu, and Ng during the assessment of a medium-sized software venture. The data set contains testing time in weeks, and fault detection, and correction data for 21 days. Only the fault detection and cumulative fault data were utilized from the dataset. A total of 191 cumulative failures were reported as shown in Table 4.3. The distribution of the failure data set is illustrated via a graph.

Table 4.3. The data set of a middle-size project

Time Testing time (In weeks)	No. of failures	Cumulative no. of failure
1	15	15
2	20	35
3	25	60
4	14	74
5	20	94
6	8	102
7	12	114
8	20	134
9	5	139
10	2	141
11	7	148
12	1	149
13	8	157
14	15	172
15	6	178
16	3	181

17	2	183
18	1	184
19	2	186
20	4	190
21	1	191

The graph shown below in Figure 4.2 is plotted to examine the behavior of the failure data set DS2. Y-axis represents the cumulative failures while X-axis represents the time of fault measured in days. The orange line shows the pattern of cumulative errors while the blue line depicts the behavior of the number of failures measured per day.

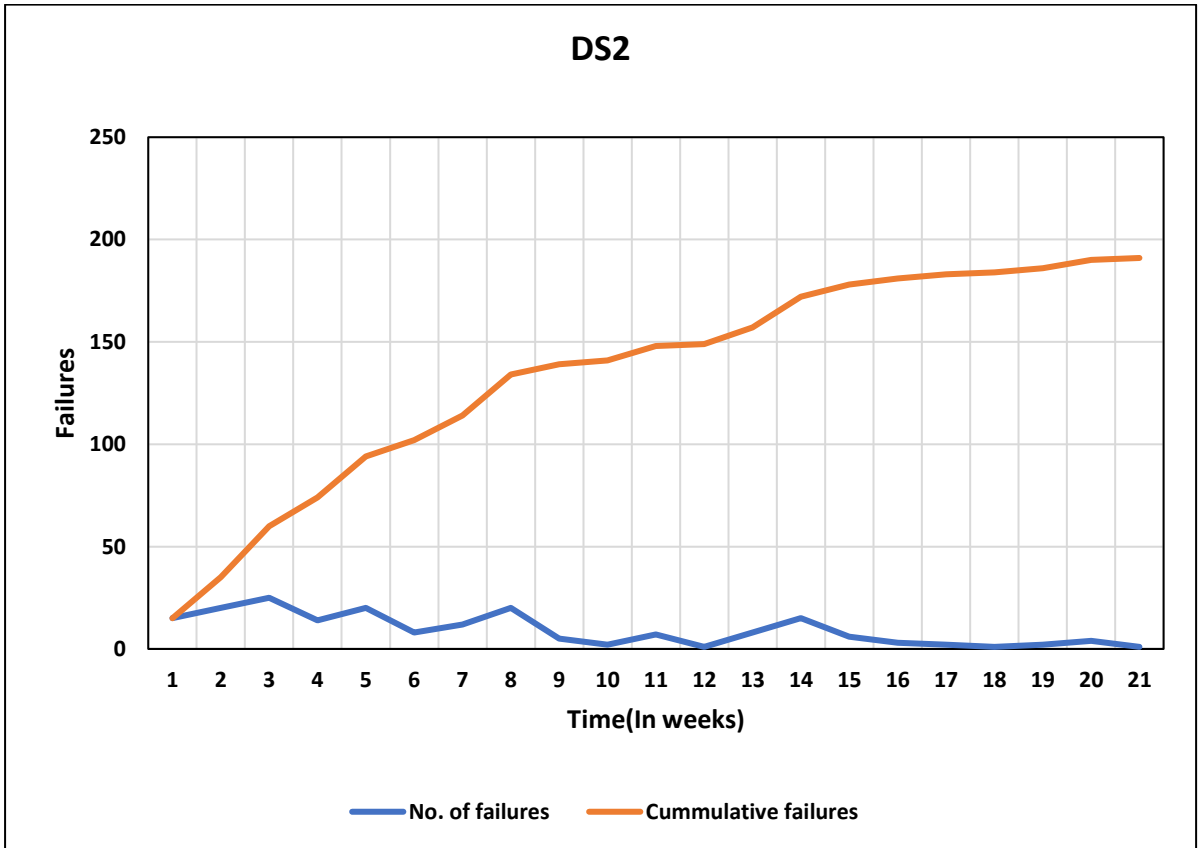


Figure 4.2. Number of failures and cumulative failures with time (days) of DS2

4.1.3. Data Set #3 (DS3)

This dataset represents real test and debugs data. The data set contains testing time in days and is tabulated below in Table 4.4 along with its graphical representation. The total cumulative faults detected were 534 over 109 days.

Table 4.4. Real-time control application dataset

Time Testing time (In weeks)	No. of failures	Cumulative no. of failure
1	4	4
2	0	4
3	7	11
4	10	21
5	13	34
6	8	42
7	13	55
8	4	59
9	7	66
10	8	74
11	1	75
12	6	81
13	13	94
14	7	101
15	9	110
16	8	118
17	5	123
18	10	133
19	7	140
20	11	151

21	5	156
22	8	164
23	13	177
24	9	186
25	7	193
26	7	200
27	5	205
28	7	212
29	6	218
30	6	224
31	4	228
32	12	240
33	6	246
34	7	253
35	8	261
36	11	272
37	6	278
38	9	287
39	7	294
40	12	306
41	12	318
42	15	333
43	14	347
44	7	354
45	9	363
46	11	374
47	5	379
48	7	386
49	7	393
50	14	407

51	13	420
52	14	434
53	11	445
54	2	447
55	4	451
56	4	455
57	3	458
58	6	464
59	6	470
60	2	472
61	0	472
62	0	472
63	3	475
64	0	475
65	4	479
66	0	479
67	1	480
68	2	482
69	0	482
70	1	483
71	2	485
72	5	490
73	3	493
74	2	495
75	1	496
76	11	507
77	1	508
78	0	508
79	2	510
80	2	512

81	4	516
82	1	517
83	0	517
84	4	521
85	1	522
86	1	523
87	0	523
88	2	525
89	0	525
90	0	525
91	1	526
92	1	527
93	0	527
94	0	527
95	0	527
96	0	527
97	1	528
98	0	528
99	1	529
100	0	529
101	0	529
102	0	529
103	0	529
104	2	531
105	0	531
106	1	532
107	0	532
108	2	534
109	0	534

The graph shown below in Figure 4.3 is plotted to examine the behavior of the failure data set DS3. Y-axis represents the cumulative failures while X-axis represents the time of fault measured in days. The orange line shows the pattern of cumulative errors while the blue line depicts the behavior of the number of failures measured per day.

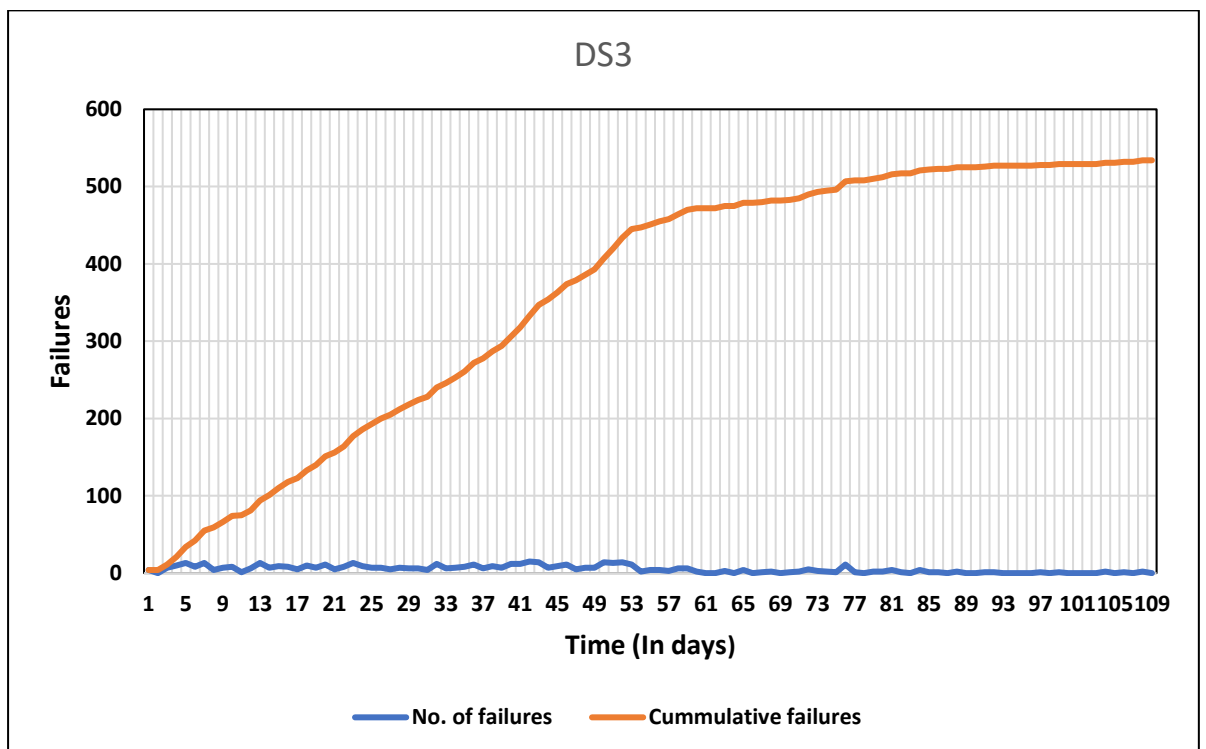


Figure 4.3. Failures and cumulative failures plot concerning time for DS3

4.1.4. Data Set #4 (DS4)

This data set is derived from Lyu's (1996) handbook on software reliability. The testing time is given in weeks. We calculated the cumulative failures and tabulated the data set in Table 4.5. The total cumulative faults detected were 133 over 62 days.

Table 4.5. Failure count data

Time Testing time (In weeks)	No. of failures	Cumulative no. of failure
1	6	6
2	1	7
3	1	8
4	0	8
5	1	9
6	3	12
7	0	12
8	5	17
9	6	23
10	1	24
11	0	24
12	3	27
13	9	36
14	3	39
15	2	41
16	3	44
17	0	44
18	2	46
19	4	50
20	0	50
21	0	50
22	0	50
23	0	50
24	0	50
25	2	52
26	0	52

27	0	52
28	2	54
29	3	57
30	11	68
31	5	73
32	3	76
33	2	78
34	4	82
35	2	84
36	0	84
37	1	85
38	2	87
39	0	87
40	1	88
41	3	91
42	0	91
43	1	92
44	6	98
45	2	100
46	0	100
47	5	105
48	10	115
49	3	118
50	3	121
51	2	123
52	1	124
53	0	124
54	3	127
55	0	127
56	2	129

57	0	129
58	1	130
59	0	130
60	1	131
61	0	131
62	2	133

The graph shown below in Figure 4.4 is plotted to examine the behavior of the failure data set DS4. Y-axis represents the cumulative failures while X-axis represents the time of fault measured in days. The orange line shows the pattern of cumulative errors while the blue line depicts the behavior of the number of failures measured per day.

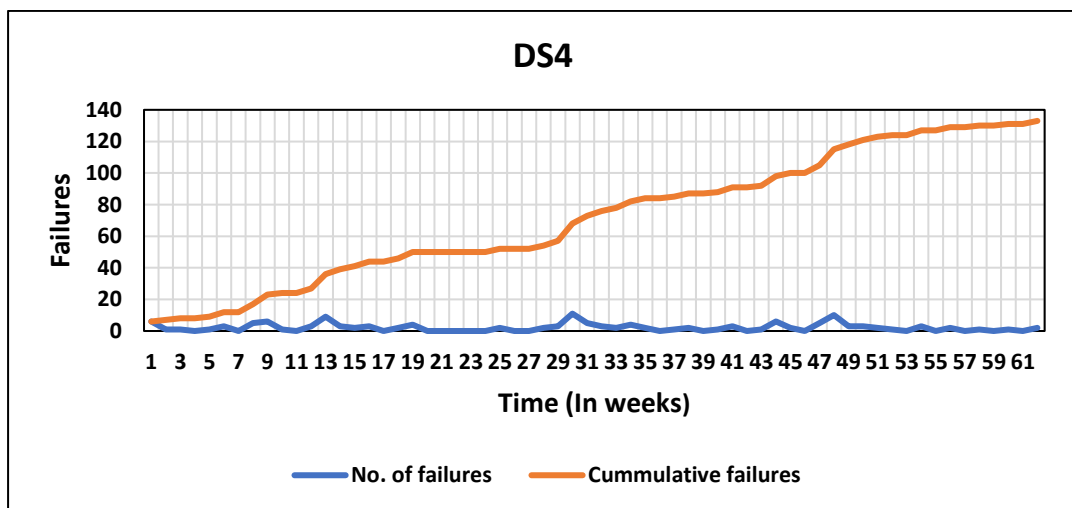


Figure 4.4. Number of failures and cumulative failures with the time of DS4

4.2. PARAMETER EVALUATION

Using the right technique for parameter evaluation is crucial for model design. Reliability and failure behavior are estimated using various methods or a combination of techniques to get an accurate estimation using precise parameters value. The reliability estimation and model validation all depend upon the true value of the

parameters. Using large datasets, statistical methods provide a quantifiable measurement of parameter values, which is then utilized to assess the model's fitness. The good fit specifies how precisely a developed model reflects the observed data. To figure out the gap between experimental and predicted values, it is necessary to minimize a given objective function. If the developed model is non-linear, it will be very difficult to estimate parameters value accurately using statistical methods. For non-linear and complex models, hybrid techniques employing statistical methods in combination with soft computing techniques were developed by researchers in the literature. In situations where the statistical estimation is inaccurate, the parameters' values were optimized using soft computing methods. LSE and MLE are the two most frequently employed statistical methods for estimating parameter values. Commonly used soft computing techniques for optimization are GA, PSO, NN, fuzzy system, Gray-Wolf, Firefly, Cuckoo search, Sparrow search, etc. Using any technique has some advantages and disadvantages, and none is 100% accurate. It's up to the user to employ one that is best suited for their model and datasets.

Reliability modelling parameter estimation uses MLE and LSE. Uniformity and adequateness make MLE a popular standard technique. LSE works well for linear models with moderate or small amounts of data and is assessed mainly using linear regression. LSE minimizes the observed-estimated divergence to assess parameters.

4.2.1. Least Square Estimation

The LSE better predicted small data sets because of either a smaller bias or a faster normality approach. The goal of LSE is to reduce the relative difference between measured and predicted values as much as possible. The intensity function value, or rate of error, is used to estimate the model's parameters.

$$\mathbf{LSE} = \sum_{i=1}^n ((\lambda(t)_{Eval} - \lambda(t)_{Oval})^2) \quad (4.1)$$

4.2.2. Maximum Likelihood Estimation

Fisher (1920s) first introduced the principle of MLE. He claims that the optimal choices for the parameters are those that maximize the loglikelihood function, after which the likelihood function L may be derived from the intensity function $\lambda(t)$ using the following equation as

$$L = \prod_{i=1}^n \lambda(t_i) \quad (4.2)$$

As stated by Pham (2006) the log of L can be derived as

$$\text{Log } L = \log \left(\prod_{i=1}^n \lambda(t_i) \right) \quad (4.3)$$

$$= \sum_{i=1}^n \log (\lambda(t_i) - m(t_n)) \quad (4.4)$$

4.2.3. Algorithms Used in Evaluating Parameters

The preceding formula is differentiated m times for every parameter with regards to time, leading to m equations to be solved for a model with m parameters. Algorithms for solving m equations are then utilized to solve these equations. The parameter accuracy thus now depends upon the precision of these methods to overcome local minima and find suitable values. Newton Raphson, Finite Difference, and Brayden's method are some numerical methods used to solve non-linear equations. Many software platforms have inbuilt solvers to solve non-linear equations, but how and when to use them is not discussed in the literature, making it very hard to explore all possible methods which might give better results. We use MATLAB solver FSOLVE with the default Levenberg-Marquardt algorithm to evaluate parameters value.

4.3. SOFT COMPUTING OPTIMIZATION

Soft computing is emerging as a popular solution for optimizing parameters value. Recently, many researchers have employed several soft computing approaches for optimization and reliability evaluation in software engineering. These include genetic

algorithms (GA), neural networks (NN), fuzzy logic (FL), support vector machines (SVMs), particle swarms (PS), ant colonies (ACs), gray wolves (GWs), cuckoo swarms (Cs), sparrow swarms (SS), and artificial bee colonies (ABCs), among others. Soft-computing methods exploit uncertain, approximate, partial, and imprecise behavior to obtain practical, robust, and low-cost reliability models. Fuzzy sets, artificial neural networks, and evolutionary algorithms are the three primary subfields of soft computing. Research on various applications of soft computing for reliability assessment and parameter optimization has been carried out in recent years. Undermentioned are some of these techniques:

4.3.1. Genetic Algorithm (GA)

The genetic algorithm (GA) is a method for optimizing evolutionary processes that are based on genetics and evolution. It is a subset of evolutionary computations and uses natural selection in biology evolution to solve the problem using the global optimization search method. GA was introduced by Holland (1973,1975), a pioneer of GA, and his students and colleagues. Inspired by Darwin's theory of evolution, it was further refined by Goldberg (1989), who used it with a higher success rate to solve various optimization problems. Initially, a pool of possible solutions regarding the stated problem is defined in GA called population. These multiple solutions are then mutated to reproduce a new generation as in natural genetics. The process of producing children is repeated over many generations by assigning a fitness value to each candidate solution. According to Darwin's principle of survival of the fittest, healthier people have a greater chance of finding a suitable partner and producing healthier offspring. This process continued over generations, and better candidates kept evolving until a stopping criterion was met. The nature of GA is random, and it makes use of knowledge about the past to resolve a particular issue. In the population pool, each potential solution is represented by a chromosome, and the fitness function is used to determine its position in the ranking. From this pool of individuals, the parents are picked using fitness-based selection criteria. Offspring with improved chromosomal counts and fitness levels are produced by applying operations like crossover and mutation to the parent. Over several

generations, a chromosome's fitness value determines which mutations are picked, and ultimately, the optimal solution to the issue. GA is used widely in various areas, i.e., cryptography, Sequence scheduling, asset allocation, robotics, etc. The bottleneck of GA is the selection and coding of fitness functions. Wrong choice of mutation parameters and crossover rate results in bad output.

4.3.2. Fuzzy Logic (FL)

Zadeh (1965) introduced FL. Fuzzy logic imitates human decision-making by considering vague and imprecise information to solve real-world problems. It maps statistical input to scalar data. It's logical to describe fuzziness by using the fuzzy Inference system(FIS) to make decisions. The FIS relied on IF/THEN principles and OR/AND logic to make its inferences. The outcome of FIS will always be fuzzy, even if the input provided is fuzzy or crisp. FIS uses a defuzzification unit to convert the fuzzy variables to crisp variables. Rule base (RB), database (DB), decision-making unit(DMU), fuzzification interference unit (FIU), and defuzzification unit (DFU) are the five primary functional components that make up a FIS. The RB contains the fuzzy rules in IF/THEN form, whereas the database defines the membership function. Rules are used by the DMU to guide its actions. FU and DFU are used to convert the crisp values into fuzzy values and vice-versa, respectively. The main application area of fuzzy logic is in controllers of speed and temperature units of various devices. Making fuzzy rules for huge, complicated issues is arduous and time-consuming, which is a major drawback of fuzzy logic.

4.3.3. Neural Network (NN)

NN is modelled after the biological neural network, which learns and changes itself based on the provided inputs. Learning and improving the property of NN affects its structure on the basis of information flows through it. NN is utilized for trend detection and complex pattern extraction. To solve a problem, NN uses a network of many linked neurons that function and process data in the same way as a human brain does. NN is

classified into feed-forward (FFNN) and feed-backward (FBNN) networks. In FFNN, the information flows in one direction only, and it may be single-layered or multi-layered. Single-layer perceptrons have three distinct layers: input, hidden, and output whereas multi-layered NN have more than one hidden layer beside other layers. Feedback NN supports the bi-directional flow of information from input to output as well as from output to hidden layers. Various layers are connected through weights which are modified by using a learning algorithm. NN employed supervised and unsupervised learning techniques and can learn by example data. NN-based reliability models can be categorized as non-parametric and better predictors with fault tolerance to noisy data.

4.3.4. Support Vector Machines (SVM)

SVM is another learning algorithm proposed by Vapnik (1998,2000). It consists of an optimal hyperplane for linearly separable patterns, including kernel function as a support vector for pattern recognition. SVM provides an optimal solution by maximizing the margin near the hyperplane and specifying the decision function by supporting vectors and training samples subset. SVM gives better performance as its answer is absent from local minima, unlike other non-linear optimizers that get stuck in local minima. The major drawback of SVM is its high computation cost.

4.3.5. Particle Swarm Optimization (PSO)

PSO, introduced by Kennedy and Eberhart (1995) is a population-based stochastic metaheuristic nature-inspired optimization algorithm. PSO starts with a swarm which is nothing but a randomly generated set of particles denoting a probable solution vector for the problem to be optimized. Each particle assesses its location based on velocity, previous position, and neighboring information in the multi-dimensional search space. Thus, the behavior of a single particle contributes toward the collective behavior of the swarm to which it belongs. Each particle in the search space represents a possible solution with fitness corresponding to the objective function. Classical PSO converges

quickly, but complicated problems might trap it in local minima, and prematurely converge. PSO is a robust stochastic search technique used in many application areas like data mining, telecommunications, power systems, combinatorial optimization, etc. PSO is more flexible than GA and can control and balance the local and global exploration of search space.

4.3.6. Simulated Annealing (SA)

SA is another optimization technique that is useful in finding global optima when a large number of local optima are present. Annealing word is borrowed from thermodynamics which specifies the heating of metal and then cooling it to alter its physical properties. Metal keeps its changed structure and characteristics after cooling. SA optimizes its objective function rather than material energy and temperature is treated as a variable to simulate the heating and cooling process when the temperature is high, the algorithm accepts more solutions with high frequency, allowing it to escape local optima. As temperature drops, the algorithm must concentrate on the search space where the near-optimal solution may be located to avoid accepting an inferior answer.

4.4. PARAMETERS EVALUATION AND OPTIMIZATION USING SOFT COMPUTING METHODS AND DS#1

Since all models we developed are non-linear and complex in nature with a number of parameters ranging from six(min) to ten(max), we use MLE to obtain model parameters. The value of the initial guess vector starts with 0.025 for all variables, incremented by 0.025 till all guess vectors become 2. After trying various possible permutations and combinations for parameter guess values, we utilized eighty guess vectors as models were not converging to a single solution. Since we were getting numerous minima for different values of parameters, we wanted to select one value that gives the best possible outcome.

4.4.1. Proposed Methodology for Parameter Selection

Sharma et al. (2010) ranking technique was used to choose the parameter with the best estimate among the many possibilities. The algorithm tries to determine the minimum distance of all feasible solutions from the optimal solution. Considering the observed values as optimum vectors, we determine the distance of each feasible solution vector (estimated values vector corresponding to each of eighty parameter values) from the optimal vector and arrange them in the increasing order of their distance. Distance, in our case, is the error between the observed and estimated value vectors. Undermentioned is the adopted methodology

Ranking_Parameter ()

{

Matrix I(n, m) ← Intensity value of all m values of parameters for t= 1 to n.

Matrix O(n) ← Observed values (Optimal vector)

Matrix P(m, k) ← m parameter vector of length k

Delete P ← Delete Entries corresponding parameters having negative values from possible candidate parameter vectors

Delete (P, I) ← Delete Entries regarding parameters giving constant or zero Intensity distribution.

Update P ← Substitute 0 by 0.0001 as we assumed the value of all parameters to be positive.

Matrix R ← Determine the ranking of each m parameter value for m initial guess vector using a ranking algorithm.

Sort (PI) ← Sort parameters index according to their ranks in ascending order.

Parameter ← P(PI) Select the parameter with the lowest rank.

}

4.4.2. Parameters of all Proposed Models

The undermentioned Table 4.6. reflects the parameters evaluated using MLE for all developed hybrid models. The proposed selection methodology was adopted to choose parameter values giving the best estimate out of 80 values calculated using 80 guess vectors.

Table 4.6. Evaluated parameters value using MLE of proposed hybrid models

Models	Evaluated Parameters of Proposed Models									
SKK-1	0.00	0.00	0.09	4.31	232.85	126.46				
SKK-2	0.52	0.03	0.09	4.30	231.89	126.79				
SKK-3	0.26	0.08	0.08	25.34	5.48	342.35	118.85			
SKK-4	0.00	0.26	0.10	0.00	0.38	0.58				
SKK-5	0.00	1.97	0.09	4.43	239.26	126.18	0.34			
SKK-6	1.47	2.71	0.10	0.00	1.00	1.53				
SKK-7	0.00	0.01	0.09	3.26	187.22	132.02	1.29			
SKK-8	49.50	0.00	0.00	0.40	0.10	2605.80				
SKK-9	0.07	0.01	4.35	234.50	0.09	126.33	22.75			
SKK-10	0.13	1.29	0.10	671.24	0.00	0.36	0.36			
SKK-11	0.07	0.26	0.09	3.77	199.71	130.66	0.03	0.03		
SKK-12	0.70	1.70	1.70	1.70	1.63	0.54	1.69			
SKK-13	0.38	0.93	38.66	0.01	305.34	0.00	0.57			
SKK-14	0.90	1.00	0.00	0.10	32104.00	0.00				
SKK-15	0.00	0.18	0.00	0.00	0.23	0.86	1.17			
SKK-16	17.20	0.00	0.00	0.30	0.20	0.90	3622.80			
SKK-17	1.65	1.65	1.65	1.65	5.54	3.39	0.28	1.65		
SKK-18	16.70	0.10	0.00	0.00	54.00	8140.90	22.30	6.60		
SKK-19	3.1	0.1	0.1	-0.1	50.2	6400.9	19.3		3.1	0.1
SKK-20	0.0001	0.0001	-9378	0.0001	30	670	-10	-20	0.0001	0.0001
SKK-21	0.0001	0.0001	-7114	0.0001	30	570	10	10	0.0001	0.0001
SKK-22	2.00	2.00	2.00	1.89	0.51	2.00	2.00	2.00	2.00	
SKK-23	0.03	1.79	0.83	0.23	0.86	0.01	0.10	0.01	0.01	
SKK-24	27.96	0.01	0.05	0.95	0.24	0.02	0.24	0.24		
SKK-25	0.01	0.05	4.67	251.20	0.02	0.17	0.86	0.47	0.47	

SKK-26	0.38	0.07	15.43	44.82	0.10	0.13	0.85	0.42	0.42	
SKK-27	0.72	2.81	0.23	0.86	0.00	0.11	0.00	0.00		
SKK-28	0.11	0.01	5.47	4.63	2.28	0.25	0.03	0.01	0.03	0.03
SKK-29	25.80	0.10	0.30	50.90	7660.10	22.80	0.00	0.00	0.00	0.00
SKK-30	0.00	0.00	51.00	22427.00	69.00	0.00	0.00	0.00	0.00	
SKK-31	7.99	10.25	207.75	4.85	2.48	0.26	0.06	0.01	0.07	0.07
SKK-32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
SKK-33	0.00	0.00	50.30	6325.30	19.10	0.00	0.00	0.00	0.00	

4.4.3. Parameters of Existing NHPP Models

Parameter values of some existing models were calculated using FSOLVE with default trust-region-dogleg or Levenberg-Marquardt algorithm and Parameter ranking in Table 4.7.

Table 4.7. Evaluated values of parameters of traditional models using MLE

Models	Parameters Value					
GG	3.4973	3.0321	0.0683			
GMPZ	5.6556	0.0001	0.0001			
GO	65.4959	0.0021				
YMD	35.9874	0.0112				
YMI	41.3078	0.0113	8.3713			
GD-1	0.2	0.01	2910	0.0001	0.1	
GD-2	0.0016	0.0157	703.9721	0.0003	0.099	
GD-3	0.0228	0.3871	1.8808	512.0351	0.0005	0.0991

4.4.4. Parameters Evaluated using LSE, GA, SA & PSO

The LSE method was utilized to calculate the parameter value of selected ten proposed models and well-known traditional models in Table 4.8.

Table 4.8. Parameters evaluated of proposed ten best models using LSE

Models	Pure S/W Parameters				H/W induced Parameters			User induced Parameters			
	x	Y	z	w	a	b	c	x1	x2	p	r
SKK-1	0.3144	0.0158			5.8329	1.2452	0.7874	0.1000			
SKK-2	1.7290	1.8794			2.1661	2.5305	0.4991	2.7218			
SKK-3	0.8343	0.8225	0.8259		2.5579	2.7020	0.2538	1.2642			
SKK-5	0.9563	0.9490	0.9544		2.8189	2.0014	0.5697	1.8561			
SKK-7	1.8481	1.8536	1.8467		2.1641	2.5686	0.5288	2.7400			
SKK-11	1.8498	1.8506	1.8505	1.8505	2.1671	2.5727	0.5291	2.7410			
SKK-25	1.4775	1.4748	1.4754		1.5441	1.5182		1.4757	1.4741	1.4745	1.4745
SKK-26	1.5513	1.0361	1.6980		3.6137	1.2840		1.9324	1.7655	1.7655	1.7655
SKK-28	1.7256	1.7249	1.7245		2.9901	2.6318	0.6384	1.7252	1.7249	1.7247	1.7247
SKK-31	0.2891	0.0916	0.1554		6.9368	2.4461	0.0838	0.1529	0.1489	0.1529	0.1529

GA was used to generate the parameters of 10 suggested models and well-known classic models using LSE as an objective function in Table 4.9. given below.

Table 4.9. Parameters evaluated of proposed ten best models using GA

Models	Parameters										
	PURE S/W				H/W induced			User Induced			
	x	y	z	w	a	b	c	x1	x2	p	r
SKK-1	-110.94	4.72			5.67	-10.08	0.33	36.27			
SKK-2	-9.50	4.40			-1.79	-4.96	-0.32	2.45			
SKK-3	-8.69	-0.24	0.87		-5.88	-7.70	0.40	1.13			
SKK-5	9.99	129.76	-3.54		1.86	-13.80	0.69	19.48			
SKK-7	-11.40	2.30	-133.73		2.44	-5.25	0.49	8.86			
SKK-11	5.06	17.71	-5.49	2.28	-1.27	-7.78	-0.45	8.46			
SKK-25	102.81	-0.40	5.22		4.54	1.22		84.83	1.63	-0.79	-4.51
SKK-26	-44.01	0.47	10.07		7.90	1.09		64.77	20.74	-1.43	6.76
SKK-28	-61.61	-3.48	-2.20		-45.24	-0.19	0.81	46.15	0.00	-2.79	21.65
SKK-31	5.50	1.52	80.95		-3.23	-4.73	0.75	45.44	0.13	-0.28	-2.59

SA was used to generate the parameters of 10 suggested models and well-known classic models using LSE as an objective function in Table 4.10. given below.

Table 4.10. Evaluated parameters of proposed ten best models using SA

Models	Parameters										
	PURE S/W				H/W induced			User Induced			
	x	y	z	w	a	b	c	x1	x2	p	r
SKK-1	18.28	16.91	0.00	0.00	4.72	49.09	0.03	0.23	0.00	0.00	0.00
SKK-2	7.85	30.34	0.00	0.00	7.91	0.33	0.61	2.71	0.00	0.00	0.00
SKK-3	0.94	2.15	4.19	0.00	1.95	1.16	0.29	8.38	0.00	0.00	0.00
SKK-5	0.08	30.47	15.07	0.00	18.30	0.23	67.48	9.63	0.00	0.00	0.00
SKK-7	34.06	43.59	7.29	0.00	8.91	8.64	0.02	5.21	0.00	0.00	0.00
SKK-11	15.11	4.16	19.18	8.96	5.58	0.26	0.50	4.22	0.00	0.00	0.00
SKK-25	3.16	2.41	2.17	0.00	3.49	1.29	0.00	2.01	1.07	6.65	3.12
SKK-26	11.33	9.90	17.46	0.00	3.78	1.27	0.00	7.63	9.96	6.97	6.15
SKK-28	1.59	2.08	10.45	0.00	8.98	0.61	0.71	13.13	7.44	3.89	11.25
SKK-31	7.47	16.77	4.68	0.00	2.52	3.29	0.64	10.86	2.24	11.93	12.33

PSO was used to generate the parameters of 10 suggested models and well-known classic models using LSE as an objective function in Table 4.11. given below.

Table 4.11. Parameters evaluated of proposed ten best models using PSO

Model s	Parameters										
	PURE S/W				H/W induced			User Induced			
	x	y	z	w	a	b	c	x1	x2	p	r
SKK-1	9768	0			10198	0	68	7218			
SKK-2	730.80	0			2020.4 0	0	5028.1 0	0			
SKK-3	1636.2 0	1323.9 0	2474.3 0		198.40	32.90	0	0			
SKK-5	18002	701	0		526	0	0	0			
SKK-7	1532	19504	0		526	0	0	0			
SKK-11	5850	165130	13060	519 0	200	0	0	10			

SKK-25	977	0	0		0	0		3812.0 0	22429	66976	32217
SKK-26	25250	1290	430		0	0		202260	0	50	0
SKK-28	383.10	162960 0	13590 0		136570 0	1893500. 00	172810 0	102200 0.	136760 0	102000 0	72650 0
SKK-31	7454	14226	0		22	424	0	4759	0	1079	346

4.5. PARAMETERS EVALUATION AND SOFT COMPUTING OPTIMIZATION USING DS#3

The undermentioned Tables 4.12. and 4.13. contains the parameter value of the candidate models and existing models evaluated using soft computing techniques in MATLAB 2022a. The constraints applied were that the lower bound is fixed to 0.1 and the upper bound to 1000 for SA and PSO. In the absence of these constraints, the value of the parameter became too big and we get, + inf or Nan values during Intensity function and comparison criteria evaluation.

Table 4.12. Parameters value of proposed models using LSE, GA, SA, and PSO

USING LSE											
Models	x	y	z	w	a	b	c	x1	x2	p	r
SKK-1	0.08	0.04	0.00	0.00	2.99	2.99	1.33	0.00	0.00	0.03	0.00
SKK-2	0.03	0.03	0.00	0.00	2.34	3.27	1.43	0.00	0.00	1.27	0.00
SKK-3	0.05	0.04	0.02	0.00	1.27	1.38	1.28	0.00	0.00	0.24	0.00
SKK-5	0.02	0.02	0.02	0.00	2.34	3.27	1.43	0.00	0.00	1.27	0.00
SKK-7	0.02	0.01	0.05	0.00	2.76	2.70	1.44	0.00	0.00	1.65	0.00
SKK-11	0.03	0.03	0.03	0.03	2.34	3.27	1.43	0.00	0.00	1.27	0.00
SKK-25	5.72	2.06	0.01	0.00	1.72	1.19	0.00	0.03	0.03	0.03	0.03
SKK-26	0.17	0.08	0.17	0.00	1.64	1.20	0.00	0.03	0.03	0.03	0.03
SKK-28	0.03	0.03	0.03	0.00	2.99	2.99	1.33	0.03	0.03	0.03	0.03
SKK-31	0.08	0.04	0.03	0.00	2.99	2.99	1.33	0.03	0.03	0.03	0.03
GA											
SKK-1	-24.42	0.54	0.00	0.00	-0.04	-52.50	1.39	0.00	0.00	22.04	0.00

SKK-2	77.05	12.91	0.00	0.00	-3.39	-1.93	1.43	0.00	0.00	3.81	0.00	
SKK-3	-68.30	7.25	-62.11	0.00	1.73	-0.59	6.74	0.00	0.00	-19.22	0.00	
SKK-5	24.89	3.71	-3.04	0.00	-1.45	-7.13	0.09	0.00	0.00	10.06	0.00	
SKK-7	16.98	1.90	3.64	0.00	-73.96	-0.42	3.56	0.00	0.00	4.48	0.00	
SKK-11	-46.63	-0.17	-1.19	1.97	8.58	2.06	1.30	0.00	0.00	-4.22	0.00	
SKK-25	233.89	-0.88	2.85	0.00	30.46	0.64	0.00	-89.73	10.10	-0.39	11.69	
SKK-26	-32.50	1.94	46.48	0.00	32.62	0.63	0.00	-	389.31	4.18	1.59	0.69
SKK-28	-82.51	21.99	16.45	0.00	-1.87	-2.31	0.77	175.62	2.50	57.67	0.10	
SKK-31	-66.84	1.07	6.46	0.00	2.67	5.51	1.87	13.31	5.73	0.84	-1.86	
SA												
SKK-1	0.10	1.76	0.00	0.00	0.62	11.77	1.54	0.00	0.00	0.57	0.00	
SKK-2	4.35	7.98	0.00	0.00	0.54	12.96	1.57	0.00	0.00	0.10	0.00	
SKK-3	6.46	0.10	0.50	0.00	6.59	0.62	1.28	0.00	0.00	0.10	0.00	
SKK-5	0.11	5.31	15.44	0.00	0.56	12.62	1.57	0.00	0.00	0.10	0.00	
SKK-7	16.89	17.49	1.70	0.00	4.50	2.16	1.33	0.00	0.00	0.10	0.00	
SKK-11	3.78	6.27	2.02	2.79	0.60	11.95	1.55	0.00	0.00	0.10	0.00	
SKK-25	0.37	0.10	14.54	0.00	14.57	0.77	0.00	10.87	17.78	15.10	26.58	
SKK-26	0.10	1.27	5.05	0.00	14.94	0.77	0.00	10.30	1.53	2.18	26.20	
SKK-28	10.62	4.65	7.41	0.00	0.21	28.19	1.84	8.33	30.83	4.22	18.76	
SKK-31	5.53	8.46	0.25	0.00	0.27	23.32	1.75	2.72	7.02	6.97	1.98	
PSO												
SKK-1	889.65	964.27	0.00	0.00	0.10	49.35	2.18	0.00	0.00	939.32	0.00	
SKK-2	850.19	874.02	0.00	0.00	0.10	48.74	2.19	0.00	0.00	0.10	0.00	
SKK-3	437.91	0.10	31.80	0.00	49.17	0.10	977.26	0.00	0.00	507.15	0.00	
SKK-5	1000.00	1000.00	135.40	0.00	0.10	207.00	0.10	0.00	0.00	996.70	0.00	
SKK-7	1.65	527.07	0.10	0.00	0.10	430.56	0.10	0.00	0.00	0.10	0.00	
SKK-11	0.62	0.88	1.00	0.98	0.00	0.00	0.93	0.00	0.00	0.00	0.00	
SKK-25	410.80	0.10	38.78	0.00	94.74	0.10	0.00	864.15	319.96	338.87	847.54	
SKK-26	421.40	0.10	38.78	0.00	94.74	0.10	0.00	78.99	592.31	520.81	989.88	
SKK-28	880.42	979.27	0.10	114.59	0.10	279.83	196.01	364.71	279.65	0.10	667.40	
SKK-31	437.55	0.10	32.15	0.00	992.13	5.48	72.26	519.48	997.93	977.95	149.69	

Table 4.13. Parameters value of existing models using LSE, GA, SA, and PSO

Using LSE						
Models	a	b	c	w	p	r
Duane	1.6392	1.1967	0.00	0.00	0.00	0.00
GO	225.0253	0.1	0.00	0.00	0.00	0.00
MO	44.3025	0.4845	0.00	0.00	0.00	0.00
YDM	303.7175	8.6539	0.00	0.00	0.00	0.00
GG	1196	0.1	1	0.00	0.00	0.00
GMPZ	4.57	0.1	20.7638	0.00	0.00	0.00
MD	2.9376	3.0351	1.3259	0.00	0.00	0.00
YIM	647.3118	0.1102	117.9695	0.00	0.00	0.00
LGC	115.7953	0.12	78.58	0.00	0.00	0.00
YMID1	1	1	1	0.00	0.00	0.00
YMID2	1.0017	0.9991	1.0067	0.00	0.00	0.00
PZIFD	3.5727	0.6311	2.8181	0.00	0.00	0.00
YR	13.1409	0.1	2.1355	2.1355	0.00	0.00
YE	223.1181	0.1	0.5617	0.5617	0.00	0.00
PNZ	0.1	7.1172	0.1	0.1	0.00	0.00
PZ	0.9766	1.5907	0.9766	0.5938	0.1	0.00
ZTP	0.2811	0.1063	0.3034	0.1	0.137	0.2348
Using GA						
Models	a	b	c	w	p	r
Duane	18.6497	0.719	0.00	0.00	0.00	0.00
GO	407.397	0.0251	0.00	0.00	0.00	0.00
MO	252.6591	0.0563	0.00	0.00	0.00	0.00
YDM	521.2111	0.0506	0.00	0.00	0.00	0.00
GG	297.2607	0.3129	86.5723	0.00	0.00	0.00
GMPZ	4.0941	41.6711	1.2004	0.00	0.00	0.00
MD	1.1259	6.6853	1.3713	0.00	0.00	0.00
YIM	379.6384	0.1248	103.3072	0.00	0.00	0.00
LGC	461.7704	0.10	27.86	0.00	0.00	0.00
YMID1	299.2651	5.3338	0.0081	0.00	0.00	0.00
YMID2	4.0941	41.6711	1.2004	0.00	0.00	0.00
PZIFD	51.4282	0.5704	0.4132	0.00	0.00	0.00
YR	338.9798	0.0012	7.2419	0.4322	0.00	0.00

YE	103.058	0.0573	0.7897	0.4379	0.00	0.00
PNZ	94.6938	0.2773	0.0704	231.6193	0.00	0.00
PZ	12.2833	0.0292	4.2764	9.1458	58.2949	0.00
ZTP	5.503	4.1216	-13.2735	-28.3087	18.2626	19.3442
Using SA						
Models	a	b	c	w	p	r
Duane	50.3804	0.5018	0.00	0.00	0.00	0.00
GO	0.1	0.1	0.00	0.00	0.00	0.00
MO	174.8743	0.1	0.00	0.00	0.00	0.00
YDM	2.2205	0.1	0.00	0.00	0.00	0.00
GG	25.6764	0.3261	4.3896	0.00	0.00	0.00
GMPZ	59.1202	5.9122	65.5597	0.00	0.00	0.00
MD	0.1	49.3404	2.18	0.00	0.00	0.00
YIM	439.4487	0.1	31.9303	0.00	0.00	0.00
LGC	7.4454	0.48	0.48	0.00	0.00	0.00
YMID1	0.1	0.1	0.1	0.00	0.00	0.00
YMID2	4.64	0.8304	0.808	0.00	0.00	0.00
PZIFD	1	1	1	0.00	0.00	0.00
YR	20.8451	0.1237	7.23	9.2987	0.00	0.00
YE	66.7955	0.1	0.1	6.3686	0.00	0.00
PNZ	83.2186	0.1781	0.1	63.4938	0.00	0.00
PZ	2.4062	0.1	38.1124	0.209	0.1	0.00
ZTP	13.2595	20.3141	0.5808	7.394	58.4827	58.4826
Using PSO						
models	a	b	c	w	p	r
Duane	225.0253	0.1	0.00	0.00	0.00	0.00
GO	150.4174	0.1	0.00	0.00	0.00	0.00
MO	174.8729	0.1	0.00	0.00	0.00	0.00
GG	2307.8	0.1	0.1	0.00	0.00	0.00
GMPZ	3229.5	11	665.3	0.00	0.00	0.00
MD	0.1	0.1	0.1	0.00	0.00	0.00
YDM	310.1484	0.1		0.00	0.00	0.00
YIM	437.9114	0.1	31.8047	0.00	0.00	0.00
LGC	451.6789	0.10	31.80	0.00	0.00	0.00
YMID1	0.1	0.1	0.1	0.00	0.00	0.00

YMID2	49	3100	0.1	0.00	0.00	0.00
PZIFD	153.69	0.1612	0.1	0.00	0.00	0.00
YE	3121.8	0.1	0.1	0.1	0.00	0.00
YR	5590.8	0.1	0.1	0.1	0.00	0.00
PNZ	285.7985	11.1987	10.3021	101.2182	0.00	0.00
PZ	0.1	0.1	0.1	243.9541	0.1	0.00
ZTP	0.1	1965.4	49	46	0.1	0.1

4.6. PARAMETERS EVALUATION AND SOFT COMPUTING OPTIMIZATION USING DS#4

The undermentioned tables 4.14. and 4.15. contains the value of the model's, (developed candidates and existing well-known) parameters evaluated using LSE, GA, SA, and PSO in MATLAB 2022a. The constraints applied were that the lower bound is fixed to 0.1 and the upper bound to 1000 for SA and PSO. In the absence of these constraints, the parameter's value became too big and we get, + inf or Nan values during Intensity function and comparison criteria evaluation.

Table 4.14. Parameters value of proposed models using LSE, GA, SA, and PSO

Using LSE											
Models	x	y	z	w	a	b	c	x1	x2	p	r
SKK-1	0.08	0.06	0.00	0.00	1.68	1.52	1.10	0.00	0.00	0.03	0.00
SKK-2	0.03	0.03	0.00	0.00	1.25	0.84	1.54	0.00	0.00	1.86	0.00
SKK-3	0.06	0.05	0.02	0.00	1.07	0.97	1.66	0.00	0.00	0.15	0.00
SKK-5	0.03	0.03	0.03	0.00	1.25	0.84	1.54	0.00	0.00	1.86	0.00
SKK-7	0.03	0.03	0.05	0.00	1.28	0.97	1.14	0.00	0.00	1.27	0.00
SKK-11	0.03	0.03	0.03	0.03	1.25	0.84	1.54	0.00	0.00	1.86	0.00
SKK-25	2.31	1.02	0.00	0.00	0.03	0.83	1.21	0.03	0.03	0.03	0.03
SKK-26	0.08	0.05	0.00	0.00	0.08	0.96	1.18	0.03	0.03	0.03	0.03
SKK-28	0.03	0.03	0.03	0.00	1.68	1.52	1.10	0.03	0.03	0.03	0.03
SKK-31	0.08	0.04	0.03	0.00	2.99	2.99	1.33	0.03	0.03	0.03	0.03

Using GA											
Models	x	y	z	w	a	b	c	x1	x2	p	r
SKK-1	133.64	46.31	0.00	0.00	-2.60	-1.72	1.29	0.00	0.00	18.80	0.00
SKK-2	-6.85	62.85	0.00	0.00	-4.33	-0.97	1.29	0.00	0.00	0.15	0.00
SKK-3	-29.44	-1.76	1.08	0.00	-0.14	-13.73	0.31	0.00	0.00	-30.13	0.00
SKK-5	-12.23	35.41	-3.12	0.00	0.94	-3.55	1.46	0.00	0.00	2.17	0.00
SKK-7	-9.73	1.54	-8.10	0.00	57.42	0.20	2.60	0.00	0.00	2.46	0.00
SKK-11	-8.09	2.59	-2.70	0.49	42.73	-0.23	1.42	0.00	0.00	2.47	0.00
SKK-25	-24.85	6.61	0.00	0.00	-27.20	2.02	1.01	-8.27	0.16	2.05	1.73
SKK-26	103.05	-2.13	0.00	0.00	-5.86	34.23	0.42	48.54	0.43	-0.46	1.61
SKK-28	61.71	12.43	0.31	0.00	1.25	46.83	0.02	-49.78	1.54	2.20	3.78
SKK-31	-56.77	12.90	-29.91	0.00	28.19	-0.48	8.48	232.88	27.50	-0.05	8.04
Using SA											
Models	x	y	z	w	a	b	c	x1	x2	p	r
SKK-1	10.77	1.93	0.00	0.00	4.77	0.45	1.01	0.00	0.00	4.46	0.00
SKK-2	5.21	5.29	0.00	0.00	0.10	14.88	1.07	0.00	0.00	0.59	0.00
SKK-3	11.99	6.06	3.40	0.00	0.11	0.81	2.64	0.00	0.00	12.14	0.00
SKK-5	2.19	3.57	17.81	0.00	1.58	0.60	16.57	0.00	0.00	2.08	0.00
SKK-7	19.71	59.40	26.86	0.00	39.86	8.89	84.93	0.00	0.00	2.08	0.00
SKK-11	18.20	11.99	1.13	6.70	26.68	3.62	27.75	0.00	0.00	2.08	0.00
SKK-25	28.44	41.74	0.00	0.00	31.17	3.55	0.88	2.33	12.63	94.63	28.91
SKK-26	8.61	4.39	0.00	0.00	3.64	2.81	0.93	3.27	3.50	0.55	0.82
SKK-28	6.68	1.31	0.84	0.00	0.25	8.58	1.05	11.79	0.26	20.00	17.44
SKK-31	16.84	33.65	2.75	0.00	0.11	46.59	2.13	39.90	15.20	12.95	1.97
Using PSO											
Models	x	y	z	w	a	b	c	x1	x2	p	r
SKK-1	69.82	126.88	0.00	0.00	0.10	186.40	0.10	0.00	0.00	84.55	0.00
SKK-2	8.98	0.10	0.00	0.00	0.10	0.10	0.10	0.00	0.00	0.10	0.00
SKK-3	755.83	14.45	707.63	0.00	0.10	0.10	0.10	0.00	0.00	239.73	0.00
SKK-5	0.10	791.82	0.10	0.00	0.10	0.10	61.21	0.00	0.00	2.24	0.00
SKK-7	784.40	1000.00	1000.00	0.00	210.60	0.10	20.40	0.00	0.00	2.10	0.00
SKK-11	1000.00	999.40	0.10	18.10	1.30	176.40	0.10	0.00	0.00	0.10	0.00
SKK-25	103.62	0.10	0.00	0.00	39.84	52.62	0.10	165.04	772.05	383.90	15.22
SKK-26	106.21	0.10	0.00	0.00	39.83	52.62	0.10	408.06	196.34	635.39	187.18
SKK-28	444.16	-584.64	-797.34	0.00	-622.73	2.60	173.25	924.60	498.52	-33.07	800.17

SKK-31	712.20	127.10	-324.70	0.00	1436.90	152.90	0.00	-943.00	-681.10	-1519.90	103.70
---------------	--------	--------	---------	------	---------	--------	------	---------	---------	----------	--------

Table 4.15. Parameters value of existing models using LSE, GA, SA, and PSO

Using LSE						
Models	a	b	c	w	p	r
Duane	0.7639	1.2823	0.00	0.00	0.00	0.00
GO	49.4449	0.1	0.00	0.00	0.00	0.00
MO	26.9533	0.207	0.00	0.00	0.00	0.00
YDM	98.6311	0.1	0.00	0.00	0.00	0.00
GG	10.8446	1.4973	1	0.00	0.00	0.00
GMPZ	0.8479	1.3636	138.1299	0.00	0.00	0.00
MD	1.6151	1.5963	1.1025	0.00	0.00	0.00
YIM	22.7848	0.1084	19.8891	0.00	0.00	0.00
LGC	6.523	0.16	3.60	0.00	0.00	0.00
YMID1	1	1	1	0.00	0.00	0.00
YMID2	1.0001	1	1.0002	0.00	0.00	0.00
PZIFD	6.3431	4.0536	3.4572	0.00	0.00	0.00
YR	18.054	0.2401	2.6265	2.6265	0.00	0.00
YE	16.246	0.1156	2.748	2.748	0.00	0.00
PNZ	1	33.6963	0.1	1	0.00	0.00
PZ	1	1	1	1	1.0012	0.00
ZTP	1.0004	1	1.0004	1	0.9991	1.0101
Using GA						
Models	a	b	c	w	p	r
Duane	1.9374	1.0287	0.00	0.00	0.00	0.00
GO	207.4256	0.0155	0.00	0.00	0.00	0.00
MO	39.7592	0.1598	0.00	0.00	0.00	0.00
YDM	141.5727	0.0547	0.00	0.00	0.00	0.00
GG	133.3192	0.2818	78.1872	0.00	0.00	0.00
GMPZ	18.9848	2.0933	-10.5472	0.00	0.00	0.00
MD	-3.187	-1.5424	1.3645	0.00	0.00	0.00
YIM	111.4086	0.1137	44.677	0.00	0.00	0.00

LGC	-31.2966	4.95	-6.02	0.00	0.00	0.00
YMID1	-36.6345	20.2261	-0.1402	0.00	0.00	0.00
YMID2	-8.8121	41.2357	-0.2423	0.00	0.00	0.00
PZIFD	6.5884	4.6282	4.6326	0.00	0.00	0.00
YR	-18.0379	20.952	-2.2079	6.409	0.00	0.00
YE	-4.21	9.2598	6.4101	-0.8827	0.00	0.00
PNZ	-28.4724	-2.1161	8.769	61.1013	0.00	0.00
PZ	-21.5471	8.8904	18.1951	-1.0723	0.92	0.00
ZTP	-12.6531	3.5602	-9.7462	-15.8466	5.7515	7.3925
Using SA						
Models	a	b	c	w	p	r
Duane	19.9172	0.4421	0.00	0.00	0.00	0.00
GO	49.7598	0.1	0.00	0.00	0.00	0.00
MO	53.6351	0.1	0.00	0.00	0.00	0.00
YDM	242.6952	7.8108	0.00	0.00	0.00	0.00
GG	152.1458	0.2599	33.5759	0.00	0.00	0.00
GMPZ	402.6484	7.8843	149.7526	0.00	0.00	0.00
MD	43.6476	0.1	1.157	0.00	0.00	0.00
YIM	122.7276	0.1	24.9035	0.00	0.00	0.00
LGC	121.8336	0.10	22.30	0.00	0.00	0.00
YMID1	0.1237	0.1	0.1	0.00	0.00	0.00
YMID2	20.904	2.903	0.1	0.00	0.00	0.00
PZIFD	109.788	9.3411	6.9634	0.00	0.00	0.00
YR	20.04	0.1	2.1455	9.8199	0.00	0.00
YE	0.1	0.1	0.1	0.1	0.00	0.00
PNZ	6.5613	15.1291	2.745	21.2942	0.00	0.00
PZ	0.1	9.8527	0.1	0.1002	4.3421	0.00
ZTP	18.004	12.3468	0.4364	26.4662	0.1	0.1
Using PSO						
Models	a	b	c	w	p	r
Duane	81.1259	0.1	0.00	0.00	0.00	0.00
GO	49.7088	0.1	0.00	0.00	0.00	0.00
MO	53.4543	0.1	0.00	0.00	0.00	0.00
YDM	88.9666	0.1	0.00	0.00	0.00	0.00
GG	893.6732	0.1	241.316	0.00	0.00	0.00

GMPZ	0.1	0.1	177.0499	0.00	0.00	0.00
MD	0.1	0.1	0.1	0.00	0.00	0.00
YIM	122.9165	0.1	24.7396	0.00	0.00	0.00
LGC	118.1409	0.10	24.74	0.00	0.00	0.00
YMID1	0.1	0.1	0.1	0.00	0.00	0.00
YMID2	0.1	0.1	0.1	0.00	0.00	0.00
PZIFD	41.0329	0.1904	0.1	0.00	0.00	0.00
YR	895.577	0.1	0.1	0.1	0.00	0.00
YE	7819.3	0.1	0.1	983.4	0.00	0.00
PNZ	564.1	1833.1	900.1	426.9	0.00	0.00
PZ	0.1	0.1	0.1	106.6234	1.2707	0.00
ZTP	188.1	0.1	0.1	3.9	2938.7	2939.1

4.7. SELECTION OF BEST MODEL FROM TEN CANDIDATE MODELS USING DS#1

The following Table 4.16. shows the error deviation of various candidate models.

Table 4.16. The deviation between observed and estimated values

SKK-1	SKK-2	SKK-3	SKK-5	SKK-7	SKK-11	SKK-25	SKK-26	SKK-28	SKK-31
74214	73623	157973	79394	40031	51433	1	2	2	1
43449	42947	112134	47196	19993	27001	1	2	0	1
25491	25106	79671	28093	10017	14216	5	5	2	1
14993	14713	56667	16797	5041	7511	2	2	3	3
8830	8635	40336	9991	2538	3974	8	8	3	2
5217	5083	28746	6015	1286	2112	4	4	2	3
3088	2998	20505	3616	653	1125	3	3	4	4
1831	1772	14641	2177	333	601	2	2	5	6
1082	1043	10457	1311	164	315	8	8	0	1
645	620	7482	790	84	169	4	4	4	5
374	358	5347	468	33	80	14	13	4	3
203	193	3814	264	4	22	29	28	18	18

135	129	2753	174	9	24	4	4	7	8
78	73	1974	103	2	9	6	6	6	7
41	39	1414	58	4	1	9	9	3	4
30	28	1023	40	2	4	1	1	12	13
27	28	692	20	43	42	45	45	32	31
3	3	517	2	12	13	14	14	0	1
2	3	374	1	7	8	9	9	5	6
2	2	271	0	5	6	6	6	9	10
6	7	191	5	8	9	9	9	6	7
1	1	142	0	2	3	3	3	13	14
4	4	100	3	4	5	5	5	11	12
22	22	53	22	22	23	23	23	6	5
3	3	52	2	2	3	3	3	14	15
5	5	35	4	4	5	5	5	13	14
8	8	21	7	7	8	8	8	10	11
14	14	7	14	13	14	14	14	4	6
22	22	7	22	21	22	22	22	3	2
5	5	6	5	4	5	5	5	14	16

After calculating deviations, we need to determine which model is giving the best approximate prediction by utilizing the weighted estimation function. The undermentioned Table 4.17. represents the number of different error deviation values from each E_i .

Table 4.17. Weighted estimation function

Error	SKK-	SKK-	SKK-	SKK-	SKK-	SKK-	SKK-	SKK-	SKK-	SKK-
	1	2	3	5	7	11	25	26	28	31
E_5	2	2	0	2	1	3	4	4	2	2
E_4	1	1	0	1	5	1	3	3	4	2
E_3	2	3	0	1	0	2	3	3	4	3
E_2	2	1	0	2	4	0	2	4	3	2

E_1	1	1	0	1	0	1	3	1	0	5
E_0	0	0	0	2	0	0	0	0	3	0
W.Fun	32	30	0	48	35	21	58	54	74	70
Ranking	7	8	10	5	6	9	3	4	1	2

It is clear from Table 4.17. that model SKK-28 gives the best approximate estimation out of all the ten models followed by SKK-31, SKK-25, SKK-26, SKK-5, and so on. SKK-3 gives the worst performance with no deviation till the 5-point dimension. Also, SKK-28 gives the maximum number of exact matches between observed and predicted values. SKK-31 has a maximum number of 1 point deviation and SKK-26 & 4 gives the maximum number of 2-point error deviation, SKK-28 has both 3 - points and 4 - points maximum deviation point and SKK-25 and SKK-26 both give a maximum number of 5 - point error difference.

4.8. RANKING CANDIDATE MODELS USING OPTIMIZED PARAMETERS VALUES FOR DS#3

We evaluated the rank of ten candidate models using DS#3 to determine the best model when the fault data used is not related to big data. The undermentioned tables from Table 4.18 to Table 4.22 show the weighted rank vector used and the weighted rank matrix evaluated during the procedure using dataset-3. The comparison criteria set used to contain four measures MSE, RMSE, RAE, and, RRSE.

Table 4.18. Weight matrix corresponding to criteria measures rank

Rank	1	2	3	4	5	6	7	8	9	10
Weight	0.0001	0.002	0.003	0.01	0.05	0.1	1	4	12	40

Table 4.19. WRM using LSE

LSE	R_MS E	R_RMS E	R_RA E	R_RRS E
SKK-1	3	3	3	3
SKK-2	4	4	5	4
SKK-3	10	10	10	10
SKK-5	5	5	6	5
SKK-7	6	6	7	6
SKK-11	7	7	4	7
SKK-25	8	8	9	8
SKK-26	9	9	8	9
SKK-28	2	2	1	2
SKK-31	1	1	2	1

Table 4.20. WRM using GA

GA	R_MSE	R_RMSE	R_RAE	R_RRSE
SKK-1	8	8	9	8
SKK-2	1	2	5	2
SKK-3	10	10	10	10
SKK-5	4	4	1	4
SKK-7	2	1	7	1
SKK-11	7	7	6	7
SKK-25	5	5	4	5
SKK-26	9	9	2	9
SKK-28	3	3	8	3
SKK-31	6	6	3	6

Table 4.21. WRM using SA

LSE	R_MSE	R_RMSE	R_RAE	R_RRSE
SKK-1	5	5	5	5
SKK-2	3	3	3	3
SKK-3	10	10	10	10
SKK-5	4	4	4	4
SKK-7	6	6	6	6
SKK-11	9	9	7	9
SKK-25	7	7	9	7
SKK-26	8	8	8	8
SKK-28	2	2	1	2
SKK-31	1	1	2	1

Table 4.22. WRM using PSO

GA	R_MSE	R_RMSE	R_RAE	R_RRSE
SKK-1	8	8	7	8
SKK-2	3	3	4	3
SKK-3	7	7	6	7
SKK-5	9	9	8	9
SKK-7	4	4	5	4
SKK-11	5	5	9	5
SKK-25	1	1	2	1
SKK-26	6	6	3	6
SKK-28	10	10	10	10
SKK-31	2	2	1	2

Table 4.23 to Table 4.26 evaluate the rank f candidate models when parameters were evaluated using LSE and GA, SA, and PSO optimization methods.

Table 4.23. Rank matrix using LSE

MODELS	LSE(WR)
SKK-1	3
SKK-2	4
SKK-3	10
SKK-5	5
SKK-7	6
SKK-11	7
SKK-25	8
SKK-26	9
SKK-28	2
SKK-31	1

Table 4.24. Rank matrix using GA

MODELS	GA(WR)
SKK-1	8
SKK-2	3
SKK-3	10
SKK-5	2
SKK-7	6
SKK-11	7
SKK-25	4
SKK-26	9
SKK-28	1
SKK-31	5

Table 4.25. Rank matrix using SA

MODELS	SA.WR
SKK-1	5
SKK-2	3
SKK-3	10
SKK-5	4
SKK-7	6
SKK-11	9
SKK-25	7
SKK-26	8
SKK-28	2
SKK-31	1

Table 4.26. Rank matrix using PSO

MODELS	PSO.WR
SKK-1	8
SKK-2	6
SKK-3	5
SKK-5	9
SKK-7	3
SKK-11	7
SKK-25	1
SKK-26	4
SKK-28	10
SKK-31	2

Combining all method's ranks and taking the average we get Table 4.27 as shown below.

Table 4.27. Average optimized rank determination of candidate models

MODELS	LSE	GA	SA	PSO	Avg. RANK
SKK-1	3	8	5	8	7
SKK-2	4	3	3	6	3
SKK-3	10	10	10	5	10
SKK-5	5	2	4	9	4
SKK-7	6	6	6	3	6
SKK-11	7	7	9	7	8
SKK-25	8	4	7	1	5
SKK-26	9	9	8	4	9
SKK-28	2	1	2	10	2
SKK-31	1	5	1	2	1

4.9. RANKING CANDIDATE MODELS USING OPTIMIZED PARAMETERS VALUES FOR DS#4

Utilizing DS#4 to determine the individual ranking corresponding to each evaluation method used. Table 4.28 to Table 4.31. corresponds to the weighted rank matrix (WRM) using various methods. Table 4.32. to Table 4.35. shows the rank of candidate models for various techniques and Table 4.36. aggregate the ranks.

Table 4.28. WRM using LSE

LSE	R_MSE	R_RMSE	R_RAE	R_RRSE
SKK-1	0.01	0.01	0.003	0.01
SKK-2	0.0001	0.0001	0.01	0.0001
SKK-3	0.1	0.1	0.1	0.1
SKK-5	0.002	0.002	0.05	0.002
SKK-7	0.05	0.05	8	0.05
SKK-11	1	1	1	1
SKK-25	12	12	12	12
SKK-26	8	8	40	8
SKK-28	0.003	0.003	0.002	0.003
SKK-31	40	40	0.0001	40

Table 4.29. WRM using GA

GA	R_MSE	R_RMSE	R_RAE	R_RRSE
SKK-1	1	1	12	1
SKK-2	0.01	0.01	0.1	0.01
SKK-3	4	4	40	4
SKK-5	0.002	0.0001	0.01	0.0001
SKK-7	0.003	0.002	0.05	0.002
SKK-11	0.05	0.1	1	0.1
SKK-25	12	12	0.003	12
SKK-26	40	40	0.0001	40
SKK-28	0.1	0.05	4	0.05
SKK-31	0.0001	0.003	0.002	0.003

Table 4.30. WRM using LSE

LSE	R_MSE	R_RMSE	R_RAE	R_RRSE
SKK-1	0.05	0.05	12	0.05
SKK-2	0.01	0.01	1	0.01
SKK-3	4	4	40	4
SKK-5	0.0001	0.0001	0.01	0.0001
SKK-7	0.002	0.002	0.05	0.002
SKK-11	1	1	4	1
SKK-25	40	40	0.0001	40
SKK-26	12	12	0.002	12
SKK-28	0.003	0.003	0.1	0.003
SKK-31	0.1	0.1	0.003	0.1

Table 4.31. WRM using GA

GA	R_MSE	R_RMSE	R_RAE	R_RRSE
SKK-1	0.1	0.1	1	0.1
SKK-2	0.003	0.003	0.1	0.003
SKK-3	1	1	4	1
SKK-5	0.002	0.002	40	0.002
SKK-7	0.0001	0.0001	12	0.0001
SKK-11	0.05	0.05	0.05	0.05
SKK-25	4	4	0.003	4
SKK-26	12	12	0.002	12
SKK-28	40	40	0.0001	40
SKK-31	0.01	0.01	0.01	0.01

Table 4.32. Rank matrix using LSE

MODEL	LSE (WR)
SKK-1	5
SKK-2	4
SKK-3	10
SKK-5	7
SKK-7	6
SKK-11	8
SKK-25	9
SKK-26	2
SKK-28	1
SKK-31	3

Table 4.33. Rank matrix using GA

MODEL	GA (WR)
SKK-1	9
SKK-2	2
SKK-3	1
SKK-5	3
SKK-7	5
SKK-11	8
SKK-25	10
SKK-26	6
SKK-28	4
SKK-31	7

Table 4.34. Rank matrix using LSE

MODELS	SA (WR)
SKK-1	7
SKK-2	5
SKK-3	9
SKK-5	1
SKK-7	2
SKK-11	6
SKK-25	10
SKK-26	8
SKK-28	3
SKK-31	4

Table 4.35. Rank matrix using GA

MODELS	PSO (WR)
SKK-1	4
SKK-2	2
SKK-3	5
SKK-5	9
SKK-7	6
SKK-11	3
SKK-25	7
SKK-26	8
SKK-28	10
SKK-31	1

Combining all method's ranks and taking the average we get Table 4.36 as shown below.

Table 4.36. Average optimized rank determination of candidate models

MODELS	LSE	GA	SA	PSO	Avg. RANK
SKK-1	5	9	7	4	7
SKK-2	4	2	5	2	1
SKK-3	10	1	9	5	8
SKK-5	7	3	1	9	5
SKK-7	6	5	2	6	4
SKK-11	8	8	6	3	9
SKK-25	9	10	10	7	10
SKK-26	2	6	8	8	6
SKK-28	1	4	3	10	3
SKK-31	3	7	4	1	2

CONCLUSION

Fault data from the software product's testing phase predicts or estimates future defects, determining its dependability. Most models were developed using data available in published research papers. The non-availability of new data sets regarding new technologies makes it difficult for accurate parameter estimation and validation of the model. Parameters were evaluated of developed 33 hybrid models as well-known traditional models using MLE and LSE statistical techniques. Since these proposed models were nonlinear in nature so soft computing methods like GA, PSO, and SA were used to optimize the result. According to comparison criteria ranks SKK-26 model

performs better when parameters were evaluated using the LSE method. SKK-1 gives better performance in the case of GA evaluation, SKK-3 when parameters were evaluated using the PSO method and SKK-25 performs better than others when SA was employed.

Software is released only when it reached a defined level of reliability which signifies software quality within specified limits. Various SRGM are in existence and one can either choose from them or develop a model based on the environment, requirements, and techniques used. While developing models it is a common trend to use develop multiple models first and then choose the best among them. We developed 33 models and then narrow it down to ten models based on parameters and estimation value. SKK-28 is selected as the best model out of candidate ten based on the estimation function. This best-suited model is then required to be validated by comparing it with other models based on certain criteria to obtain a good fit model. The proposed ranking methodology shows a good result when checked against DS#2 and ranks the models taking into account their estimation capability apart from individual rank in the criteria set.

CHAPTER 5

RESULTS AND DISCUSSION

In this chapter, we utilized various methodologies for model validation. The suggested model is compared to others to establish it is the best estimator for massive fault data. We determine the correlation between actual and predicted values using a t-test and show that they are drawn from the same population pool. Various graphs were analysed to confirm our claim of a better-performing model and methodology.

5.1. MODEL VALIDATION

Validation determines the model's estimate capabilities for which we used three methods. First, the estimation values of all models are calculated to have an idea regarding the best estimator. We also utilized the criteria set values to analytically determine the model's capability. The cumulative mean value and Cumulative error rate are plotted to have an idea regarding the model's validity and enhanced performance. Finally, a t-test shows that actual and predicted values are from the same pool.

5.1.1. Failure Estimation

The error rate of the proposed models and selected traditional models are shown in Table 5.1. For each time period, all current models predict a value close to zero. In comparison to the GG, GO, GMPZ, YMD, YMI, GD-1, GD-2, and GD-3 models, the SKK-28 model's predicted values provide the most precise approximation. According to our data analysis, we can state that the developed hybrid model has the greatest accuracy rate for the provided DS#1 dataset.

Table 5.1. Intensity function (estimation value) calculation of models

t(Days)	O(i)	SKK-28	GMPZ	GG	GO	YDS	YDI	GD1	GD2	GD3	YME
1.00	6.00	4.00	0.00	0.03	0.14	0.00	0.05	0.41	0.31	0.36	0.17
2.00	4.00	4.00	0.00	0.02	0.14	0.01	0.10	0.26	0.21	0.23	0.17
3.00	7.00	5.00	0.00	0.01	0.14	0.01	0.14	0.21	0.17	0.19	0.17
4.00	3.00	6.00	0.00	0.01	0.14	0.02	0.19	0.19	0.15	0.17	0.17
5.00	9.00	6.00	0.00	0.01	0.14	0.02	0.24	0.17	0.14	0.15	0.17
6.00	5.00	7.00	0.00	0.00	0.14	0.03	0.29	0.16	0.13	0.14	0.17
7.00	4.00	8.00	0.00	0.00	0.14	0.03	0.34	0.15	0.13	0.14	0.17
8.00	3.00	8.00	0.00	0.00	0.14	0.03	0.39	0.15	0.13	0.13	0.17
9.00	9.00	9.00	0.00	0.00	0.14	0.04	0.44	0.14	0.12	0.13	0.17
10.00	5.00	9.00	0.00	0.00	0.14	0.04	0.49	0.14	0.12	0.13	0.17
11.00	14.00	10.00	0.00	0.00	0.13	0.04	0.55	0.13	0.12	0.12	0.17
12.00	29.00	10.00	0.00	0.00	0.13	0.05	0.60	0.13	0.12	0.12	0.17
13.00	4.00	10.00	0.00	0.00	0.13	0.05	0.65	0.13	0.12	0.12	0.17
14.00	6.00	10.00	0.00	0.00	0.13	0.05	0.71	0.13	0.11	0.12	0.17
15.00	9.00	10.00	0.00	0.00	0.13	0.06	0.77	0.12	0.11	0.12	0.17
16.00	1.00	10.00	0.00	0.00	0.13	0.06	0.82	0.12	0.11	0.12	0.17
17.00	45.00	10.00	0.00	0.00	0.13	0.06	0.88	0.12	0.11	0.11	0.16
18.00	14.00	10.00	0.00	0.00	0.13	0.07	0.94	0.12	0.11	0.11	0.16
19.00	9.00	10.00	0.00	0.00	0.13	0.07	1.00	0.12	0.11	0.11	0.16
20.00	6.00	10.00	0.00	0.00	0.13	0.07	1.06	0.12	0.11	0.11	0.16
21.00	9.00	20.00	0.00	0.00	0.13	0.07	1.13	0.12	0.11	0.11	0.16
22.00	3.00	20.00	0.00	0.00	0.13	0.08	1.19	0.12	0.11	0.11	0.16
23.00	5.00	20.00	0.00	0.00	0.13	0.08	1.25	0.12	0.11	0.11	0.16
24.00	23.00	20.00	0.00	0.00	0.13	0.08	1.32	0.11	0.11	0.11	0.16
25.00	3.00	20.00	0.00	0.00	0.13	0.09	1.38	0.11	0.11	0.11	0.16
26.00	5.00	20.00	0.00	0.00	0.13	0.09	1.45	0.11	0.11	0.11	0.16
27.00	8.00	20.00	0.00	0.00	0.13	0.09	1.52	0.11	0.11	0.11	0.16
28.00	14.00	20.00	0.00	0.00	0.13	0.09	1.59	0.11	0.11	0.11	0.16
29.00	22.00	20.00	0.00	0.00	0.13	0.09	1.66	0.11	0.11	0.11	0.16
30.00	5.00	19.00	0.00	0.00	0.13	0.10	1.73	0.11	0.11	0.11	0.16

5.1.2. Graphical Representation

We calculated the hazard rate function to determine the estimated and accumulated number of faults. In figure 5.1 we plot the observed and estimated values of SKK-28 with other models. Figure 5.2 represents the cumulative plotting of the SKK-28 model's estimated values and observed values.

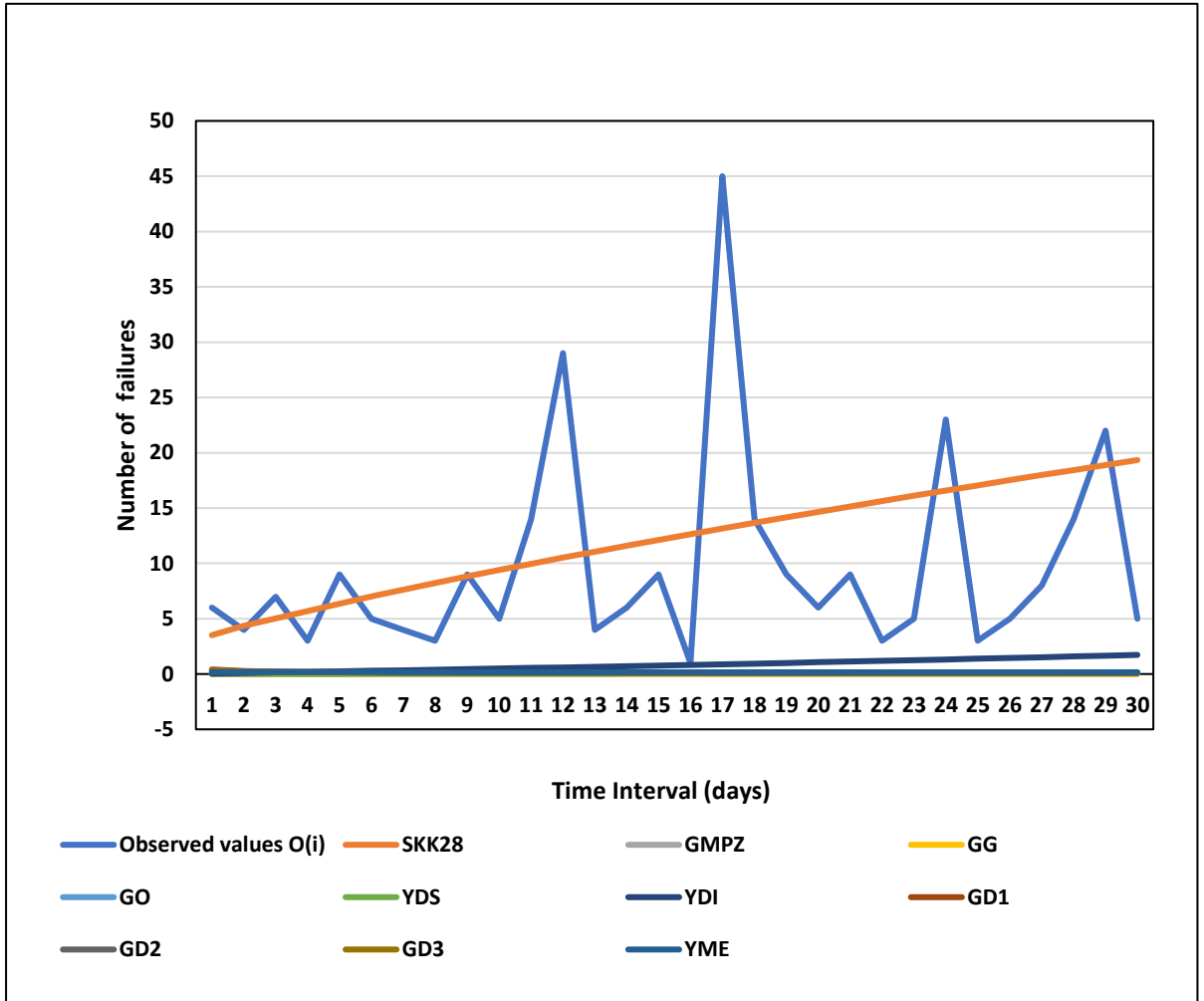


Figure 5.1. Observed values and estimation values comparison between models

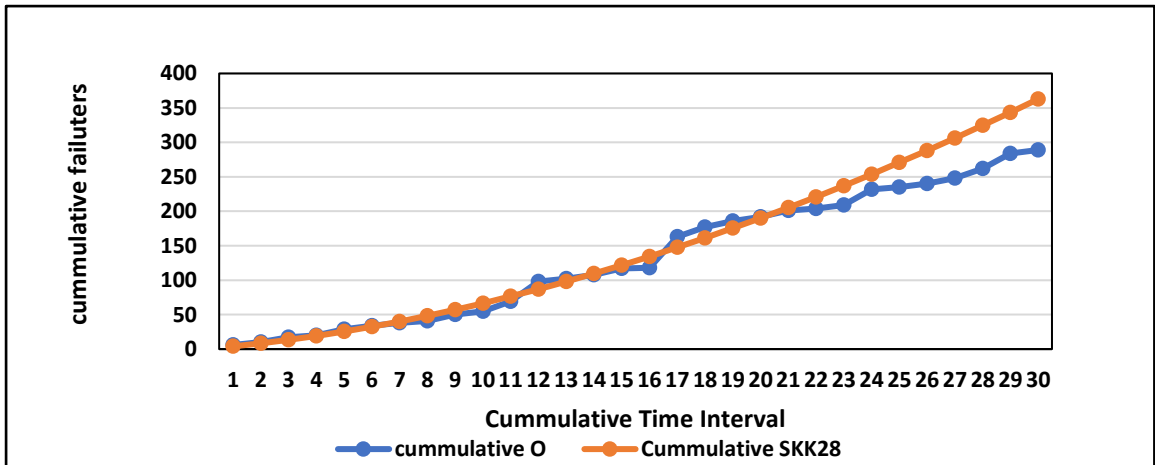


Figure 5.2. Cumulative experimental and estimation value of model SKK-28

5.1.3. Comparison Criteria

A comparison criterion having thirteen measures is used to determine the SKK-28 model’s ability to predict fault in comparison with other models.

- BIAS: A model must have a low value of bias for better prediction. The plot of bias for SKK-28 and other existing models is given below in Figure 5.3.

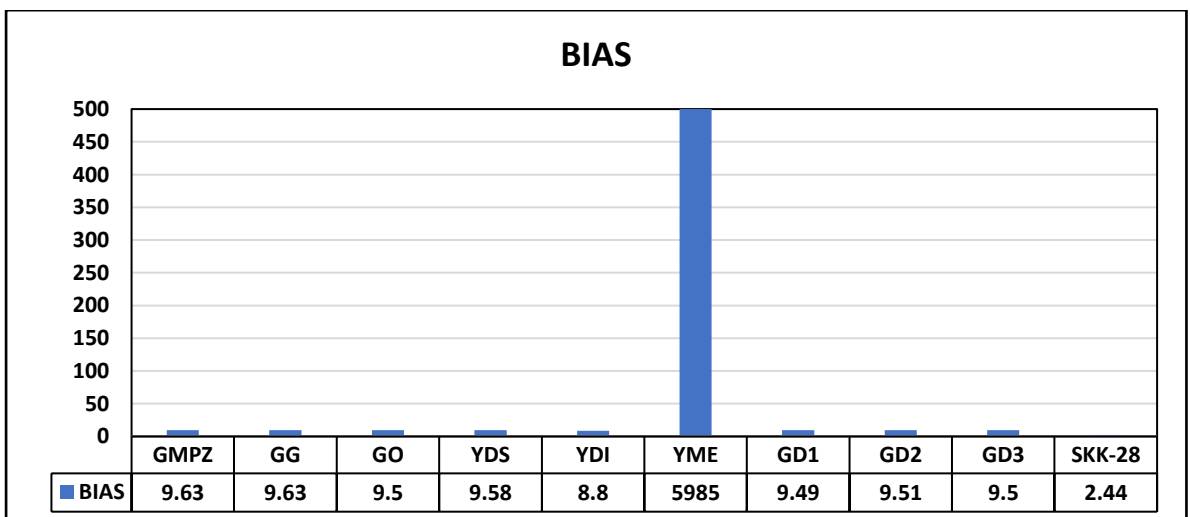


Figure 5.3. Comparison of SKK with other models for bias value.

It is clear from the graph that SKK-28 has the lowest value in comparison to other models.

- **MSE:** Smaller values of MSE are desired for a better fit. The mean squared error (MSE) for each best-predictor candidate model is shown in Figure 5.4. We can see that SKK-28 has the smallest MSE value in comparison to other models for big-fault data.

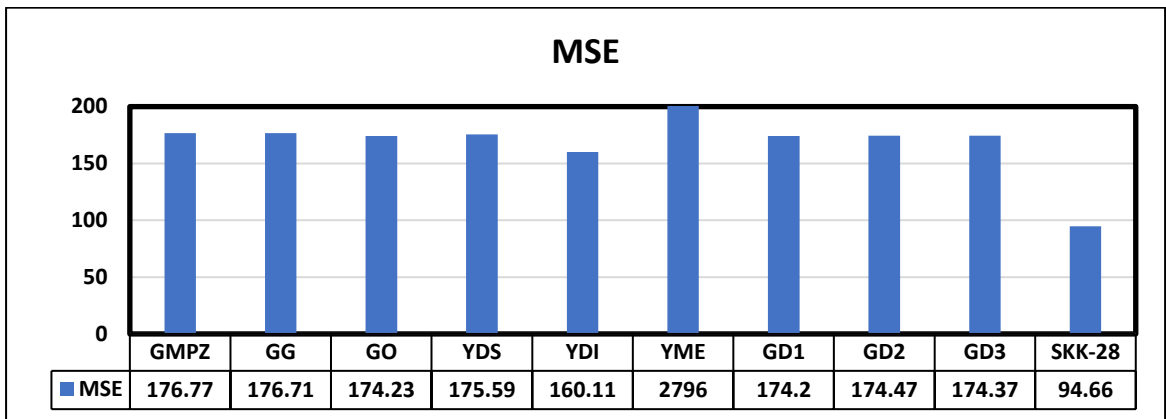


Figure 5.4. Comparative view of MSE values of various models

- **MAD:** Smaller values indicate lesser deviation so a small value of MAD for a model is considered to be the best fit for it. Figure 5.5 represents the MAD values of various models and SKK-28 have the smallest value among them.

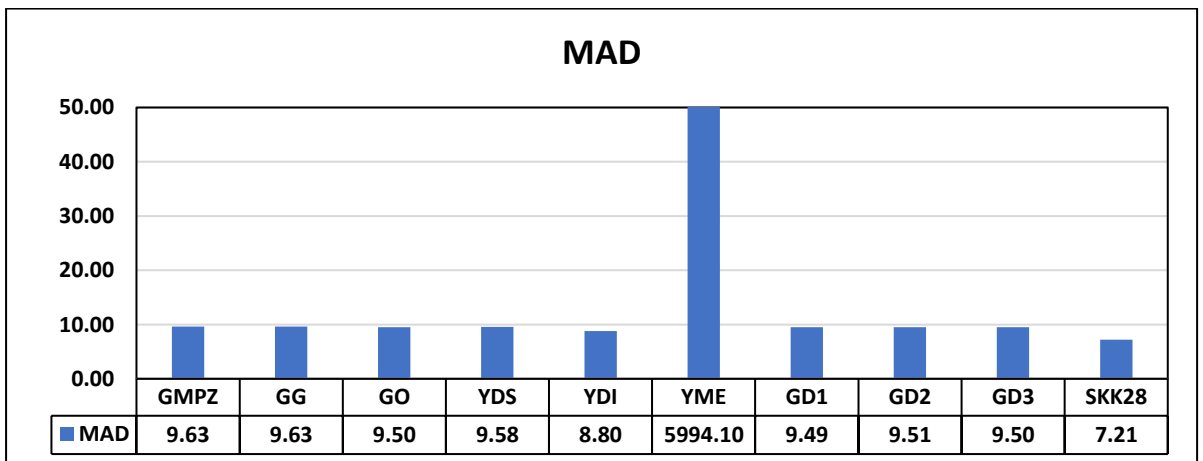


Figure 5.5. Comparative view of MAD values of various models.

- PRR: Small value of PRR is considered best for a model. Figure 5.6 given below demonstrates the PRR values of different models. SKK-28 model is the best estimator in comparison to other models for PRR measure.

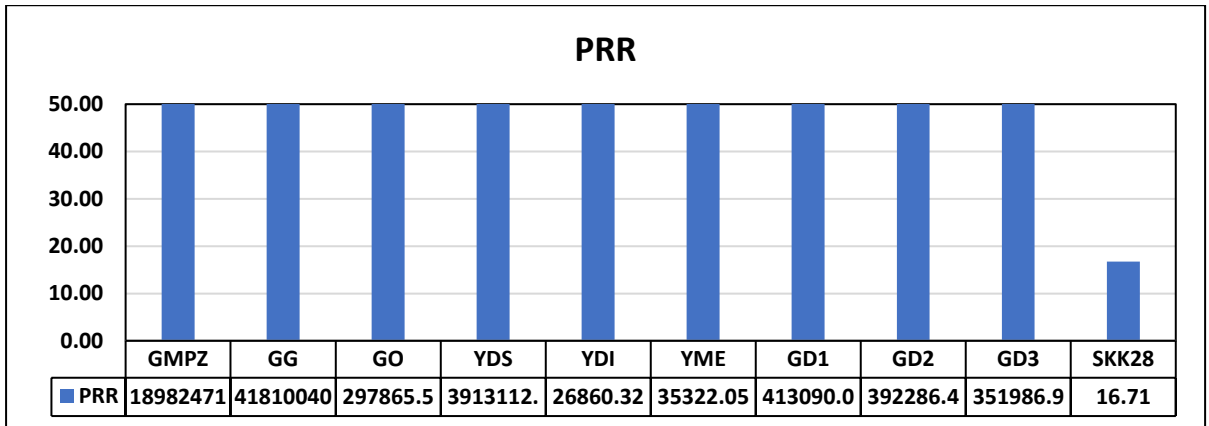


Figure 5.6. Comparative view of PRR values of various models.

- NOISE: Since it represents an error rate smaller values of noise are desirable for a good-fitting model. Figure 5.7 gives the noise in various models for dataset DS#1. GMPZ has the smallest value in comparison to other models.

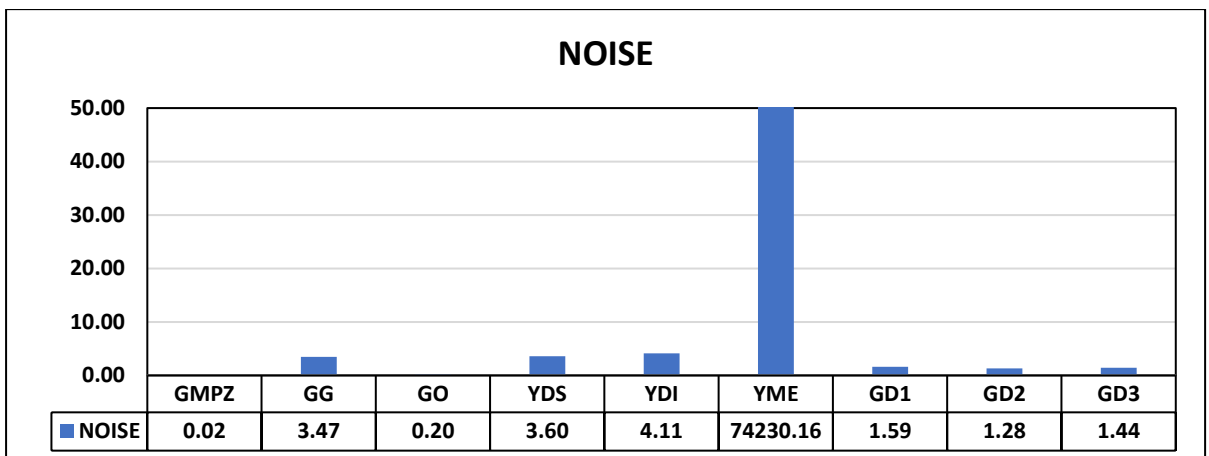


Figure 5.7. Comparative view of NOISE values of various models

- RMSE: Lower values of RMSE are desirable for a better-fitting model. Figure 5.8 given below indicates RMSE values of various models. SKK-28 gives the lowest value of RMSE.

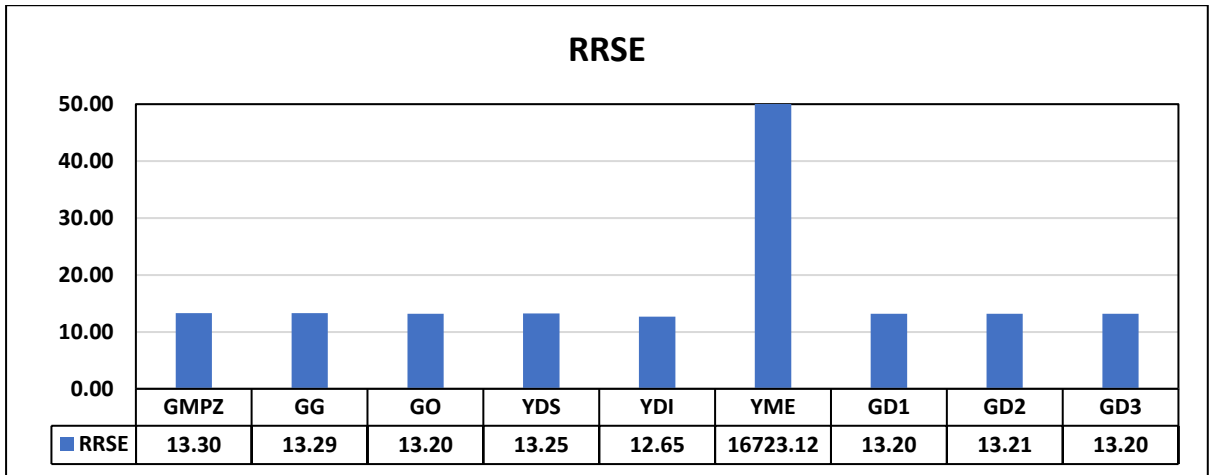


Figure 5.8. Comparative view of RRSE values of various models

- RAE: A model is a better predictor if its RAE value is near zero. Figure 5.9 shows the RAE values of various models in comparison to SKK-28 developed model. With the lowest RAE value, YME is the best option for the DS#2 dataset, followed closely by SKK-28.

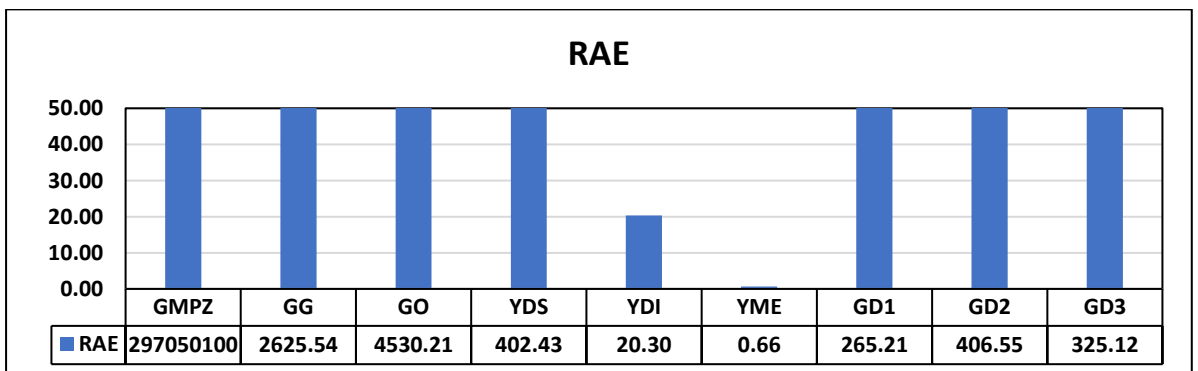


Figure 5.9. Comparative view of RAE values of various models

- RRSE: RRSE values smaller values are desirable for the best predictor model. Figure 5.10 below gives the RRSE value of SKK-28 in comparison with other models. SKK-28 gives the smallest RRSE value.

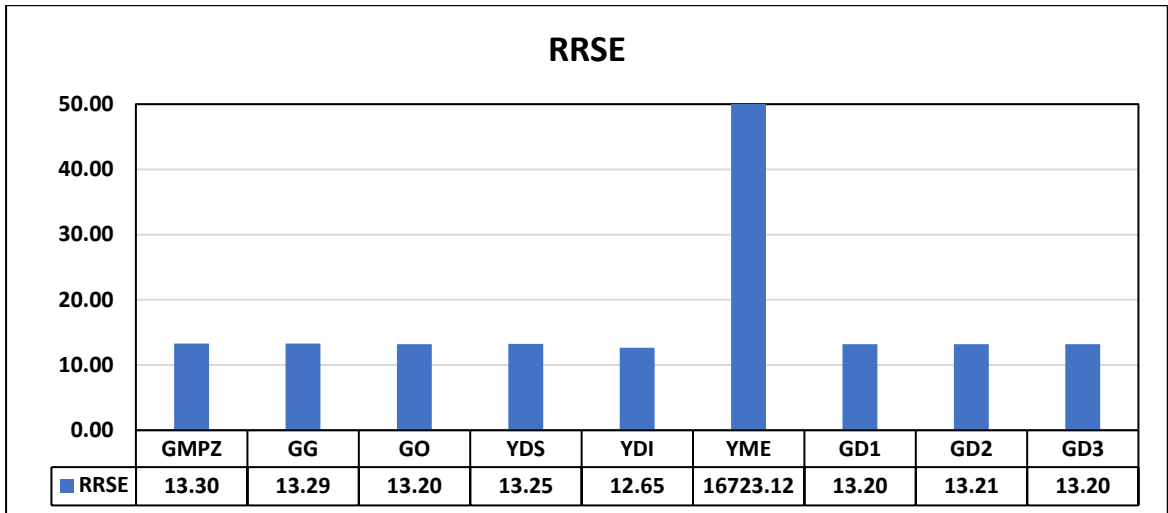


Figure 5.10. Comparative view of RRSE values of various models

- MMRE: For a perfect fit, a model’s MMRE must be closer to zero. Figure 5.11 represents the MMRE value of various candidate models for the best predictor. Among various candidate models, YDI is the best predictor using the MMRE measure.

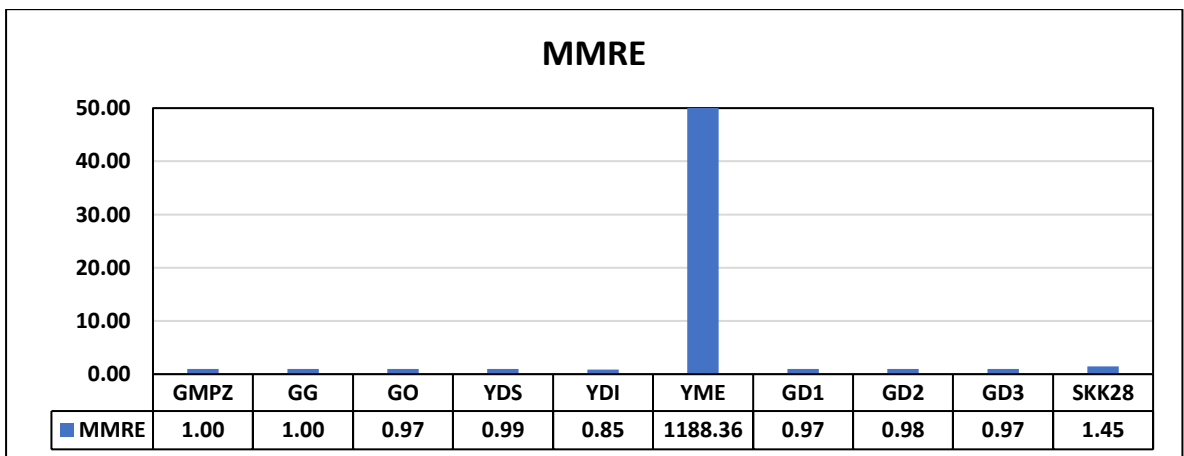


Figure 5.11. Comparative view of MMRE values of various models

- PP: PP is squared deviation, its low values for a model are required for it to be the best predictor. Figure 5.12 shows the value comparison of PP values for various models and SKK-28 gives the lowest value.

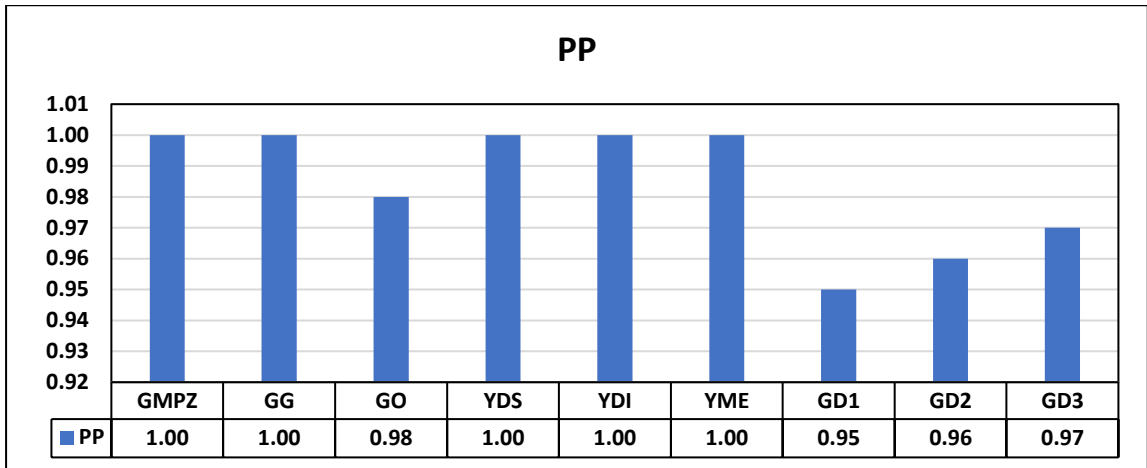


Figure 5.12. Comparative view of PP values of various models

- R^2 : High value of R^2 near 1 is required for a model to be a better predictor. Figure 5.13 given below shows the comparison between various models for R^2 values. Since all values of this measure are negative so smallest negative number represents the maximum value. For this measure too SKK-28 gives better performance than other models.

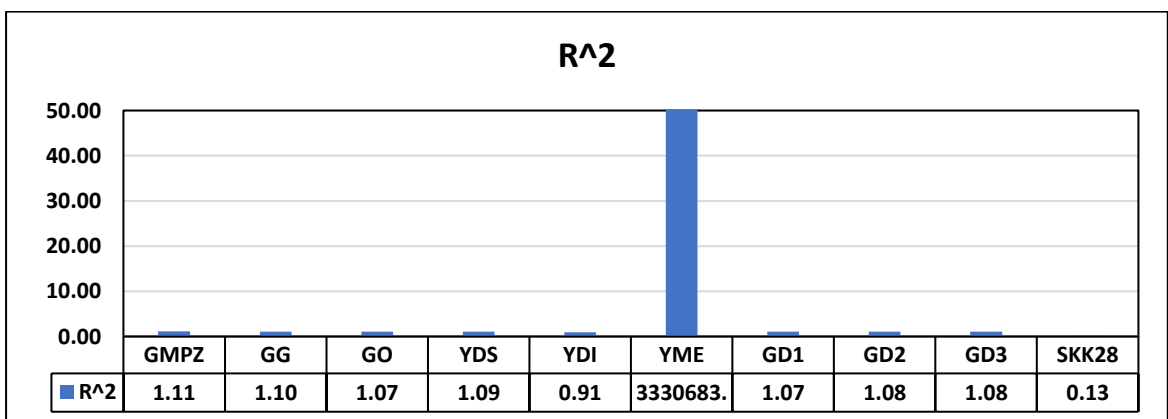


Figure 5.13. Comparative view of R^2 values of various models

- AE: AE represents cumulative error and smaller values near zero give a good fitting model. Figure 5.14 shown below gives us a comparative view of AE values for various models. 0.2 of SKK-28 represents near zero value making it a better-fit model in comparison to other models

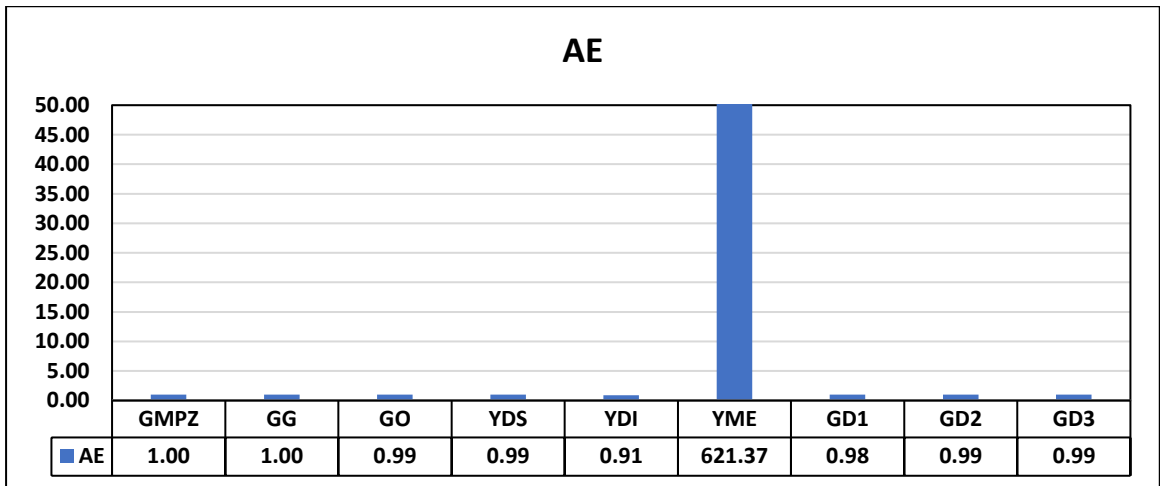


Figure 5.14. Comparative view of AE values of various models

- TS: Smaller values of TS represent a good fit for a model. Figure 5.15 represents a comparative view of various models. SKK-28 has the smallest value of TS.

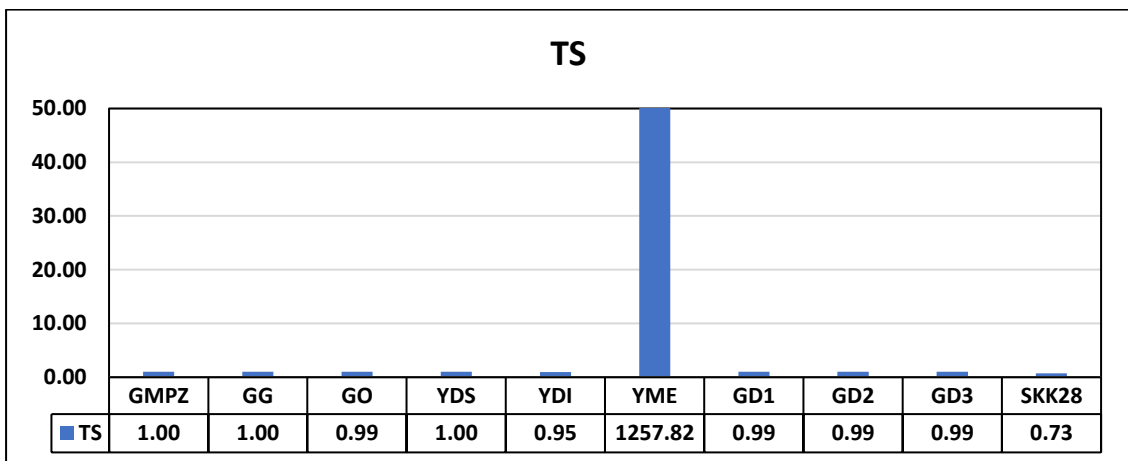


Figure 5.15. Comparative view of TS values of various models

Figure 5.16 gives represents the minimum value of various measures for various models. SKK-28 gives the best fit in ten measures out of 13. GMPZ, YDI, and YME give min measure corresponding to NOISE, RAE, and MMRE and SKK-28 gives a deviation of 5.28,1.14 and 0.60 from these minimum values. Thus, we can safely state that SKK-28 gives the best fit out of GMPZ, GG, YDS, YDI, YME, GD1, GD2

S. No.	Models
1	GMPZ
2	GG
3	GO
4	YDS
5	YDI
6	YME
7	GD1
8	GD2
9	GD3
10	SKK28

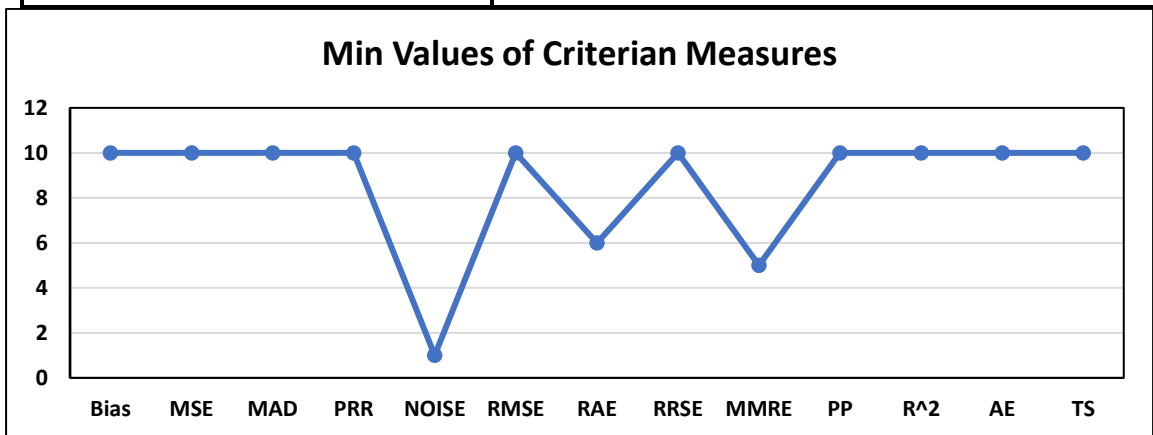


Figure 5.16. Min value of various models for all measures in comparison criteria

5.2. A Statistical Test: t-test

When comparing two samples taken from the same population, the t-test is conducted as shown in Table 5.2 to demonstrate that the difference between the sample means (observed and estimated) is not statistically significant. To further verify that the SKK-28 model gives the best fit we determine whether estimation values give the same variance as observed values for DS#2 fault data. we conducted a t-test using the different variances in Excel. Null and alternate hypotheses are formulated as

- 1) *H0: There is no significant difference between the population mean of observed and estimated values and cumulative values.*
- 2) *H1: There is a significant difference between the population mean of observed and estimated values and cumulative values.*

Table 5.2. t-test for observed and estimated values

t-test: Two-Sample Assuming Unequal Variances		
	Observed values O(i)	SKK-28
Mean	9.6333333	12.077669
Variance	86.86092	22.257145
Observations	30	30
Hypothesized Mean Difference	0	
df	43	
t Stat	-1.281661	
P(T<=t) one-tail	0.1034148	
t Critical one-tail	1.6810707	
P(T<=t) two-tail	0.2068296	
t Critical two-tail	2.0166922	

The value of α is set to default as 0.05. We check the value of p. Null hypothesis is accepted if $p > \alpha$ otherwise we reject it.

Since $(p = 0.1034148) > (\alpha=0.05) \rightarrow$ We accept the null hypothesis, that is there is no significant difference between the population mean of observed and estimated values. So, we can state that both values belong to the same population pool. For cumulative values too the null hypothesis is accepted as shown in Table 5.3, since the value of $(p = 0.310282141) > (\alpha=0.05)$.

Table 5.3. t-test for cumulative observed and estimated values

t-Test: Two-Sample Assuming Unequal Variances		
	<i>cumulative O</i>	<i>Cumulative SKK-28</i>
Mean	134.4666667	147.6292267
Variance	8589.981609	12383.34571
Observations	30	30
Hypothesized Mean Difference	0	
df	56	
t Stat	-0.497814252	
P(T<=t) one-tail	0.310282141	
t Critical one-tail	1.672522303	
P(T<=t) two-tail	0.620564281	
t Critical two-tail	2.003240719	

5.3. RANKING METHODOLOGY VALIDATION USING DS#2

We calculated the value of the parameter given in Table 5.4 by analyzing the DS2 fault data collected by Rajpal Garg et al. (2010).

Table 5.4. Parameters value of NHPP model using DS#2

MODELS	Parameter1	Parameter2	Parameter3
GO	216	0	-
YMD	191	0	-
MO	113	0	-
GG	185	0	3
GMPZ	192	0	0
LGC	188	0	7
YMI	203	0	1
PZIF	191	0	0
MD	238	40	4
YIDM1	128	0	0
YIDM2	128	0	0

We also determine the values of all measures in the criteria set shown in Table 5.5 for all eleven NHPP models using MATLAB 2020 b.

Table 5.5. Criteria's value for NHPP models

MODELS	BIA S	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	R ²	AE	TS
GO	- 0.46	24.37	2.97	60.76	22.96	4.94	0.92	0.76	0.71	0.00	0.42	0.07	0.49
YMD	- 0.33	33.59	3.54	773.7 1	16.09	5.80	0.81	0.90	0.66	0.06	0.19	0.05	0.57
MO	- 0.27	23.70	3.12	19.71	22.62	4.87	1.23	0.75	0.88	0.04	0.43	0.04	0.48
GG	- 0.39	95.80	6.05	28905 60.00	12.12	9.79	1.18	1.52	1.41	0.06	- 1.30	0.06	0.96
GMPZ	- 12.3 4	315.0 7	12.34	786.0 7	14.46	17.75	3.33	2.75	1.78	0.49	- 6.56	1.88	1.75
LGC	- 1.20	41.41	3.95	2229. 61	12.31	6.44	1.14	1.00	0.74	0.00	0.01	0.18	0.63

YMI	12.3 4	289.0 0	12.78	11.12	21.15	17.00	2.95	2.63	4.55	0.21	- 5.93	1.88	1.67
PZIF	- 0.33	33.59	3.54	773.4 5	16.09	5.80	0.81	0.90	0.66	0.93	0.19	0.05	0.57
MD	1429 8.55	33260 8000. 00	14298 .55	20.98	36485.3 1	18237. 54	3.15	2824. 27	3415.26	11.6 1	- 7976 513. 00	217 7.44	179 4.39
YIDM1	- 6.53	102.3 5	6.53	42724 53.00	2.62	10.12	270.9 7	1.57	0.69	0.02	- 1.45	0.99	1.00
YIDM2	- 0.49	23.23	3.04	20.03	22.38	4.82	1.16	0.75	0.81	37.2 5	0.44	0.08	0.47

Models raking based on individual criteria is determined and shown in Table 5.6.

Table 5.6. Ranking of models based on individual criteria measure

MODELS	BIAS	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	R ²	AE	TS
GO	5	3	1	5	10	3	3	3	4	2	3	5	3
YMD	2	4	4	7	6	4	1	4	1	6	4	2	4
MO	1	2	3	2	9	2	7	2	7	4	2	1	2
GG	4	7	7	10	2	7	6	7	8	5	7	4	7
GMPZ	10	10	9	8	4	10	10	10	9	8	10	10	10
LGC	7	6	6	9	3	6	4	6	5	1	6	7	6
YMI	9	9	10	1	7	9	8	9	10	7	9	9	9
PZIF	3	5	5	6	5	5	2	5	2	9	5	3	5
MD	11	11	11	4	11	11	9	11	11	10	11	11	11
YIDM1	8	8	8	11	1	8	11	8	3	3	8	8	8
YIDM2	6	1	2	3	8	1	5	1	6	11	1	6	1

Figure 5.17, represents the graphical view of ranking of models in individual criteria measures.

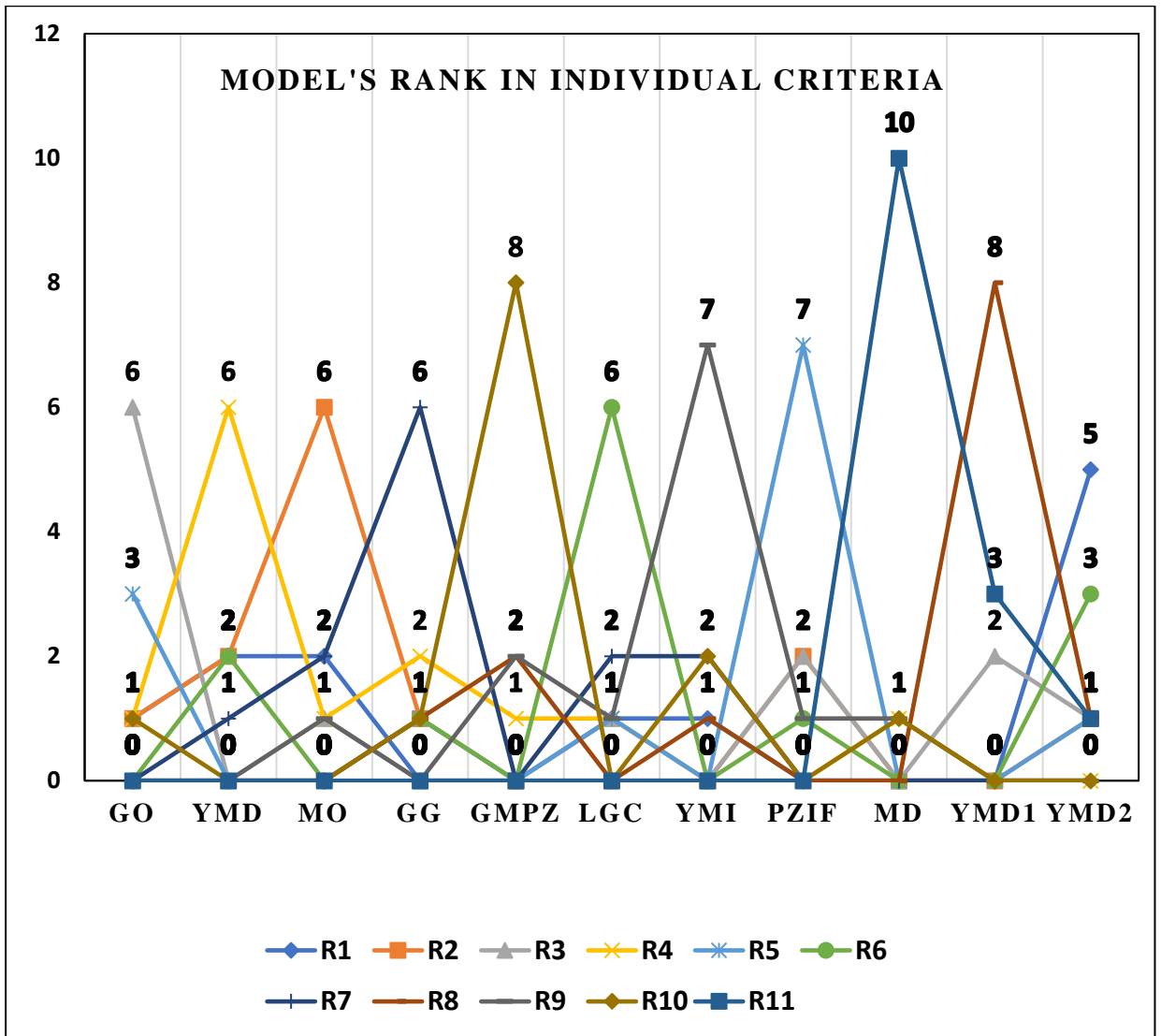


Figure 5.17. Individual criteria measures and the rank of all models

Weights were assigned to ranks and the weight matrix and weighted rank matrix are evaluated in Table 5.7 and Table 5.8.

Table 5.7. Weight matrix

MODEL	BIAS	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	R²	AE	TS
GO	1.0000	0.0010	0.0001	1.0000	6.0000	0.0010	0.0010	0.0010	0.0050	0.0005	0.0010	1.0000	0.0010
YMD	0.0005	0.0050	0.0050	3.0000	2.0000	0.0050	0.0001	0.0050	0.0001	2.0000	0.0050	0.0005	0.0050
MO	0.0001	0.0005	0.0010	0.0005	5.0000	0.0005	3.0000	0.0005	3.0000	0.0050	0.0005	0.0001	0.0005
GG	0.0050	3.0000	3.0000	6.0000	0.0005	3.0000	2.0000	3.0000	4.0000	1.0000	3.0000	0.0050	3.0000
GMPZ	6.0000	6.0000	5.0000	4.0000	0.0050	6.0000	6.0000	6.0000	5.0000	4.0000	6.0000	6.0000	6.0000
LGC	3.0000	2.0000	2.0000	5.0000	0.0010	2.0000	0.0050	2.0000	1.0000	0.0001	2.0000	3.0000	2.0000
YMI	5.0000	5.0000	6.0000	0.0001	3.0000	5.0000	4.0000	5.0000	6.0000	3.0000	5.0000	5.0000	5.0000
PZIF	0.0010	1.0000	1.0000	2.0000	1.0000	1.0000	0.0005	1.0000	0.0005	5.0000	1.0000	0.0010	1.0000
MD	7.0000	7.0000	7.0000	0.0050	7.0000	7.0000	5.0000	7.0000	7.0000	6.0000	7.0000	7.0000	7.0000
YIDM1	4.0000	4.0000	4.0000	7.0000	0.0001	4.0000	7.0000	4.0000	0.0010	0.0010	4.0000	4.0000	4.0000
YIDM2	2.0000	0.0001	0.0005	0.0010	4.0000	0.0001	1.0000	0.0001	2.0000	7.0000	0.0001	2.0000	0.0001

Table 5.8. Weighted rank matrix

MODELS	BIAS	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	R ²	AE	TS
GO	5	0.003	0.0001	5	60	0.003	0.003	0.003	0.020	0.001	0.003	5	0.003
YMD	0.001	0.02	0.02	21	12	0.02	0.0001	0.02	0.0001	12	0.02	0.001	0.02
MO	0.0001	0.001	0.0030	0.0010	45	0.001	21	0.001	21	0.02	0.001	0.0001	0.001
GG	0.02	21	21	60	0.001	21	12	21	32	5	21	0.02	21
GMPZ	60	60	45	32	0.02	60	60	60	45	32	60	60	60
LGC	21	12	12	45	0.003	12	0.02	12	5	0.0001	12	21	12
YMI	45	45	60	0.0001	21	45	32	45	60	21	45	45	45
PZIF	0.003	5	5	12	5	5	0.001	5	0.001	45	5	0.003	5
MD	77	77	77	0.02	77	77	45	77	77	60	77	77	77
YDM1	32	32	32	77	0.0001	32	77	32	0.003	0.003	32	32	32
YDM2	12	0.0001	0.001	0.003	32	0.0001	5	0.0001	12	77	0.0001	12	0.0001

The model rank is calculated by dividing the weighted sum by the weighted estimation function value. The undermentioned Figures 5.18 & 5.19, show the weighted function calculation and estimation function calculation which were used as permanent in literature and in our study respectively.

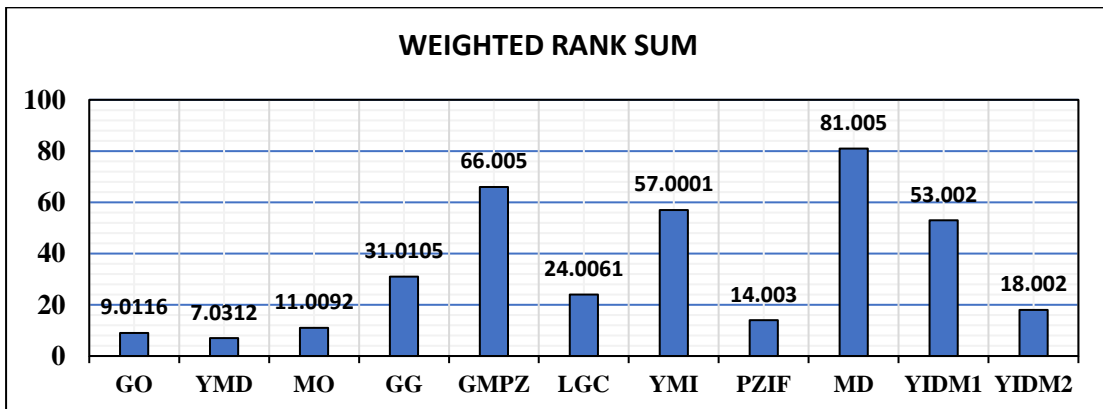


Figure 5.18. Weighted rank sum calculations for existing models

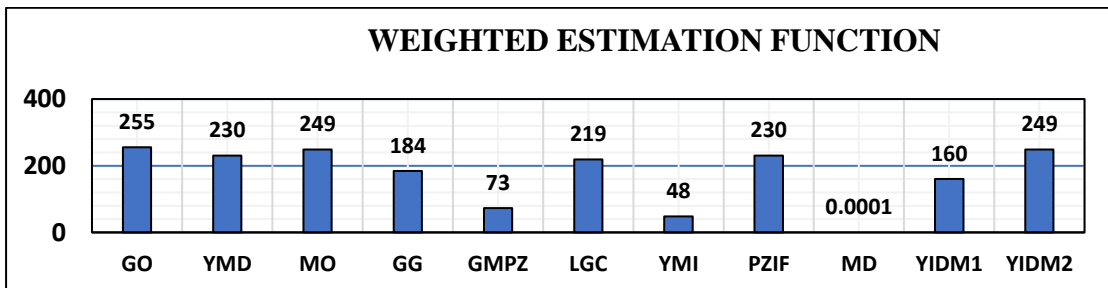


Figure 5.19. Estimation function rank sum calculations for existing models

Table 5.9 shows the models arranged according to their rank, where the lowest rank specifies the best model.

Table 5.9. Ranking of existing models based on the developed methodology

S. No.	Models	Rank
1	GO	2
2	YMD	1
3	MO	3
4	GG	7
5	GMPZ	9
6	LGC	6
7	YMI	10
8	PZIF	4
9	MD	11
10	YIDM1	8
11	YIDM2	5

Undermentioned Figure 5.20 and Figure 5.21, give a comparative view of model ranking using the developed methodology in our research and the existing, methodology in the literature

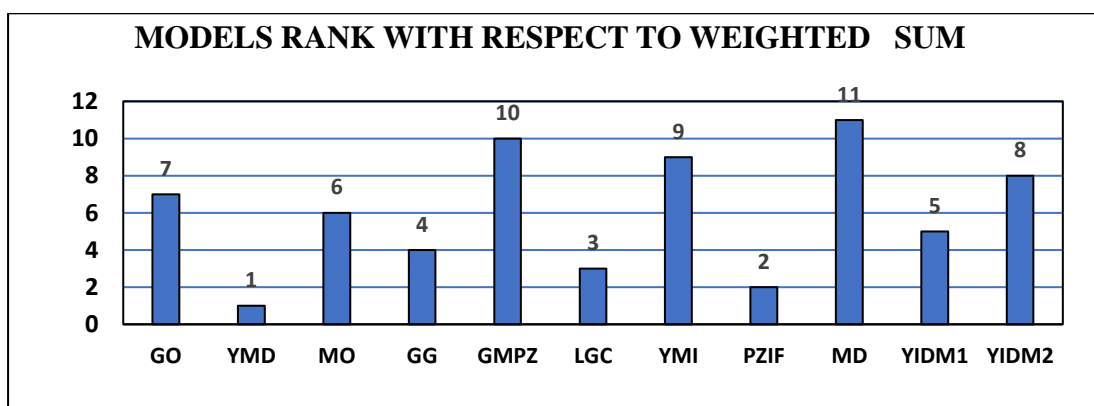


Figure 5.20. Models ranking using existing methods in literature

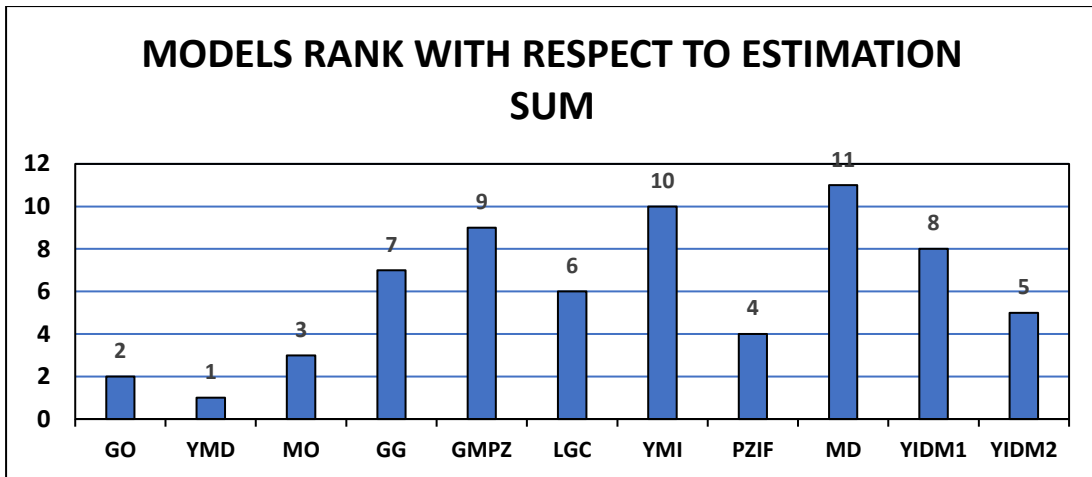


Figure 5.21. Models ranking using developed methodology

We observed that even though YIMD2 is the best model according to min criteria values but our proposed approach shows that the best model is YMD as it considers the weighted criteria ranks as well as the estimation capability of the model into account.

5.4. VALIDATING DEVELOPED MODELS USING RANKING METHODOLOGY FOR DS#3

We utilized Dataset DS#1 which is a fault dataset collected during Big Data Analysis and selected the Best Model SKK-28 from the candidate models. We also tested the candidate models for their performance on other Dataset DS#3 and DS#4, which is not related to Big-data. We utilized a developed ranking methodology to determine the performance of our candidate models to rank them against well-known existing NHPP models using criteria set. We used MSE, RMSE, RAE, and RRSE measures in the criteria set. The undermentioned tables from Table 5.10 to Table 5.13 represent the weighted rank matrix.

Table 5.10. WRM using LSE

LSE	R_MSE	R_RMSE	R_RAE	R_RRSE
GO	0.01	0.01	0.003	0.1
MO	0.003	0.003	0.01	0.05
YDM	0.002	0.002	0.002	0.01
GG	0.1	0.1	0.1	4
GMPZ	0.05	0.05	1	1
MD	0.0001	0.0001	4	0.002
YIM	40	40	0.0001	40
SKK2	1	1	40	0.0001
SKK9	4	4	12	0.003
SKK10	12	12	0.05	12

Table 5.11. WRM using GA

GA	R_MSE	R_RMSE	R_RAE	R_RRSE
GO	0.01	0.05	0.05	0.1
MO	0.003	0.01	0.01	0.05
YDM	0.05	0.1	12	1
GG	4	12	0.0001	12
GMPZ	0.1	1	40	4
MD	0.002	0.003	4	0.003
YIM	40	40	0.003	40
SKK2	0.0001	0.002	0.1	0.002
SKK9	1	4	1	0.01
SKK10	12	0.0001	0.002	0.0001

Table 5.12. WRM using SA

SA	R_MSE	R_RMSE	R_RAE	R_RRSE
GO	0.003	0.003	4	0.05
MO	0.05	0.05	0.003	1
YDM	0.01	0.01	0.05	0.1
GG	4	4	0.0001	4
GMPZ	0.0001	0.0001	0.01	0.0001
MD	0.002	0.002	1	0.002
YIM	40	40	0.002	40
SKK2	1	1	40	0.01
SKK9	0.1	0.1	12	0.003
SKK10	12	12	0.1	12

Table 5.13. WRM using PSO

PSO	R_MSE	R_RMSE	R_RAE	R_RRSE
GO	0.003	0.003	0.1	0.003
MO	0.0001	0.0001	0.05	0.0001
YDM	0.002	0.002	0.01	0.002
GG	1	1	0.002	1
GMPZ	0.01	0.01	1	0.05
MD	0.05	0.05	12	0.1
YIM	12	12	0.003	12
SKK2	0.1	0.1	40	0.01
SKK9	40	40	0.0001	10
SKK10	4	4	4	4

The evaluated rank of seven existing NHPP models and three best-performing models are shown in Table 5.14 to Table 5.17 corresponding to LSE, GA, SA, and PSO evaluation methods for parameter optimization.

Table 5.14. Rank matrix using LSE

Models	LSE_Rank
GO	4
MO	6
YDM	2
GG	7
GMPZ	1
MD	8
YIM	5
SKK2	9
SKK9	3
SKK10	10

Table 5.15. Rank matrix using GA

Models	GA_Rank
GO	4
MO	6
YDM	2
GG	7
GMPZ	1
MD	8
YIM	5
SKK2	9
SKK9	3
SKK10	10

Table 5.16. Rank matrix using SA

Models	SA_Rank
GO	7
MO	5
YDM	3
GG	8
GMPZ	10
MD	9
YIM	6
SKK2	2
SKK9	4
SKK10	1

Table 5.17. Rank matrix using PSO

Models	PSO_Rank
GO	7
MO	10
YDM	9
GG	6
GMPZ	4
MD	2
YIM	5
SKK2	3
SKK9	8
SKK10	1

Table 5.18. Given below gives the average rank of selected models

MODELS	LSE	GA	SA	PSO	AVG_RAN
					K
GO	4	4	7	7	3
MO	6	6	5	10	5
YDM	2	2	3	9	9
GG	7	7	8	6	7
GMPZ	1	1	10	4	1
MD	8	8	9	2	10
YIM	5	5	6	5	8
SKK2	9	9	2	3	2
SKK9	3	3	4	8	6
SKK10	10	10	1	1	4

According to Table 5.18, the ensemble of proposed models shows good performance if not the best as SKK-9 is ranked 3rd when using LSE and GA method whereas GMPZ is ranked 1st and YDM. SKK-10 is ranked 1st, whereas SKK -2 is ranked 2nd and 3rd when evaluating parameters using SA and PSO. The average rank evaluation shows that GMPZ is the best-performing model followed by GO and then SKK-2 in third place. It is clear that even though models SKK-1 to SKK-10 were developed for big data but they are showing good performance even though we use other fault data sets.

5.5. VALIDATING DEVELOPED MODELS USING RANKING METHODOLOGY FOR DS#4

We tested the candidate models for their performance for Dataset DS#3 according to their average rank by combining all methods. We calculated the model's criteria values using LSE, GA, SA, and PSO contained in tables from Table 5.19 to Table 5.22 by including the first three ranked models from SKK-1 to SKK-10 using four criteria

measures. The rank of models is determined by using these three models along with seven existing NHPP models to demonstrate their performance for DS#4, which is also not related to Big-data.

Table 5.19. WRM using LSE

LSE	R_MSE	R_RMSE	R_RAE	R_RRSE
GO	0.01	0.01	0.003	0.1
MO	0.003	0.003	0.01	0.05
YDM	0.002	0.002	0.002	0.01
GG	0.1	0.1	0.1	4
GMPZ	0.05	0.05	1	1
MD	0.0001	0.0001	4	0.002
YIM	40	40	0.0001	40
SKK8	1	1	40	0.0001
SKK9	4	4	12	0.003
SKK10	12	12	0.05	12

Table 5.20. WRM using GA

GA	R_MSE	R_RMSE	R_RAE	R_RRSE
GO	0.01	0.05	0.05	0.1
MO	0.003	0.01	0.01	0.05
YDM	0.05	0.1	12	1
GG	4	12	0.0001	12
GMPZ	0.1	1	40	4
MD	0.002	0.003	4	0.003
YIM	40	40	0.003	40
SKK2	0.0001	0.002	0.1	0.002
SKK3	1	4	1	0.01
SKK4	12	0.0001	0.002	0.0001

Table 5.21. WRM using SA

SA	R_MSE	R_RMSE	R_RAE	R_RRSE
GO	0.003	0.003	4	0.05
MO	0.05	0.05	0.003	1
YDM	0.01	0.01	0.05	0.1
GG	4	4	0.0001	4
GMPZ	0.0001	0.0001	0.01	0.0001
MD	0.002	0.002	1	0.002
YIM	40	40	0.002	40
SKK4	1	1	40	0.01
SKK5	0.1	0.1	12	0.003
SKK10	12	12	0.1	12

Table 5.22. WRM using PSO

PSO	R_MSE	R_RMSE	R_RAE	R_RRSE
GO	0.003	0.003	0.05	0.003
MO	0.0001	0.0001	0.01	0.0001
YDM	0.002	0.002	0.003	0.002
GG	1	1	0.0001	1
GMPZ	0.01	0.01	0.1	0.05
MD	0.05	0.05	12	0.1
YIM	40	40	0.002	40
SKK2	0.1	0.1	40	0.01
SKK6	12	12	4	12
SKK10	4	4	1	4

Table 5.23 to Table 5.26 below contains the Ranking corresponding to the evaluation method.

Table 5.23. Rank matrix using LSE

MODEL	LSE(WR)
GO	3
MO	2
YDM	1
GG	6
GMPZ	4
MD	5
YIM	10
SKK8	9
SKK9	7
SKK10	8

Table 5.24. Rank matrix using GA

MODEL	GA(WR)
GO	3
MO	1
YDM	7
GG	8
GMPZ	9
MD	4
YIM	10
SKK2	2
SKK3	5
SKK4	6

Table 5.25. Rank atrix using SA

MODEL	SA(WR)
GO	5
MO	4
YDM	2
GG	6
GMPZ	1
MD	3
YIM	10
SKK4	9
SKK5	7
SKK10	8

Table 5.26. Rank matrix using PSO

MODEL	PSO(WR)
GO	3
MO	2
YDM	1
GG	5
GMPZ	4
MD	6
YIM	10
SKK2	9
SKK6	8
SKK10	7

Developed model SKK-2 ranked 2nd and SKK-9 and SKK-10 as 5th and 6th when using GA for optimization. In the case of LSE, SA, and PSO developed models didn't show good performance but still not the worst performance. Thus, we may conclude that model performance is very sensitive to the dataset and assessment technique used.

5.6. PARAMETER OPTIMIZATION OF SKK-28 MODEL USING GA, SA, AND PSO METHODS

- The parameter's value of the SKK-28 model was evaluated using LSE and optimized by using GA, SA, and PSO using the LSE equation as an objective function.
- All thirteen measures of the comparison criteria set were evaluated.
- These measures were then ranked in increasing order to have an idea regarding the best soft computing technique as an optimizer.
- Any technique is best than others if it includes the min value in maximum criteria measures.
- Estimated values were also calculated using the value of the parameters obtained by GA, SA, and PSO techniques.

The graph in Figure 5.22, below shows the individual criteria ranking with respect to various soft computing methods.

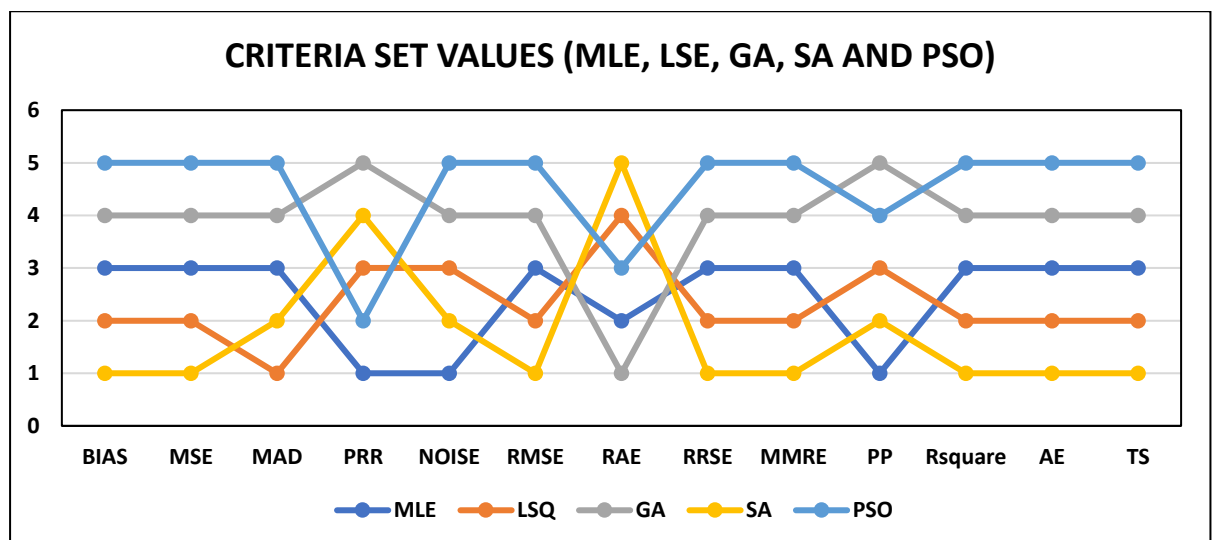


Figure 5.22. Criteria set values of SKK-28 using various methods

- SA (Yellow line) has a minimum value in most criteria measures – 8.
- MLE (Blue line) gives the minimum value in 3.
- LSE (Orange line) gives the minimum value in 1.
- GA (Grey line) gives minimum value in 1 criterion.
- Out of the three optimizers PSO raked last concerning criteria set values.

Thus, we can safely say that SA is the best optimizer. Shown below in Figure 5.23 are the maximum number of criteria values (Cobalt blue bar) corresponding to increasing order of minimum values (first minimum (blue bar), second minimum (orange bar), third minimum (grey bar), etc.)

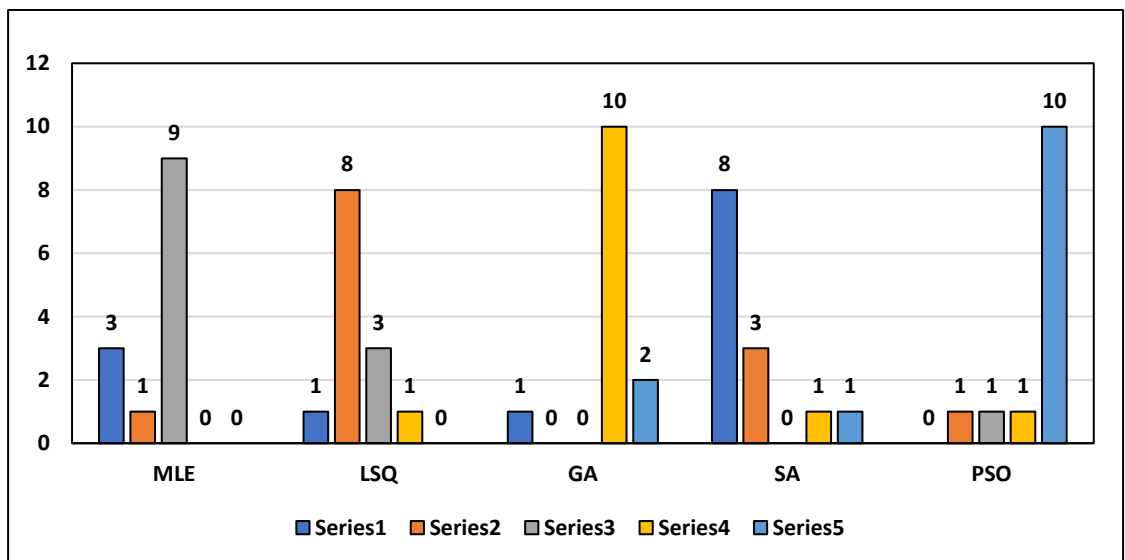


Figure 5.23. Maximum number of minimum criteria values of SKK-28

Shown below in Figure 5.24. is the estimation capability of the SKK-28 model using various evaluation methods. It is clear from the figure that SA shows the best and PSO worst Estimation capability among all optimization methods.

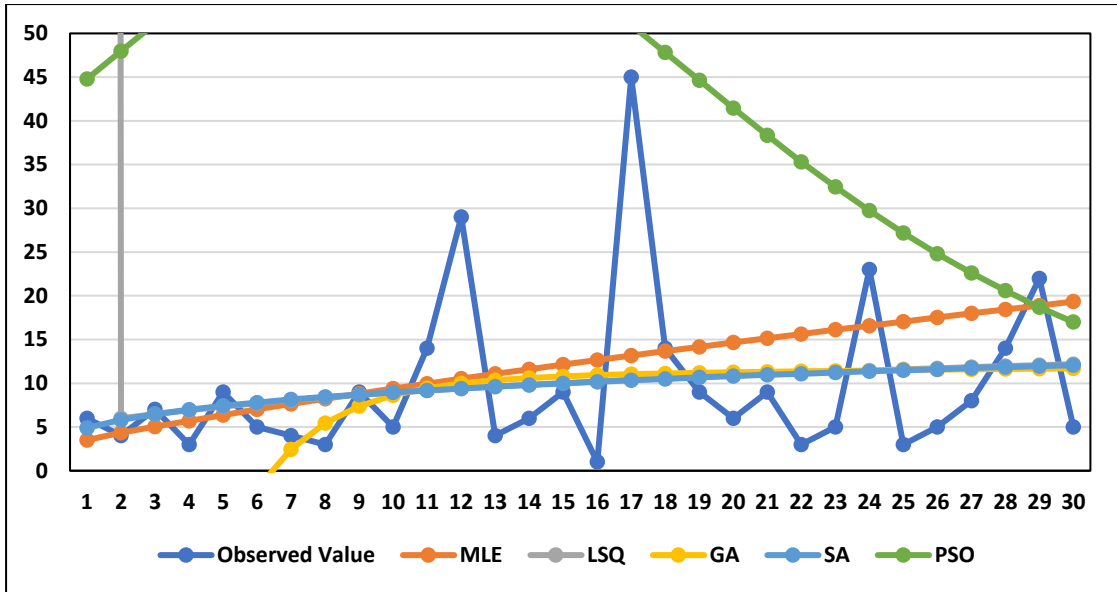


Figure 5.24. Estimation capacity of SKK-28 using various methods

CONCLUSION

The goal of validation is to determine whether or not the model is fit for its designated function. The result shows that model SKK-28 is the best fit among other candidate models considered in the comparison. Utilizing other datasets i.e., DS#3 and DS#4 we confirmed that our developed models show good performance not only for big fault data but other software fault data too. Soft computing optimization methods i.e., GA, SA, and PSO were used for parameter evaluation apart from LSE and MLE and we found that SA is the best optimizer while PSO is worst in the case of chosen best model SKK-28 for the dataset DS#1.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

The model development was considered by utilizing NHPP models to obtain various hybrid models suitable for big data environments. A ranking methodology and estimation function was developed to select the best model out of a few based on their prediction capabilities. Various parameter evaluation techniques were employed to determine the one suitable for highly complex non-linear reliability models.

6.1. CONCLUSION

Some major findings of said research work are

1. Popular and widely used NHPP models are combined to obtain a hybrid model which itself is an NHPP model, this characteristic of NHPP models was used in developing hybrid models based upon different scenarios.
2. Applications of Big data range from home agriculture, industry, tourism, transportation, medicines, etc. Acquiring big data tools and techniques helps in managing data and increasing the organization's capability to capture required data.
3. A manager can choose a software reliability model from a readily available one based on the requirements of the product. With the advent of big-data applications and changes in the architecture and technologies used, the complexity of models keeps on increasing to accommodate the parameters reflecting new interactions making their implementation difficult.
4. Parameters were usually evaluated using two statistical techniques LSE and MLE. We evaluated the selected ten models based on their accurate estimation and find that MLE gives better results than LSM in the case of developed hybrid non-linear models.

5. A ranking methodology was developed based on accurate estimation and comparison criteria. Comparison criteria consist of thirteen error measures and it was found that a model having maximum low ranks in maximum criteria doesn't need to be a good predictor. Including the estimation capability of models in determining their rank ensures that a model with consistent performance in all criteria is ranked above others.
6. Soft computing optimization algorithms GA, SA, and PSO were utilized to optimize the parameters value of SKK-28. It was found that SA is the best optimizer while PSO is the worst optimizer for SKK-28 using DS#1.
7. The best-selected model SKK-28 shows the best performance for DS#1, good for DS#3, and bad for DS#4. Another developed candidate model SKK-2 performed better than SKK-28 and other existing NHPP models for DS#3 in all three considered soft computing techniques. SKK-2 also performed well for DS#4 using GA.

6.2. SCOPE FOR FUTURE RESEARCH

The developed hybrid models can be tested for various other applications using various datasets.

- Hybrid models can be modified to include other external factors like interactions between open-source software, and networking software and their effect on the reliability of big data processing software.
- Optimization algorithms combining two soft computing techniques or one statistical and other soft computing can be developed for parameter evaluation of reliability models.

- A developed ranking methodology and estimation function can be utilized to rank and select the model for a specific application.

REFERENCES

- [1] AL-Saati, D. N. A., & Abd-AlKareem, M. (2013). "The Use of Cuckoo Search in Estimating the Parameters of Software Reliability Growth Models". (IJCSIS) International Journal of Computer Science and Information Security, *11*(6). <http://arxiv.org/abs/1307.6023>.
- [2] Anjum, M., Haque, M. A., & Ahmad, N. (2013). "Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value". International Journal of Information Technology and Computer Science, *5*(2), pp. 1–14. <https://doi.org/10.5815/ijitcs.2013.02.01>.
- [3] Banga, M., Bansal, A., & Singh, A. (2019). "Proposed hybrid approach to predict software fault detection". International Journal of Performability Engineering, *15*(8), pp. 2049–2061. <https://doi.org/10.23940/ijpe.19.08.p4.20492061>.
- [4] Bellon-maurel, V., Brossard, L., Garcia, F., & Termier, A. (2022). "Agriculture and Digital Technology: Getting the most out of Digital Technology to Contribute to the Transition to Sustainable Agriculture and Food Systems". HAL, pp. 1-185, [ff10.17180/wmkb-ty56-enff. fihal-03604970](https://hal.archives-ouvertes.fr/hal-03604970).
- [5] Bidhan, K., & Awasthi, A. (2014). "Estimation of Reliability Parameters of Software Growth Models using a Variation of Particle Swarm Optimization". *Confluence The Next Generation Information Technology Summit (Confluence)*, IEEE, Noida, pp. 800-805.
- [6] Blackburn, M., & Huddell, B. (2012). "Hybrid Bayesian Network Models for Predicting Software Reliability". *International Conference on Software Security and Reliability Companion*, IEEE, Gaithersburg, pp. 33–34. <https://doi.org/10.1109/SERE-C.2012.38>.
- [7] Bo, Y., & Xiang, L. (2007). "A Study on Software Reliability Prediction based on Support Vector Machines". *International Conference on Industrial Engineering and Engineering Management*, IEEE, Singapore, pp. 1176-1180. [https://doi: 10.1109/IEEM.2007.4419377](https://doi.org/10.1109/IEEM.2007.4419377).

- [8] Cai, L., & Zhu, Y. (2015). “The Challenges of Data Quality and Data Quality Assessment in The Big Data Era”. *Data Science Journal*, **14**, pp. 1–10. <https://doi.org/10.5334/dsj-2015-002>.
- [9] Caiuta, R., Pozo, A., Emmendorfer, L., & Vergilio, S. R. (2008). “Selecting Software Reliability Models with a Neural Network Meta Classifier”. *International Joint Conference on Neural Networks*, IEEE, Hongkong, pp. 3747–3754. <https://doi.org/10.1109/IJCNN.2008.4634336>.
- [10] Cao, R., & Gao, J. (2018). “Research on Reliability Evaluation of Big Data System”. *2018 3rd IEEE International Conference on Cloud Computing and Big Data Analysis*, IEEE, pp. 261–265. <https://doi.org/10.1109/ICCCBDA.2018.8386523>.
- [11] Cao, Y., & Zhu, Q. (2010). “The Software Reliability Model using Hybrid Model of Fractals and ARIMA”. *IEICE Transactions on Information and Systems*, **E93-D(11)**, pp. 3116–3119. <https://doi.org/10.1587/transinf.E93.D.3116>
- [12] Chan, T., & Shi, K. (2011). “Parameter Estimation Using Neural Networks”. *Applied Intelligent Control of Induction Motor Drives*, John Wiley & Sons, Singapore, pp. 199–241. <https://doi.org/10.1002/9780470825587.ch8>
- [13] Chang, P. T., Lin, K. P., & PF, P. (2004). “Hybrid Learning Fuzzy Neural Models in Forecasting Engine System Reliability”. *Fifth Asia Pacific Industrial Engineering and Management Systems Conference*, pp. 2361–2366.
- [14] Chiu, K. C., Huang, Y. S., & Lee, T. Z. (2008). “A Study of Software Reliability Growth from the Perspective of Learning Effects”. *Reliability Engineering and System Safety*, **93(10)**, pp. 1410–1421. <https://doi.org/10.1016/j.res.2007.11.004>
- [15] Choudhary, A., Baghel, A. S., & Sangwan, O. P. (2017). “An Efficient Parameter Estimation of Software Reliability Growth Models using Gravitational Search Algorithm”. *International Journal of System Assurance Engineering and Management*, **8(1)**, pp. 79–88. <https://doi.org/10.1007/s13198-016-0541-0>
- [16] Choudhary, A., Baghel, A. S., & Sangwan, O. P. (2018). “Parameter Estimation of Software Reliability Model using Firefly Optimization”. *Advances in*

- Intelligent Systems and Computing, **542**, pp. 407–415.
https://doi.org/10.1007/978-981-10-3223-3_39
- [17] De Bustamante, A. S., & De Bustamante, B. S. (2003). “Multinomial Exponential Reliability Function: A Software Reliability Model”. *Reliability Engineering and System Safety*, **79**(3), pp.281–288. [https://doi.org/10.1016/S0951-8320\(02\)00160-6](https://doi.org/10.1016/S0951-8320(02)00160-6)
- [18] Diwaker, C., & Goyat, S. (2014). “Parameter Estimation of Software Reliability Growth Models Using Simulated Annealing Method”. *International Journal of Computer Applications Technology and Research*, **3**(6), pp. 377–380. <https://doi.org/10.7753/ijcatr0306.1013>
- [19] Gandhi, P., Khan, M. Z., Sharma, R. K., Alhazmi, O. H., Bhatia, S., & Chakraborty, C. (2022). “Software Reliability Assessment using Hybrid Neuro-Fuzzy Model”. *Computer Systems Science and Engineering*, **41**(3), pp. 891–902. <https://doi.org/10.32604/csse.2022.019943>
- [20] Gao, Z. M., & Zhao, J. (2019). “An improved Grey Wolf Optimization Algorithm with Variable Weights”. *Computational Intelligence and Neuroscience*, <https://doi.org/10.1155/2019/2981282>
- [21] Garg, Rajpal, Sharma, K., Kumar, R., & Garg, R. K. (2010a). “Performance Analysis of Software Reliability Models using Matrix Method”. *World Academy of Science, Engineering and Technology*, **71**(11), pp. 31–38.
- [22] Garg, Rakesh. (2019). “Parametric Selection of Software Reliability Growth Models using Multi-Criteria Decision Making Approach”. *International Journal of Reliability and Safety*, **13**(4), pp. 291–309. <https://doi.org/10.1504/IJRS.2019.102888>
- [23] Ghavifekr, S., & Wong, S. Y. (2021). “Role of Big Data in Education: Challenges and Opportunities for the Digital Revolution in Malaysia”. *Handbook of Research on Big Data, Green Growth, and Technology Disruption in Asian Companies and Societies*, pp. 22-37.
- [24] Gill, S. S., Buyya, R., & Chana, I. (2017). “IoT based Agriculture as a Cloud and Big Data Service: The Beginning of Digital India”. *Journal of Organizational and*

- End User Computing, **29**(4), pp. 1–23.
<https://doi.org/10.4018/JOEUC.2017100101>
- [25] Goel, A. L. (1985). “Software Reliability Models: Assumptions, Limitations, and Applicability”. *IEEE Transactions on Software Engineering*, **SE-11**(12), pp. 1411–1423. <https://doi.org/10.1109/TSE.1985.232177>
- [26] Goel, A. L., & Okumoto, K. (1979). “Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures”. *IEEE Transactions on Reliability*, **R-28**(3), pp. 206–211. <https://doi.org/10.1109/TR.1979.5220566>
- [27] Gokhale, S. S., Marinos, P. N., & Trivedi, K. S. (1996). “Important Milestones in Software Reliability Modelling”. *Seke*, pp. 345–352.
- [28] Gokhale, S. S., Wong, W. E., Horgan, J. R., & Trivedi, K. S. (2004). “An Analytical Approach to Architecture Based Software Performance and Reliability Prediction”. *Performance Evaluation*, **58**(4), pp. 391–412. <https://doi.org/10.1016/j.peva.2004.04.003>
- [29] Govindasamy, P., & Dillibabu, R. (2020b). “Development of Software Reliability Models Using a Hybrid Approach and Validation of the Proposed Models using Big Data”. *Journal of Supercomputing*, Springer, **76**(4), pp. 2252–2265. <https://doi.org/10.1007/s11227-018-2457-8>
- [30] Han, X., Tian, L., Yoon, M., & Lee, M. (2012). “A Big Data Model supporting Information Recommendation in Social Networks”. *2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications*, Xiangtan, pp. 810–813. <https://doi.org/10.1109/CGC.2012.125>
- [31] Haque, M. A., & Ahmad, N. (2021). “An Effective Software Reliability Growth Model”. *Safety and Reliability*, pp. 1–12. <https://doi.org/10.1080/09617353.2021.1921547>
- [32] Hema Latha, D., & Premchand, P. (2018). “Estimating Software Reliability using Ant Colony Optimization Technique with Salesman Problem for Software Process”. *International Journal of Advanced Trends in Computer Science and Engineering*, **7**(2), pp. 20–29. <https://doi.org/10.30534/ijatcse/2018/04722018>

- [33] Hong, Y., Zhang, M., & Meeker, W. Q. (2018). “Big data and reliability applications: The complexity dimension”. *Journal of Quality Technology*, **50**(2), pp. 135–149. <https://doi.org/10.1080/00224065.2018.1438007>
- [34] Hossain, S. A., & Dahiya, R. C. (1993). “Estimating the Parameters of a Non-homogeneous Poisson-Process Model for Software Reliability”. *IEEE Transactions on Reliability*, **42**(4), pp. 604–612. <https://doi.org/10.1109/24.273589>
- [35] Hu, L., Liu, K. Y., Diao, Y., Meng, X., & Sheng, W. (2017). “Operational Reliability Evaluation Method based on Big Data Technology”. *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Chengdu, pp. 341–344. <https://doi.org/10.1109/CyberC.2016.71>
- [36] Huang, C. Y., & Huang, W. C. (2008). “Software Reliability Analysis and Measurement using Finite and Infinite Server Queueing Models”. *IEEE Transactions on Reliability*, **57**(1), pp. 192–203. <https://doi.org/10.1109/TR.2007.909777>
- [37] Huang, C. Y., & Kuo, S. Y. (2002). “Analysis of Incorporating Logistic Testing Effort Function into Software Reliability Modelling”. *IEEE Transactions on Reliability*, **51**(3), pp. 261–270. <https://doi.org/10.1109/TR.2002.801847>
- [38] Huang, C. Y., Hung, T. Y., & Hsu, C. J. (2009). “Software Reliability Prediction and Analysis using Queueing Models with Multiple Change-Points”. *International Conference on Secure Software Integration Reliability Improvement*, Shanghai, pp. 212–221. <https://doi.org/10.1109/SSIRI.2009.11>
- [39] Huang, C. Y., Lyu, M. R., & Kuo, S. Y. (2003). “A Unified Scheme of some Nonhomogenous Poisson Process Models for Software Reliability Estimation”. *IEEE Transactions on Software Engineering*, **29**(3), pp. 261–269. <https://doi.org/10.1109/TSE.2003.1183936>
- [40] Hwang, S., & Pham, H. (2009). “Quasi Renewal Time Delay Fault Removal Consideration in Software Reliability Modelling”. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, **39**(1), pp. 200–209. <https://doi.org/10.1109/TSMCA.2008.2007982>

- [41] Iannino, A., & Musa, J. D. (1990). “Software Reliability”. *Advances in Computers*, **30**(C). [https://doi.org/10.1016/S0065-2458\(08\)60299-5](https://doi.org/10.1016/S0065-2458(08)60299-5)
- [42] Ishii, T., & Dohi, T. (2008). “A New Paradigm for Software Reliability Modelling-From NHPP to NHGP”. *International Symposium on Dependable Computing*, Taipei, pp. 224–231. <https://doi.org/10.1109/PRDC.2008.24>
- [43] Jabeen, G., Yang, X., Ping, L., Rahim, S., Sahar, G., & Shah, A. A. (2018). “Hybrid Software Reliability Prediction Model Based on Residual Errors”. *International Conference on Software Engineering and Service Sciences*, Beijing, pp. 479–482. <https://doi.org/10.1109/ICSESS.2017.8342959>
- [44] Jin, C. (2011). “Software Reliability Prediction based on Support Vector Regression using a Hybrid Genetic Algorithm and Simulated Annealing Algorithm”. *IET Software*, **5**(4), pp. 398–405. <https://doi.org/10.1049/iet-sen.2010.0073>
- [45] Jin, Cong, & Jin, S. W. (2016). “Parameter Optimization of Software Reliability Growth Model with S-Shaped Testing-Effort Function using Improved Swarm Intelligent Optimization”. *Applied Soft Computing Journal*, **40**, pp. 283–291. <https://doi.org/10.1016/j.asoc.2015.11.041>
- [46] Kalaivani, K., & Somsundaram, S. (2014). “A Hybrid GA - PSO Approach for Reliability Under Type II Censored Data using Exponential Distribution”. *International Journal of Mathematical Analysis*, **8**(49–52), pp. 2481–2492. <https://doi.org/10.12988/ijma.2014.49286>
- [47] Kamilaris, A., Kartakoullis, A., & Prenafeta-Boldú, F. X. (2017). “A Review on the Practice of Big Data Analysis in Agriculture”. *Computers and Electronics in Agriculture*, **143**(October), pp. 23–37. <https://doi.org/10.1016/j.compag.2017.09.037>
- [48] Kaur, K., & Anand, S. (2013). “Review on Software and Hardware Reliability and Metrics”. *International Journal of Science, Engineering and Technology Research*, **2**(5), pp. 1108–1110.
- [49] Kemp, G., Vargas-solar, G., Ferreira, C., Collet, C., Kemp, G., Vargas-solar, G., Ferreira, C., Ghodous, P., & Collet, C. (2016). *Service Oriented Big Data*

Management for Transport, Communications in Computer and Information Science, Springer, Cham.

- [50] Khurshid, S., Shrivastava, A. K., & Iqbal, J. (2021). “Effort Based Software Reliability Model with Fault Reduction Factor, Change Point and Imperfect Debugging”. *International Journal of Information Technology (Singapore)*, **13**(1), pp. 331–340. <https://doi.org/10.1007/s41870-019-00286-x>
- [51] Kumar, Anurag, Tripathi, R. P., Saraswat, P., & Gupta, P. (2018). “Parameter Estimation of Software Reliability Growth Models using Hybrid Genetic Algorithm”. *International Conference on Image Information Processing*, IEEE, Shimla, pp. 317–322. <https://doi.org/10.1109/ICIIP.2017.8313732>
- [52] Kumar, Archana, & Kapur, P. K. (2009). “SRGMs based on Stochastic Differential Equations”. *International Conference on Communication Theory, Reliability and Quality of Service*, Colmar, pp. 1–7. <https://doi.org/10.1109/CTRQ.2009.26>
- [53] Kwon, O., Lee, N., & Shin, B. (2014). “Data Quality Management, Data Usage Experience and Acquisition Intention of Big Data Analytics”. *International Journal of Information Management*, **34**(3), pp. 387–394. <https://doi.org/10.1016/j.ijinfomgt.2014.02.002>
- [54] Lee, D. H., Chang, I. H., Pham, H., & Song, K. Y. (2018). “A Software Reliability Model Considering the Syntax Error in Uncertainty Environment, Optimal Release Time, and Sensitivity Analysis”. *Applied Sciences*, **8**(9), pp. 1483. <https://doi.org/10.3390/app8091483>
- [55] Lee, D. H., Hong Chang, I., & Pham, H. (2020). “Software Reliability Model with Dependent Failures and SPRT”. *Mathematics*, **8**(8), pp. 1436. <https://doi.org/10.3390/MATH8081366>
- [56] Li, J., He, Y., & Ma, Y. (2017). “Research of Network Data Mining Based on Reliability Source under Big Data Environment”. *Neural Computing and Applications*, **28**, pp. 327–335. <https://doi.org/10.1007/s00521-016-2349-x>
- [57] Li, J.-S., Zhang, Y.-F., & Tian, Y. (2016). “Medical Big Data Analysis in Hospital Information System”. *Big Data on Real-World Applications*, IntechOpen, pp. 65, <https://doi.org/10.5772/63754>

- [58] Li, M., Sadoughi, M., Hu, Z., & Hu, C. (2020). "A Hybrid Gaussian Process Model for System Reliability Analysis". *Reliability Engineering and System Safety*, **197**(February), pp. 106816. <https://doi.org/10.1016/j.ress.2020.106816>
- [59] Li, P. L. (2005). "Forecasting Field Defect Rates Using a Combined Time-based and Metrics-based Approach : a Case Study of OpenBSD". *International Symposium on Software Reliability Engineering*, IEEE, Chicago, pp. 10
- [60] Li, Q., Li, Y., Gao, J., Zhao, B., Fan, W., & Han, J. (2014). "Resolving Conflicts in Heterogeneous Data by Truth Discovery and Source Reliability Estimation". *International Conference on Management of Data*, Association for Computing Machinery, New York, pp. 1187–1198. <https://doi.org/10.1145/2588555.2610509>
- [61] Li, Z., Yu, M., Wang, D., & Wei, H. (2019). "Using Hybrid Algorithm to Estimate and Predicate Based on Software Reliability Model". *IEEE Access*, **7**(c), pp. 84268–84283. <https://doi.org/10.1109/ACCESS.2019.2917828>
- [62] Lin, C. T., & Huang, C. Y. (2008). "Enhancing and Measuring the Predictive Capabilities of Testing Effort Dependent Software Reliability Models". *Journal of Systems and Software*, **81**(6), pp. 1025–1038. <https://doi.org/10.1016/j.jss.2007.10.002>
- [63] Littlewood, B. (1984). "Rationale for a Modified Duane Model". *IEEE Transactions on Reliability*, **33**(2), pp. 157-159.
- [64] Liu, C., Chen, J., Yang, L. T., Zhang, X., Yang, C., Ranjan, R., & Rao, K. (2014). "Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine Grained Updates". *IEEE Transactions on Parallel and Distributed Systems*, **25**(9), pp. 2234–2244. <https://doi.org/10.1109/TPDS.2013.191>
- [65] Liu, Y., & Gao, Y. (2009). "Automation Software Reliability Model Selection Based on Unascertained Set". *International Conference on Information Management, Innovation Management and Industrial Engineering*, IEEE, Xian, **4**, pp. 643–646. <https://doi.org/10.1109/ICIII.2009.614>

- [66] Lo, J. H. (2012). “A Data-Driven Model for Software Reliability Prediction”. *International Conference on Granular Computing*, IEEE, Hangzhou pp. 326–331. <https://doi.org/10.1109/GrC.2012.6468581>
- [67] Lohmor, S., & Sagar, B. B. (2017). “Estimating the Parameters of Software Reliability Growth Models Using Hybrid DEO-ANN Algorithm”. *International Journal of Enterprise Network Management*, **8**(3), pp. 247–269. <https://doi.org/10.1504/IJENM.2017.087437>
- [68] Lyu M. R. (1996), “Handbook of Software Reliability Engineering”, McGraw-Hill. ISBN:0-07-039400-8
- [69] Majumdar, J., Naraseeyappa, S., & Ankalaki, S. (2017). “Analysis of Agriculture Data using Data Mining Techniques: Application of Big Data”. *Journal of Big Data*, **4**(1). <https://doi.org/10.1186/s40537-017-0077-4>
- [70] Manjula, C., & Florence, L. (2019). “Deep Neural Network based Hybrid Approach for Software Defect Prediction using Software Metrics”. *Cluster Computing*, **22**, pp. 9847–9863. <https://doi.org/10.1007/s10586-018-1696-z>
- [71] Meeker, W. Q., & Hong, Y. (2014).” Reliability Meets Big Data: Opportunities and Challenges”. *Quality Engineering*, **26**(1), pp. 102–116. <https://doi.org/10.1080/08982112.2014.846119>
- [72] Miglani, N., & Rana, P. (2011). “Ranking of Software Reliability Growth Models using Greedy Approach”, *Global Journal of Business Management and Information Technology*, **1**(2), pp. 119–124.
- [73] Miller, D. R. (1986). “Exponential Order Statistic Models of Software Reliability Growth”. *IEEE Transactions on Software Engineering*, **SE-12**(1), pp. 12–24. <https://doi.org/10.1109/TSE.1986.6312915>
- [74] Mohanty, R., Ravi, V., & Patra, M. R. (2013). “Hybrid Intelligent Systems for Predicting Software Reliability”. *Applied Soft Computing Journal*, **13**(1), pp. 189–200. <https://doi.org/10.1016/j.asoc.2012.08.015>
- [75] Muthulakshmi, P., & Udhayapriya, S. (2018). “A Survey on Big Data Issues and Challenges”. *International Journal of Computer Sciences and Engineering*, **6**(6), pp. 1238–1244. <https://doi.org/10.26438/ijcse/v6i6.12381244>

- [76] Muthuraj, R. J., & Loganathan, A. (2017), Importance of Environmental Factors Affecting Software Reliability”. *Global and Stochastic Analysis*, **5**(4), pp. 1350–1355.
- [77] Nachiappan, R., Javadi, B., Calheiros, R. N., & Matawie, K. M. (2017). “Cloud Storage Reliability for Big Data Applications: A state of the art survey”. *Journal of Network and Computer Applications*, **97**, pp. 35–47. <https://doi.org/10.1016/j.jnca.2017.08.011>
- [78] Naganathan, V. (2018). “Comparative Analysis of Big Data, Big Data Analytics: Challenges and Trends”. *International Research Journal of Engineering and Technology*, **5**(5), pp. 1948–1964.
- [79] Ohishi, K., Okamura, H., & Dohi, T. (2009). “Gompertz Software Reliability Model: Estimation Algorithm and Empirical Validation”. *Journal of Systems and Software*, **82**(3), pp. 535–543. <https://doi.org/10.1016/j.jss.2008.11.840>
- [80] Okumoto, J. D. M. and K. (1983). “A Logarithmic Poisson Execution Time Model for Software Reliability Measurement”. *International Conf. on Software Engineering*, Whippany, pp. 230–237.
- [81] Osaki, S. (1984). “S-Shaped Software Reliability Growth Models and Their Applications”. *IEEE Transactions on Reliability*, **R-33**(4), pp. 289–292. <https://doi.org/10.1109/TR.1984.5221826>
- [82] Pai, P. F., & Lin, K. P. (2006). “Application of Hybrid Learning Neural Fuzzy Systems in Reliability Prediction”. *Quality and Reliability Engineering International*, **22**(2), pp. 199–211. <https://doi.org/10.1002/qre.696>
- [83] Pati, J., & Shukla, K. K. (2015). “A Hybrid Technique for Software Reliability Prediction”. *ACM International Conference Proceeding Series*, ACM, Bangalore, pp. 139–146. <https://doi.org/10.1145/2723742.2723756>
- [84] Pence, J., Mohaghegh, Z., Ostroff, C., Dang, V., Kee, E., Hubenak, R., & Billings, M. A. (2015). “Quantifying Organizational Factors in Human Reliability Analysis using The Big Data Theoretic Algorithm”. *International Topical Meeting on Probabilistic Safety Assessment and Analysis*, ANS, Sun Valley, **2**, pp. 650–659.

- [85] Peng, R., Hu, Q. P., & Ng, S. H. (2008, September). "Incorporating Fault Dependency and Debugging Delay in Software Reliability Analysis". *International Conference on Management of Innovation and Technology*, IEEE, Bangkok, pp. 641-645.
- [86] Pham, H. (2003). "Software Reliability and Cost Models: Perspectives, Comparison, and Practice". *European Journal of Operation Research*, **149**, pp. 475–489.
- [87] Pham, H., Nordmann, L., & Zhang, X. (1999). "A General Imperfect Software-Debugging Model with S-Shaped Fault Detection Rate". *IEEE Transactions on Reliability*, **48**(2), pp. 169–175. <https://doi.org/10.1109/24.784276>.
- [88] Pietrantuono, R., Russo, S., & Trivedi, K. S. (n.d.). "Online Reliability Monitoring : A Hybrid Approach". *Proactive Failure Avoidance, Recovery and Maintenance (PFARM) Summary of DSN Workshop at Estoril, Portugal*, pp. 3–8.
- [89] Pillai, K., & Sukumaran Nair, V. S. (1997). "A model for Software Development Effort and Cost Estimation". *IEEE Transactions on Software Engineering*, **23**(8), pp. 485–497. <https://doi.org/10.1109/32.624305>
- [90] Pillar, V. D. (1997). "Multivariate Exploratory Analysis and Randomization Testing with MULTIV. *Coenoses*, Springer, **12**(January), pp. 145–148.
- [91] Prakash, C. J., Rohini, P. M., Ganesh, R. B., & Maheswari, V. (2013). "Hybrid Reliability Model to Enhance the Efficiency of Composite Web Services". *International Conference on Emerging Trends in Computing, Communication and Nanotechnology*, IEEE, Tirunelveli, pp. 79–83. <https://doi.org/10.1109/ICE-CCN.2013.6528468>
- [92] Protopop, I., & Shanoyan, A. (2016). "Big Data and Smallholder Farmers: Big Data Applications in the Agri-Food Supply Chain in Developing Countries". *International Food and Agribusiness Management Review*, **19**, pp. 173–190.
- [93] Pushphavathi, T. P., Suma, V., & Ramaswamy, V. (2014). "A Novel Method for Software Defect Prediction: Hybrid of FCM and Random Forest". *International Conference on Electronics and Communication Systems*, IEEE, Coimbatore, pp. 1–5. <https://doi.org/10.1109/ECS.2014.6892743>

- [94] Qi, Y. D., Ying, L., Ning, Q., & Xie, X. F. (2010). "A BP Neural Network based Hybrid Model for Software Reliability Prediction". *International Conference on Computer Application and System Modelling*, IEEE, Taiyuan, pp. 11–14. <https://doi.org/10.1109/ICCASM.2010.5622531>
- [95] Radmehr, E., & Bazmara, M. (2017). "A Survey on Business Intelligence Solutions in Banking Industry and Big Data Applications". *International Journal of Mechatronics*, **7**(23), pp. 3280–3298.
- [96] Rafi, S. M., & akthar, shaheda. (2010). "Software Reliability Growth Model with Gompertz TEF and Optimal Release Time Determination by Improving the Test Efficiency". *International Journal of Computer Applications*, **7**(11), pp. 34–43. <https://doi.org/10.5120/1337-1741>
- [97] Raghuvanshi, K. K., Agarwal, A., Jain, K., & Singh, V. B. (2021). "A Time-Variant Fault Detection Software Reliability Model". *SN Applied Sciences*, **3**(1), pp. 1–10. <https://doi.org/10.1007/s42452-020-04015-z>
- [98] RahamanSk, A., RajeshK, S., Rani, G. K., Professor, A., & Scholar, R. (2018). "Challenging tools on Research Issues in Big Data Analytics". *International Journal of Engineering Development and Research*, **6**(1), pp. 2321–9939.
- [99] Raj Kiran, N., & Ravi, V. (2008). "Software Reliability Prediction by Soft Computing Techniques". *Journal of Systems and Software*, **81**(4), pp. 576–583. <https://doi.org/10.1016/j.jss.2007.05.005>
- [100] Rao, K. M., & Anuradha, K. (2016). "A Hybrid Method for Parameter Estimation of Software Reliability Growth Model using Modified Genetic Swarm Optimization with the Aid of Logistic Exponential Testing Effort Function". *International Conference on Research Advances in Integrated Navigation Systems*, IEEE, Bangalore, pp. 1-8. <https://doi.org/10.1109/RAINS.2016.7764400>
- [101] Reis, G. A., Chang, J., Vachharajani, N., Rangan, R., August, D. I., & Mukherjee, S. S. (2005). "Design and Evaluation of Hybrid Fault Detection Systems". *International Symposium on Computer Architecture*, IEEE, Madison, pp. 148–159. <https://doi.org/10.1109/ISCA.2005.21>
- [102] Roy, P., Mahapatra, G. S., & Dey, K. N. (2015). "Neuro Genetic Approach on Logistic Model based Software Reliability Prediction". *Expert Systems with*

- Applications, **42**(10), pp. 4709–4718.
<https://doi.org/10.1016/j.eswa.2015.01.043>
- [103] Sahu, K., & Srivastava, R. K. (2019). “Revisiting software reliability”. *Advances in Intelligent Systems and Computing*, **808**, pp. 221–235.
https://doi.org/10.1007/978-981-13-1402-5_17
- [104] Salahaldeen, J., & Marwan, M. (2017). “The Use of Original and Hybrid Grey Wolf Optimizer in Estimating the Parameters of Software Reliability Growth Models”. *International Journal of Computer Applications*, **167**(3), pp. 12–21.
<https://doi.org/10.5120/ijca2017914201>
- [105] Sangeeta, & Sitender. (2020). “Comprehensive Analysis of Hybrid Nature-Inspired Algorithms for Software Reliability Analysis”. *Journal of Statistics and Management Systems*, **23**(6), pp. 1037–1048.
<https://doi.org/10.1080/09720510.2020.1814498>
- [106] Sangeeta, Sharma, K., & Bala, M. (2020). “An Ecological Space Based Hybrid Swarm-Evolutionary Algorithm for Software Reliability Model Parameter Estimation”. *International Journal of System Assurance Engineering and Management*, **11**(1), pp. 77–92. <https://doi.org/10.1007/s13198-019-00926-2>
- [107] Shakya, S., & S., S. (2020). “Reliable Automated Software Testing Through Hybrid Optimization Algorithm”. *Journal of Ubiquitous Computing and Communication Technologies*, **2**(3), pp. 126–135.
<https://doi.org/10.36548/jucct.2020.3.002>
- [108] Sharma K., Garg R., Nagpal C. K. and Garg R. K.(2010). “Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach.” *IEEE Transactions on Reliability*, **59**(2) , pp. 266-276, doi: 10.1109/TR.2010.2048657
- [109] Sharma, D., & Chandra, P. (2019). “A comparative analysis of soft computing techniques in software fault prediction model development”. *International Journal of Information Technology*, **11**(1), pp. 37–46.
<https://doi.org/10.1007/s41870-018-0211-3>
- [110] Sharma, T. K. (2018). “Estimating Software Reliability Growth Model Parameters using Opposition-Based Shuffled Frog-Leaping Algorithm”. *Soft*

- Computing Applications*, Springer, Singapore, pp. 149-164.
https://doi.org/10.1007/978-981-10-8049-4_8
- [111] Sharma, T. K., Pant, M., & Abraham, A. (2011). “Dichotomous Search in ABC and its Application in Parameter Estimation of Software Reliability Growth Models”. *World Congress on Nature and Biologically Inspired Computing*, IEEE, Salamanca, pp. 207–212. <https://doi.org/10.1109/NaBIC.2011.6089460>
- [112] Sivarajah, U., Kamal, M. M., Irani, Z., & Weerakkody, V. (2017). “Critical Analysis of Big Data Challenges and Analytical Methods”. *Journal of Business Research*, **70**, pp. 263–286. <https://doi.org/10.1016/j.jbusres.2016.08.001>
- [113] Spichkova, M., Schmidt, H. W., Yusuf, I. I., Thomas, I. E., Androulakis, S., & Meyer, G. R. (2016). “Towards Modelling and Implementation of Reliability and Usability Features for Research Oriented Ccloud Computing Platforms”. *Communications in Computer and Information Science*, **703**, pp. 158–178. https://doi.org/10.1007/978-3-319-56390-9_8
- [114] Sudharson, D., & Dr, P. (2020). Hybrid Software Reliability Model with Pareto Distribution and Ant Colony Optimization (PD-ACO)”. *International Journal of Intelligent Unmanned Systems*, **8**(2), pp. 129–140. <https://doi.org/10.1108/IJIUS-09-2019-0052>
- [115] Sun, H., Wu, J., Wu, J., & Yang, H. (2019). Hybrid SVM and ARIMA Model for Failure Time Series Prediction based on EEMD”. *International Journal of Performability Engineering*, **15**(4), 1161–1170. <https://doi.org/10.23940/ijpe.19.04.p11.11611170>
- [116] Tamura, Y., & Yamada, S. (2014). “Reliability Analysis Based on AHP and Software Reliability Models for Big Data on Cloud Computing”. *International Journal of Statistics–Theory and Applications* **1**(1), pp. 43–49.
- [117] Tamura, Y., & Yamada, S. (2015a). “Reliability Analysis Based on a Jump Diffusion Model with Two Wiener Processes for Cloud Computing with Big Data”. *Entropy*, **17**(7), pp. 4533–4546. <https://doi.org/10.3390/e17074533>
- [118] Tamura, Y., & Yamada, S. (2015b). “Software Reliability Analysis Considering the Fault Detection Trends for Big Data on Cloud Computing”. *Lecture Notes in*

- Electrical Engineering, **349**, pp. 1021–1030. https://doi.org/10.1007/978-3-662-47200-2_106
- [119] Tamura, Y., & Yamada, S. (2015c). “Software Reliability Assessment Tool Based on Fault Data Clustering and Hazard Rate Model Considering Cloud Computing with Big Data”. *International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions*, IEEE, Noida, pp. 1–6. <https://doi.org/10.1109/ICRITO.2015.7359208>
- [120] Tamura, Y., & Yamada, S. (2017a). “Dependability Analysis Tool Based on Multi-Dimensional Stochastic Noisy Model for Cloud Computing with Big Data”. *International Journal of Mathematical, Engineering and Management Sciences*, **2**(4), pp. 273–287. <https://doi.org/10.33889/ijmems.2017.2.4-021>
- [121] Tamura, Y., & Yamada, S. (2017b). “Fault Identification and Reliability Assessment Tool Based on Deep Learning for Fault Big Data”. *Software Networking*, **1**, pp. 161–167. <https://doi.org/10.13052/jsn2445-9739.2017.008>
- [122] Tamura, Y., Nobukawa, Y., & Yamada, S. (2016). “A Method of Reliability Assessment Based on Neural Network and Fault Data Clustering for Cloud with Big Data”. *International Conference on Information Science and Security*, IEEE, Seoul, pp. 4–7. <https://doi.org/10.1109/ICISSEC.2015.7370965>
- [123] Tamura, Y., Takeuchi, T., & Yamada, S. (2017). “Software Reliability and Cost Analysis Considering Service User for Cloud with Big Data”. *International Journal of Reliability, Quality and Safety Engineering*, **24**(2), pp. 1–14. <https://doi.org/10.1142/S0218539317500097>
- [124] Vaitsis, C., Hervatis, V., & Zary, N. (2016). “Introduction to Big Data in Education and Its Contribution to the Quality Improvement Processes”. *Big Data on Real-World Applications*, In Tech, pp. 41-64, Chap. 3. <https://doi.org/10.5772/63896>
- [125] Vamsidhar, Y., Raju, P. S. S., & Kumar, T. R. (2012). “Performance Analysis of Reliability Growth Models using Supervised Learning Techniques”. *International Journal of Scientific & Technology Research*, **1**(1), pp. 1–7.
- [126] Viswanathan, A. S., & Ramani, S. (2018). “Algorithm and Matlab Program for Software Reliability Growth Model Based on Weibull Order Statistics

- Distribution”. *International Journal of Advanced Scientific Research and Management*, **3**(11), pp. 0–4.
- [127] Wang, J., Wu, H., & Wang, R. (2017). “A New Reliability Model in Replication-Based Big Data Storage Systems”. *Journal of Parallel and Distributed Computing*, **108**, pp. 14–27. <https://doi.org/10.1016/j.jpdc.2017.02.001>
- [128] Wang, R., Jin, F., Yang, L., & Han, X. (2018). “A Software Reliability Combination Model Based on Genetic Optimization BP Neural Network”. *Communications in Computer and Information Science*, Springer, Singapore, pp. https://doi.org/10.1007/978-981-13-0896-3_15
- [129] Weersink, A., Fraser, E., Pannell, D., Duncan, E., & Rotz, S. (2018). “Opportunities and Challenges for Big Data in Agricultural and Environmental Analysis”. *Annual Review of Resource Economics*, **10**(2018), pp. 19–37. <https://doi.org/10.1146/annurev-resource-100516-053654>
- [130] Wood, A. (1996). “Software Reliability Growth Model: Primary Failures Generate Secondary Faults under Imperfect Debugging”. *IEEE Transactions on Reliability*, **43**(3) pp. 408–413, <http://linkinghub.elsevier.com/retrieve/pii/0026271496819585>
- [131] Xiang, Z., Du, Q., Ma, Y., & Fan, W. (2018). Assessing Reliability of Social Media Data: Lessons from Mining TripAdvisor Hotel Reviews”. *Information Technology and Tourism*, **18**, pp. 43–59. <https://doi.org/10.1007/s40558-017-0098-z>
- [132] Xie, M. (1991). *Software Reliability modelling*, World Scientific, Singapore.
- [133] Xie, M. (1995). *Software Reliability Models for Practical Applications*. *Software Quality and Productivity*, Springer, Boston, pp. 211–214. https://doi.org/10.1007/978-0-387-34848-3_32
- [134] Xie, Min, Hong, G. Y., & Wohlin, C. (2003). Modelling and Analysis of Software System Reliability. *Case Studies in Reliability and Maintenance*, Wiley, pp. 233–250, Chap. 10. <https://doi.org/10.1002/0471393002.ch10>
- [135] Xuejie, Z., Zhijian, W., & Feng, X. (2013). “Reliability Evaluation of Cloud Computing Systems Using Hybrid Methods”. *Intelligent Automation and Soft Computing*, **19**(2), pp. 165–174. <https://doi.org/10.1080/10798587.2013.786969>

- [136] Yadav, H. B., & Yadav, D. K. (2017). “Early Software Reliability Analysis using Reliability Relevant Software Metrics”. *International Journal of Systems Assurance Engineering and Management*, **8**(1992), pp. 2097–2108. <https://doi.org/10.1007/s13198-014-0325-3>
- [137] Yaghoobi, T. (2020). “Parameter Optimization of Software Reliability Models using Improved Differential Evolution Algorithm”. *Mathematics and Computers in Simulation*, **177**, 46–62. <https://doi.org/10.1016/j.matcom.2020.04.003>
- [138] Yamada, S., & Osaki, S. (1985). “Software Reliability Growth Modelling: Models and Applications”. *IEEE Transactions on Software Engineering*, **SE-11**(12), pp. 1431–1437. <https://doi.org/10.1109/TSE.1985.232179>
- [139] Yamada, S., Hishitani, J., & Osaki, S. (1993). “Software-Reliability Growth with a Weibull Test-Effort: A Model & Application”. *IEEE Transactions on Reliability*, **42**(1), pp. 100–106. <https://doi.org/10.1109/24.210278>
- [140] Yamada, S., Ohba, M., & Osaki, S. (1983). “S-Shaped Reliability Growth Modelling for Software Error Detection”. *IEEE Transactions on Reliability*, **R-32**(5), pp. 475–484. <https://doi.org/10.1109/TR.1983.5221735>
- [141] Yamada, S., Ohtera, H., & Narihisa, H. (1986). “Software Reliability Growth Models with Testing-Effort”. *IEEE Transactions on Reliability*, **35**(1), pp. 19–23. <https://doi.org/10.1109/TR.1986.4335332>
- [142] Yamada, S., Tokuno, K., & Osaki, S. (1992). “Imperfect Debugging Models with Fault Introduction Rate for Software Reliability Assessment”. *International Journal of Systems Science*, **23**(12), pp. 2241–2252. <https://doi.org/10.1080/00207729208949452>
- [143] Yan, J., Meng, Y., Lu, L., & Li, L. (2017). “Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance”. *IEEE Access*, **5**(c), pp. 23484–23491. <https://doi.org/10.1109/ACCESS.2017.2765544>
- [144] Yang, L., Li, Z., Wang, D., Miao, H., & Wang, Z. (2021). “Software Defects Prediction Based on Hybrid Particle Swarm Optimization and Sparrow Search Algorithm”. *IEEE Access*, **9**, pp. 60865–60879. <https://doi.org/10.1109/ACCESS.2021.3072993>

- [145] Yanyan, Z., & Renzuo, X. (2008). “An Adaptive Exponential Smoothing Approach for Software Reliability Prediction”. *International Conference on Wireless Communications, Networking and Mobile Computing*, IEEE, Dalian, pp.1–4. <https://doi.org/10.1109/WiCom.2008.2946>
- [146] Yaremchuk, S., & Kharchenko, V. (2018). “Big Data and Similarity Based Software Reliability Assessment: The Technique and Applied Tools”. *International Conference on Dependable Systems, Services and Technologies*, IEEE, Kyiv, pp. 485–490. <https://doi.org/10.1109/DESSERT.2018.8409182>
- [147] Zeng, G. (2015). “Application of Big Data in Intelligent Traffic System”. *IOSR Journal of Computer Engineering*, **17**(1), pp. 2278–2661. <https://doi.org/10.9790/0661-17160104>
- [148] Zhang, X., Teng, X., & Pham, H. (2003). “Considering Fault Removal Efficiency in Software Reliability Assessment”. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, **33**(1), pp. 114–120. <https://doi.org/10.1109/TSMCA.2003.812597>
- [149] Zhao, J., Liu, H. W., Cui, G., & Yang, X. Z. (2006). “Software Reliability Growth Model with Change Point and Environmental Function”. *Journal of Systems and Software*, **79**(11), pp. 1578–1587. <https://doi.org/10.1016/j.jss.2006.02.030>
- [150] Zhao, M., & Xie, M. (1992). “On the Log-Power NHPP Software Reliability Model”. *International Symposium on Software Reliability Engineering*, IEEE, Trangle Park, pp. 14–22. <https://doi.org/10.1109/ISSRE.1992.285862>
- [151] Zhen, L., Liu, Y., Dongsheng, W., & Wei, Z. (2020). “Parameter Estimation of Software Reliability Model and Prediction Based on Hybrid Wolf Pack Algorithm and Particle Swarm Optimization”. *IEEE Access*, **8**, pp. 29354–29369. <https://doi.org/10.1109/ACCESS.2020.2972826>
- [152] Zicari, R. V. *Big Data: Challenges and Opportunities*. <https://pdfs.semanticscholar.org/2d43/056cf8eff8f8d6e031f51f073187808676a3.pdf>. 5/3/2019.

Big data reliability: A critical review

Shalini Sharma*, Naresh Kumar and Kuldeep Singh Kaswan

School of Computing Science and Engineering, Galgotias University, Greater Noida, India

Abstract. Big data requires new technologies and tools to process, analyze and interpret the vast amount of high-speed heterogeneous information. A simple mistake in processing software, error in data, and malfunctioning in hardware results in inaccurate analysis, compromised results, and inadequate performance. Thus, measures concerning reliability play an important role in determining the quality of Big data. Literature related to Big data software reliability was critically examined in this paper to investigate: the type of mathematical model developed, the influence of external factors, the type of data sets used, and methods employed to evaluate model parameters while determining the system reliability or component reliability of the software. Since the environmental conditions and input variables differ for each model due to varied platforms it is difficult to analyze which method gives the better prediction using the same set of data. Thus, paper summarizes some of the Big data techniques and common reliability models and compared them based on interdependencies, estimation function, parameter evaluation method, mean value function, etc. Visualization is also included in the study to represent the Big data reliability distribution, classification, analysis, and technical comparison. This study helps in choosing and developing an appropriate model for the reliability prediction of Big data software.

Keywords: Reliability models, Big data, stochastic equation, hazard rate, jump diffusion

1. Introduction

Because of rapid change in the volume, velocity, and veracity of Big data new kinds of storage and analytical methods are required [1]. Organizations nowadays thrive upon the usage of Big data. Due to the advancement of technology bulk of data is generated at an amazingly fast speed every second, which is to be stored and processed to gain useful information. Recently Big data is being accumulated and utilized in many domains [2] like agriculture [3], forestry monitoring [4], energy management [5, 6], medical [7], city management [8], operations management [9, 10], social networks [11], etc.

The main challenges associated with Big data are data quality and its maintenance. Business houses might lose sales, time, and margins due to inadequate data and incorrect interpretation [12]. The

rapidly changing nature of Big data requires a new type of technologies and analytical methods [13] for fast retrieval and storage so that pertaining information can be extracted and utilized for decision making [14]. Big data technologies like Hadoop, Cassandra, HDFS, MapReduce, NoSQL, MongoDB, HIVE, PIG, and HBASE work together to extract meaningful information from data [15], whereas the performance of an analytical method depends on the process, which visualizes data after acquisition and interpretation. Estimating the performance of a process helps in taking corrective measures before it is too late, and the problem deteriorates [16]. The major issue in Big data is reliability assessment of software utilizing Big data. Software fault is concerned primarily with process, design, and data. Hardware fault results due to wear-out, faulty design, and inappropriate environmental or operational conditions which leads to software failure, compromising the reliability of the system. Reliability of software is defined as “the probability that it will function failure-free for a specified period under specific conditions” [17].

*Corresponding author. Shalini Sharma, School of Computing Science and Engineering, Galgotias University, Greater Noida, India. E-mail: shalinisharma@kalindi.du.ac.in.

Failure of software is generally predicted and estimated with the help of software reliability models [18]. Software Reliability Growth Models (SRGM's) describe the phenomenon of software failure or fault detection in the testing phase.

Many reliability assessment methods for quality control of the software or testing process have been proposed by various researchers. Also, reliability is the foremost concern in Open System Software (OSS) because of its extensive use due to reduced cost, easy accessibility, and quick delivery. Cloud computing enables a user to perform complicated, highly parallel, data-intensive, and long-running experiments that are too complicated to run on desktop machines [19]. Mobile software, Application software, and Cloud computing source code are open to all, therefore the security and reliability of OSS [20, 21] become a major problem. This paper contributes to providing the analysis of literature related to reliability concerning Big data.

Various features of software reliability models (developed by different researchers) for predicting the reliability in Big data scenario were investigated. Different criteria like interdependencies, parameter estimation, mode of working, datasets used, etc. are used for comparing these models. For the critical review of Big data reliability, several papers related to Big data, Big data analytics, and Big data reliability were studied. Papers were then classified based on reliability in hardware, data reliability, and software reliability. Literature is further pursued to determine the interdependencies in Big data software reliability as shown in Fig. 1. Since research in Big data is a contemporary topic the research papers sought were mostly from acclaimed journals and conferences with publication years ranging from 2012 to 2019.

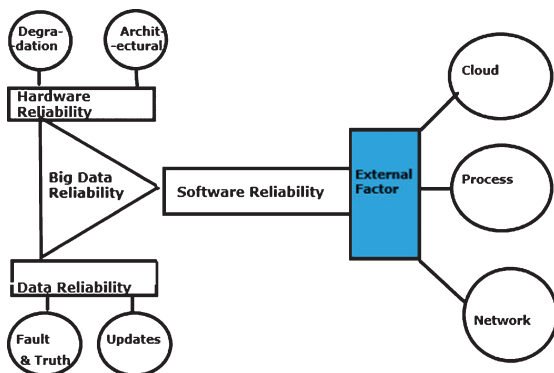


Fig. 1. Interdependencies in Big data Reliability.

2. Related work

Big data requires the use of new techniques to process large data sets, various such techniques which are generally used to get information from data are discussed in section 2.1. In Section 2.2 some traditional software reliability models are discussed. These reliability models are not accurate enough to predict the Big data reliability because of its characteristics and interdependencies. Thus, the need to develop new models that can predict the Big data reliability better than these traditional models arises. Reliability models that incorporated the external factors affecting the reliability of Big software are discussed in Section 2.3.

2.1. Big data

Big data is a gigantic amount of data that is difficult to store, process and manage, using traditional RDBMS due to its characteristics. In RDBMS, all data is related, and losing a small part of data affects other data in significant ways contrary to Big data where we employ methods that knowingly lose a fraction of data for the smooth functioning of unstructured high-velocity voluminous data in a short span. Nowadays Big data is generated in various forms like text-sound, image, voice, and video. It is exceedingly difficult to handle these types of data sets using traditional data processing applications. Also due to unstructured and scattered file system, data analysis is a challenging task, and we require some new techniques like Genetic Algorithms(GA), Neural Networks(NN), Sentiment Analysis, Network Analysis, Machine Learning(ML), etc. and tools like Apache Hadoop, Apache Storm, Apache Spark, Mongo DB, NoSQL HPCC to handle Big data. A summary of some techniques is given below.

2.1.1. Associative rule learning

It is a method to find out the correlation between database variables. It looks for patterns in a database to uncover new relationships. Associative rules can predict a variable instead of class, giving the flexibility to predict the combination of attributes. It uses a machine learning model and identifies the if-then association, called association rules. If and then are two parts of association rule called antecedents and consequent respectively which represents the frequent co-occurrence of items in the dataset [22]. Different association rules predict different things and specify different irregularities in a dataset.

Coverage represents the number of instances for which prediction is correct and accuracy represents the number of correctly predicted instances.

2.1.2. Classification tree analysis

A new observation is categorized using this method. It is a common technique used in data mining to classify new data sets depending upon the historical training data. Classification of the dependent variable is based on the forecast variable, the nodes representing data are divided into smaller subgroups and the output is a tree structure with an association between nodes [22]. In Machine Learning (ML) supervised learning and unsupervised learning are nothing but classification and clustering, respectively.

2.1.3. Regression analysis

It is a statistical method that determines the relationship between two variables, dependent often called outcome variable and independent called predictor variable. Independent variables influence the dependent variable which is yet to be predicted. Linear Regression specifies a linear relationship, whereas Nonlinear Regression which specifies a nonlinear relationship is used for complicated datasets [24]. Regression can be used for prediction as well as forecasting [23].

2.1.4. Sentiment analysis

Sentiment Analysis is contextual text mining which determines whether a piece of text is negative, positive, or neutral. Sentiment Analysis is generally used in web mining, data mining, and social media analytics. It deals with the emotions, feelings, as well as judgment responses generated from text. In Sentiment Analysis, we need to find an appropriate dataset and discover opinions while classifying and categorizing the opinion they convey. Machine Learning and Natural Language Processing techniques are combined in sentiment analysis of the text, to assign weighted sentiment scores to topics, themes categories, and entries in a sentence [25].

2.1.5. Genetic algorithms

Genetic Algorithms are based on the principles of Genetics. They provide optimal or near-optimal solutions to problems that might take a lifetime to be solved. In Machine learning these are frequently used to solve optimization problems, where optimization refers to maximizing or minimizing the objective function by varying the input values [26]. The set of all possible values constitute a search space and out of

these sets, there is one such value that gives the optimal solution. In Genetic Algorithms, there is a pool of possible solutions, by recombining and mutating these solutions we get new solutions called children. The process of mutation is repeated over many generations [26]. A fitness value is attached with each possible solution based on objective function value. According to the “Darwinian theory of survival of fittest” fitter individuals are assigned higher fitness value to yield more fitter individuals [27]. With better solutions, we keep evolving over generations till we reach the desired result or stop criteria.

2.1.6. Machine learning

Machine learning mainly focuses on the theory, performance, and properties of learning systems [28]. Machine learning provides systems to learn and improve by themselves through the experience without being programmed. It focuses on automatic learning by recognizing complex patterns and making intelligent decisions based on data [28]. It includes various software and makes predictions based on learned properties from training datasets. It is an application of AI which focuses on the development of self-learning programs for computers. For all applications of Big data, efficient infrastructure is required to support and develop efficient machine learning algorithms [29]. The aim is to make computers self-reliant to learn automatically without assistance. The learning process begins with the observations that look for specific data patterns to make better futuristic decisions based on providing examples [30]. Deep Learning, Feature Learning, Distributed Learning, Transfer Learning, and Active Learning are some of the advanced ML techniques proposed for Big data [31].

2.1.7. Neural networks

Neural Network is a set of algorithms that were developed to mimic the functioning of the human brain, when constructing an Artificial Neural Network designer must be cautious to choose the right size Network for the task [32]. Just like the human brain has neurons, a Neural Network has a node and sensory data is interpreted through machine perception, clustering, or labeling of raw input [30]. We train the Neural Network using an algorithm that compares the expected output and the desired one and adjusts the weight of nodes accordingly [33]. “Dr. Robert Hecht-Nielsen” defined the Neural Network as “The computing system made up of several simple, highly interconnected processing elements,

which process information by their dynamic state response to external inputs". Simple Neural Networks consist of an input layer, an output layer, and a hidden layer. The weighted sum of the input data is applied to a node which passes through its activation function, which acting as an on-off switch determines whether to let it pass or not and if yes to what extent, thereby assigning importance to inputs concerning the problem an algorithm is trying to solve [34]. A new model for neural-like networks called the Geometric Transformation model(GTM) provides solutions to the problems related to optimization, pattern recognition, lost data recovery, classification, etc. [72]. GTM provides the detailed analysis of the input data by replacing the slow training based on randomized input of Neural Network by high-speed training and can be used to construct the models using fractions and high degree polynomials [72].

2.1.8. Clustering techniques

In these techniques [35] data sets are termed as objects and we group or cluster the objects in such a way that objects having similarity belong in one group and dissimilar objects in another. An object is described either by a set of objects or by the relationship between the object and other objects [36]. Similarity is defined as the distance between two objects and the diameter of a cluster is the maximum distance between two objects in a cluster. Another measure of cluster quality is the centroid method which is defined as the average distance of each cluster object from the centroid of the cluster. The use of a clustering algorithm and appropriate distance function depends upon the type of datasets and intended use of the result. Cluster Analysis is an iterative process of discovering knowledge. It is often required to modify data pre-processing and parameters until the desired result is achieved with specific properties. It is generally used in many fields like Statistical Data Analysis, Data Mining, Pattern Recognition, Image Analysis, Information Retrieval, Data Compression, Machine Learning, Computer Graphics, and many more.

2.2. Reliability models

The stability of the software is determined by its reliability. Reliability is termed as the probability of smooth functioning of software without any failure under specified conditions for a given time. Software failure is nothing but a defect in the software product, which arises due to many factors like an error in soft-

ware code, a design defect, improper specifications. Software Failure mainly depends upon the number and type of errors probably encountered during a process, as the errors are discovered and corrected the reliability increases [37]. Reliability of software is assessed at various stages during development for a system to evaluate whether the laid down reliability requirements have been met or not. The reliability model generally used a mathematical equation that estimates the number of remaining errors during the debugging process of a software package [38]. The reliability of a system is analysed by using either prediction or estimation techniques, in both techniques fault data collected during the testing phase of the software is utilized to access reliability.

Software reliability modelling techniques observe and accumulate failure data and use statistical inference to analyse reliability [39]. Software reliability models try to understand how software fails and its cause, to form a random process that explains the behaviour of software failure with respect to time and attempt to quantify the reliability of software. There are almost 200 plus models that tried to quantify the reliability of software, but no one can be used universally for all conditions as they are application specific. The applicability and realism of assumptions of these models for accessing software reliability are still questionable [40].

Undermentioned are some most common reliability models for estimating software reliability.

2.2.1. The Jelinski-Moranda model [41]

In 1942 Jelinski-Moranda (J-M) developed a time between failure model using the following assumptions

- Initially, before testing n number of faults are there in code.
- Faults are equally likely and independent
- Removing a fault does not introduce any other fault
- Hazard rate $z(t)$ is proportional to the number of errors present in the system at a specified time t.

Thus, the likelihood of each fault causing failure is the same, and hazard rate $z(t)$ can be represented by a constant (θ). If $m(t)$ represents the total number of failures till time (t) then after removal of (i-1) fault from the system, we have the total number of faults as $n - m(t_{i-1})$ and the hazard rate of the software is:

$$z(\delta t/t_{i-1}) = \theta(n - m(t_{i-1})) \quad (1)$$

software reliability for this model is given as:

$$r(t_i) = e^{-\phi(N-(i-1)t} \quad (2)$$

2.2.2. The Goel-Okumoto model [42]

Goel-Okumoto (G-O) in 1979 proposed a Non-Homogeneous Poisson Process (NHPP) model using imperfect debugging as compared to perfect debugging of the J-M model. Failure counts at time t can be represented by a poison distribution with the mean value function $\mu(t)$. The failure count is a finite value n observed in infinite time. Following assumptions are made based on the model.

- Let \mathbf{u} represent undetected faults then the failure count is given by $n - \mu(t)$.
- Failure count is proportional to \mathbf{u} in the time interval $(t, t+\delta t)$.
- There is no correlation between the failure counts observed in the subsequent intervals $(0, t_1), (t_1, t_2), (t_2, t_3) \dots \dots (t_{n-1}, t_n)$.
- The hazard rate for each fault is constant and time-invariant.
- The process of removing faults upon detection of failure is instantaneous.

Above mentioned assumptions give mean

$$\text{value function : } \mu(t) = n(1 - e^{-\phi}) \quad (3)$$

and failure intensity function as:

$$\lambda(t) = N \phi e^{-\phi t} = \phi(N - \mu(t)) \quad (4)$$

for expected failure counts detected by time (t) [43].

2.2.3. Musa's basic execution time model [44]

In 1979 J.D. Musa developed a model based on execution time. The said model is easy to understand, simple, and accurately predicted the reliability using execution time which was later converted into calendar time. The failure behavior was given by the Non-Homogeneous Poisson Process. The model assumes that all observed faults are independent and are equally likely to occur. Piecewise exponentially distributed model was used to represent execution time failures. The rate of fault correction was proportional to the occurrence rate and failure intensity was proportional to the number of faults left.

If t represents the execution time and λ as failure intensity, then the mean failure experienced was given by

$$\mu(\bar{T}) = v_0 \cdot x(1 - \exp(-(\lambda_0 / v_0)\bar{T})) \quad (5)$$

Where λ_0 is **initial failure intensity**, v_0 is the total number of failures discovered over an infinite period,

Undermentioned is failure intensity function in terms of execution time and mean time μ [44]

$$\lambda(\bar{T}) = \lambda(\mu) = \lambda_0(1 - \mu / v_0) \quad (6)$$

$$\lambda(\bar{T}) = \lambda_0 \cdot x \cdot \exp(-(\lambda_0 / v_0)\bar{T}) \quad (7)$$

2.2.4. Littlewood – Verrall Bayesian model [45]

The models discussed so far assumes the availability of fault data and apply the Maximum Likelihood Estimation (MLE) statistical technique to estimate unknown but fixed parameters from the available data. In case when failure data is not available or sufficient, MLE is not a trustworthy technique as it can result in incorrect estimations [43]. Littlewood-Verrall Bayesian model [45], developed in 1973, is a Bayesian SRGM that considers reliability in the context of the detected number of faults and failure-free operation. Because of the non-availability of failure data, model parameters may be of a prior version or data based on history. The model assumes that the time between successive failures is exponential with parameters having gamma distribution. The hazard function in this model is an unknown constant having a Gamma distribution function with shape parameter α and scale parameter β .

2.2.5. Weibull model (Wm)

The Weibull function is extensively used in reliability due to its versatility. It is used both for software as well as hardware reliability estimation. Model is based on failure rate and is used quite extensively when the data indicate a monotone hazard function [46]. If on a logarithmic scale we plot failure rate versus cumulative testing time, then the model is valid for the plotted points if it gives an approximate straight line. Weibull model assumes that mean value function $m(t)$ is given by:

$$m(t) = (t / \alpha)^\beta \quad (8)$$

and occurrence of failure at time t is given by $\lambda(t)$ as:

$$\lambda(t) = dm(t)/d(t) = \beta / \alpha (t / \alpha)^{\beta-1} \quad [47] \quad (9)$$

The model can be estimated by using LSE or MLE estimations [39].

2.2.6. S-shaped model

Cumulative faults are s-shaped whereas the mean value function is exponential. The reason for the S-shaped [48] is explained: as generally the faults are dependent and are of the same size, even if we remove a fault at the initial stage it does not guarantee the decreased failure intensity because the software can still fail due to some other fault for the same test. Another reason for S-shape is explained by Yamada (1984) as testing software reliability is a learning process and with time the skill and effectiveness of test increases as one becomes familiar with test tools and software. Thus, S-shape implies that in the beginning when skill and test effectiveness is low the reliability is moderate but as the numbers of faults are discovered and corrected it increases quickly and gradually slows down. Several NHPP S-shaped models were proposed in the literature [49–53] but Delayed S-shaped and inflected S-shaped are the most interesting ones.

- **Delayed S-Shape:** The mean value and corresponding intensity function for Delayed S-shape two parameter curve is given as [47]:

$$m(t) = \alpha [1 - (1 + \beta \cdot t)e^{-\beta t}], \beta > 0 \quad (10)$$

$$\lambda(t) = \alpha \cdot \beta (1 + \beta \cdot t)e^{-\beta t} - \alpha \cdot \beta \cdot e^{-\beta t} = \alpha \cdot \beta^2 \cdot t \cdot e^{-\beta t} \quad (11)$$

where α =detected number of faults and β =failure rate. At time t expected remaining faults are given by

$$m(\infty) - m(t) = \alpha - \alpha [1 - (1 + \beta \cdot t)e^{-\beta t}] = \alpha (1 + \beta \cdot t)e^{-\beta t} \quad (12)$$

- **Inflection S-Shaped:** Basic assumption behind this model is that if the faults in a program are mutually dependent than the reliability growth curve become S shape [54].

The mean value is given by:

$$m(t) = a(1 - e^{-bt})/(1 + ce^{-bt}), b > 0, c > 0 \quad (13)$$

and intensity function is given by:

$$\lambda(t) = a \cdot b(1 + c)e^{-bt}/(1 + c \cdot e^{-bt})^2 \quad (14)$$

for this model [55]. Similarly, as in delayed S, a is detected number of faults while b is failure detection rate and c is inflection factor, respectively.

Expected number of remaining faults for inflected model is

$$m(\infty) - m(t) = a(1 + c)e^{-bt}/(1 + c \cdot e^{-bt}) \quad (15)$$

2.3. Reliability in big data

Because of Big data characteristics, the Big data software is equipped with number of advanced techniques to handle heterogeneous bulk storage, fast retrieval, and parallel processing. To determine the reliability of such Big software various techniques or methods were developed by researchers. Undermentioned is the brief insight into the work of eminent scholars working in this field.

2.3.1. Based on three-dimensional stochastic equations for big data on cloud, 2014: [56]

Researchers Yoshinobu Tamura, Kenta Miyaoka, and Shigeru Yamada proposed a new model to access the software reliability of Big data on the Cloud based on a three-dimensional Stochastic Differential Equation(SDE). Their method of analyzing software reliability depends upon three factors Network, Big data, and Cloud. If reporting a fault in the operation phase is represented by an irregular process and fault detection of software depends on time, Brownian motions are used to represent three kinds of noise reflecting the indirect effect of reliability. Under common assumptions for Software Reliability Growth Models (SRGM's) they formulated the following

$$dm(t)/dt = \beta(t)\{r(t) - m(t)\} \quad (16)$$

where $m(t)$ is a continuous real value function representing a cumulative number of detected faults, these faults were assumed to be latent in a bug tracking system of an OSS by operational time t ($t \geq 0$). $m(t)$ gradually increases during the operation phase as latent faults were detected and eliminated.

- $\beta(t)$ is a non-negative function and represents software fault-detection rate.
- $r(t)$ represents the change in requirement specifications and is defined as:

$$r(t) = -ae^{-bt} \quad (17)$$

a : number of latent faults in Open System Software (OSS) and b : represents the rate of change of requirements specification. $r(t)$ is used to represent the exponential growth of OSS requirement specification in terms of t . The positive value of β shows a

reliability growth trend and the negative value shows a reliability regression trend. Equation (1) is extended to 3D equation including Brownian motions as

$$dm_i(t)/dt = \{\beta_i(t) + \sigma_i \cdot v_i(t)\} \{r_i(t) - m_i(t)\}$$

where $i = 1, 2, 3$ (18)

where σ_1 , σ_2 , and σ_3 are positive constants representing irregular fluctuation levels, $v_1(t)$, $v_2(t)$, and $v_3(t)$ represents standardized Gaussian white noise. Also, it is assumed that $m_i(t)$ and $r_i(t)$ are related with $\beta_i(t)$ based on software failure, Cloud computing, Network environment, and Big data. The above-mentioned Equation (2) is represented as a SDE having ω_i Wiener process (an integration of white noise $v_i(t)$ with respect to time t). Then for each noise, the integrated Stochastic Differential Equation was obtained.

2.3.2. Reliability analysis based on AHP and software reliability models for big data on cloud computing, 2014: [57]

Researchers Yoshinobu Tamura and Shigeru Yamada propose a method of computing software reliability taking into consideration the 3V's characteristics of Big data on the Cloud platform. Parameters were estimated using Analytic Hierarchy Process (AHP) [58] based on the Big data model representing its characteristics as volume, velocity, and veracity. Three-dimensional Wiener process is used to model the effect of reliability

$$Dm_i(t)/dt = \{b_i(t) + \sigma_i \cdot v_i(t)\} \{r_i(t) - m_i(t)\},$$

where $i = 1, 2, 3$ (19)

function $r(t)$ represents the amount of change required in specifications. Where parameters are specified as

$b(t)$: Software fault-detection rate at time t .

σ : Irregular fluctuation magnitude.

v : Standardized Gaussian white noise.

Also, it is assumed that $m_i(t)$ and $r_i(t)$ are related with $\beta_i(t)$ based on software failure, Cloud computing Network environment, and Big data. The model was assessed based on AHP estimation results.

2.3.3. Software reliability assessment tool based on fault data clustering and hazard rate model considering cloud computing with big data, 2015: [59]

Yoshinobu Tamura and Shigeru Yamada proposed a method of accessing the reliability of software based on fault data clustering using Big data with a Cloud computing environment. Paper focused on collected fault data sets from various sources like Big database software, server software, and the Cloud. Since the amount of fault data is vast and managed by bug tracking systems K-means non-hierarchical Cluster Analysis is used because of its simplicity. The data set used is defined as θ_i and the fault data set in it is defined as a tuple (α_i, β_i) where $(i=1,2,\dots)$. The three-dimensional K-means clustering algorithm assumes that the clusters are randomly assigned to the fault data set and center points of clusters were calculated. The data sets were classified depending upon the nearest center point, calculated based on the minimum distance between fault data sets and cluster's center. This procedure continued till the specified conditions were met under the assumption that the level of fault severity depends upon the number of detected faults. Also, the cluster number m represents the severity level of classified software faults. Using the result analysis of K-means Cluster Analysis for identifying the software fault data Hazard rate model was proposed. A software reliability assessment tool AIR was also developed.

2.3.4. Reliability analysis based on a jump diffusion model with two wiener processes for cloud computing with big data, 2015: [60]

Yoshinobu Tamura and Shigeru Yamada proposed reliability analysis of Big data on the Cloud using the Jump Diffusion Model. To access the reliability of software developed as third-party software in the Cloud, one needs to consider the installed software, installer software, and Network traffic situation while monitoring the whole system for any debugging effect.

In the case of mobile devices, the security of the Network as well as the reliability of OSS become a major issue. Mobile devices are used by many individuals each using different installer software to access the Network, resulting in the installation of third-party software through the Network. These mobile devices access the Cloud service through specific Networks, and they reconfigure the data storage area of Cloud Computing Network

through unstructured and complex Big data from the Internet.

The indirect influence of Big data on reliability can result in system-wide failure. Interaction of indirect factors like Cloud, Big data and Network contributes to system reliability directly or indirectly, thus they proposed an effective method for reliability assessment.

Actual software fault data was used for the estimation of parameters used in the proposed model. An optimum maintenance problem was formulated using the amount of noise in the sample path to minimized software maintenance cost and time.

2.3.5. *Software reliability analysis considering the fault detection trends for big data on cloud computing, 2015: [61]*

Yoshinobu Tamura and Shigeru Yamada used the detected cumulative number faults and time-interval of software faults data sets to access Big data reliability on Cloud. Component-based as well as system-wise reliability was calculated. Software reliability is deeply analyzed using the Stochastic Differential Equation. The proposed model can estimate cumulative faults for database software as well as Cloud software. They found that at the end of fault detection Hadoop reliability is more stable as compared to OpenStack in the early operation phase.

2.3.6. *Method of reliability assessment based on neural network and fault data clustering for cloud with big data, 2015: [62]*

Yoshinobu Tamura, Yumi Nobukawa, and Shigeru Yamada proposed a software reliability assessment method based on neural Network and fault data clustering in the Cloud environment using Big data. K-means Cluster Analysis is used for fault data collected from Big databases (NoSQL and Hadoop) and Cloud software (Eucalyptus and OpenStack). Also proposed is the method of detecting cumulative faults using fault data for cluster analysis as input for Neural Networks. Based on Neural Network an estimation method is also proposed using estimates obtained through training data of fault data clustering. Discussed in the paper are the various numerical examples using fault data sets collected from websites of OpenStack and Hadoop using a bug tracking system from an open-source project. The result shows that the software fault of Hadoop is independent of OpenStack.

2.3.7. *Software reliability and cost analysis considering service user for cloud with big data, 2017: [63]*

Yoshinobu Tamura and Shigeru Yamada proposed a method based on Jump Diffusion Model on Cloud environment for reliability assessment. Based on Jump an optimum maintenance problem is also proposed and Genetic Algorithm is used for parameters estimation. Various examples were shown using datasets collected by using the bug tracking system. Minimizing the total software cost they estimated the optimum maintenance time for a maximum number of Cloud subscribers.

2.3.8. *Research on reliability evaluation of big data system, 2018: [64]*

Rui Cao And Jing Gao accessed Big data reliability on Cloud by using the Fault Tree Model. The type and cause of faults were summarized and analyzed using experiments. Using the Fault Tree Model one can formulate the qualitative assessment of a component and find out the probability of failure for the whole system. Computer-Aided Fault Tree Analysis (CAFTA) was used for reliability analysis by the researchers. Model is used for graphical visualization of reliability and can not only find the probability of failure but also the cause of the problem.

2.3.9. *Development of software reliability models using a hybrid approach and validation of the proposed models using big data, 2018: [65]*

P. Govindasamy and R. Dillibabu proposed three hybrid models using the NHPP model [42, 51, 52] Weibull model [66] and Exponential model [67]. Weibull model was used for hardware induced faults, NHPP models for pure software faults, and the Exponential model was adapted for user induced faults. Validation of the model was done using data collected during testing, Genetic Algorithm and Maximum Likelihood Estimation were used for estimating the value of the parameter. Calculation of cumulative and expected number of failures was done and observed values were compared to determine the performance of the model. Graphs were plotted to show the better estimation capability of the model as compared to other models and a comparison criterion was formulated to determine the estimation accuracy of the proposed model. A paired test was also done to show that the proposed model is better than other models in not only reliability estimation but also in the estimation of another metric like

mean failures, cumulative failures, and intensity of failures.

2.3.10. Study of software reliability on big data open-source software, 2019: [68]

Ranjan Kumar, Subhash Kumar, and Sanjay K. Tiwari determined the suitability of NHPP based reliability models for open-source big fault data. To determine whether a best-fit model can be the best predictor, five NHPP models on various prediction criteria and fitness scales were compared. They found out that the best-fitted model is not the best predictor for fault data.

3. Summarization

To access the reliability of software many models have been developed. However, in the context of Big data software reliability, only a few have been presented. Big database software like Spark and Hadoop is often used with OpenStack and Eucalyptus, Open-Source Software's, therefore it is important to access the reliability of big software in conjunction with Cloud computing for the overall estimation of system reliability. It is also important to determine the indirect effect of overly complex Big data characteristics on software reliability.

There are various reliability models available, but they cannot access the indirect influence on software reliability resulting due to interdependencies of Big data, Cloud service, Open-Source Software, and Network traffic. It is difficult for traditional models to formulate these indirect influences as parameters or noise for the overall reliability assessment of software.

Various techniques incorporating the external factors and interdependencies related to Big data in reliability assessment have been tabulated in Table 1.

It is clear from the summary that most models are Cloud-based having integrated reliability because of the relationship between Big data, Cloud, and Network. Various methods like AHP, Maximum Likelihood, and Least Square are used for parameter estimation along with Stochastic Differential Equation for reliability computation.

In Jump Diffusion Model a jump term is added to incorporate the indirect effect of either the Big data factor or Cloud computing factor resulting due to complicated data usage over the internet by Cloud software. Moreover, several noises have been applied for proper reliability assessment using indirect factors.

4. Comparison of different reliability assessment methods

With the help of a comparative statement, we can visualize all possible alternatives, compare, and contrast them for a specific application and select the best suitable one depending upon the underlined requirements.

In Table 2, various Big data reliability techniques are compared in terms of external factors or interdependencies affecting the reliability, the main difference between the techniques while computing reliability, the estimation and evaluation methods used for parameter estimation, type of mean value function used while calculating parameters value and whether the value of the cumulative and expected number of faults are derived or not.

This comparison outlined certain facts about the compared techniques. It was observed that all techniques incorporate the multi-factor and most of them use the Stochastic Differential Equation with Wiener process to include these interactions.

The dimension of the Wiener process is chosen to reflect these interdependencies also the jump term and hazard value is used to incorporate Big data influence while reducing the dimension of the Wiener process.

Some techniques consider the system-wide reliability including the effect of Big data in Cloud computing environment, while other techniques develop component-based reliability assessment. K-means clustering is used because of its simplicity and ease to work with complicated data.

For parameters estimation most techniques use MLE for evaluating parameters related to specified models and Genetic Algorithm is used for the estimation of jump-diffusion parameters. The rapid growth of new technologies suggests that these algorithms will be used for Big data software reliability in the coming years. The above-mentioned table helps the researchers who are working in the field of Big data reliability to understand the type of technique that can be utilized for their work.

5. Graphical representation

The graph in Fig. 2 shows the percentage of published papers per year from 2012 till 2019. Maximum papers were published in 2014, 2015, and 2017 whereas min papers were published in the year 2016 and 2013.

Table 1
Summarization of Big data software reliability papers

S.No.	Study	Year	Data	Summary of result	Ref.
1.	Xiaoyue Han, Lianhua Tian, Minjoo Yoon, Minsoo Lee	2012	Social Network	Increase in the reliability of recommended information	[69]
2.	Yoshinobu Tamura and Shigeru Yamada	2014	Data collected using Bug tracking system on OpenStack website	The model can access the combined system reliability dealing with factors like Big data, Network traffic, and Cloud computing. Also, the model can visually assess the reliability because of sensitivity analysis using the volume, velocity, and variety: the 3V's of Big data as noise.	[57]
3.	Yoshinobu Tamura, Kenta Miyaoka and Shigeru Yamada	2014	Bug tracking data of OpenStack collected from the website.	System reliability can be accessed in terms of Big data renewal rate, Network traffic, and software failure. Reasonably selecting the value of model parameters results in a more accurate assessment than traditional models.	[56]
4.	Yoshinobu Tamura and Shigeru Yamada	2015	Data collected using Bug tracking system on OpenStack website	Efficient reliability assessment using developed AIR tool.	[59]
5.	Yoshinobu Tamura and Shigeru Yamada	2015	Actual software fault data.	Optimal maintenance problem on Cloud environment is defined based on the noise in sample-path. The proposed method can be used for dependability assessment.	[60]
6.	Yoshinobu Tamura and Shigeru Yamada	2015	Data collected using Bug tracking system on OpenStack and Hadoop website of Open system projects.	The reliability of Hadoop is more stable than OpenStack after the end of fault detection time during the early phase.	[61]
7.	Yoshinobu Tamura, Yumi Nobukawa, Shigeru Yamada	2015	Software fault dataset, Hadoop, and OpenStack bug tracking data	Hadoop and OpenStack might have independent software faults.	[62]
8.	Yoshinobu Tamura and Shigeru Yamada	2017	Data collected using Bug tracking system on OpenStack and Hadoop website of Open system projects.	Useful for software managers for reliability assessment.	[63]
9.	Yoshinobu Tamura and Shigeru Yamada	2017	Data sets of Apache HTTP server	The proposed deep learning method increases the reliability of OSS and application software was developed using NW.js technology as the cross-platform to analyze fault data available on the bug tracking system.	[70]
10.	Rui Cao and Jing Gao	2018	uses CAFTA	Used for Quantitative assessment for a component and visualization for reliability assessment.	[64]
11.	P Govindasamy R. Dillibabu	2018	Software failure data collected during the testing phase	Better estimation capability of the proposed hybrid model than traditional NHPP models.	[65]
12.	Svitlana Yaremchuck, Vyacheslav Kharchenko	2018	Tera-PROMISE Repository	Formalized the software similarity features and method of reliability analysis and search is extended to similar systems	[71]
13.	Ranjan Kumar, Subhash Kumar, Sanjay K. Tiwari	2019	Data collected from JIRA bug tracking and management tool of Hadoop and Open Spark on OSS.	It was found that the best-fitted model on the bug dataset is not the best predictor.	[68]
14.	Ritu Ratra and Preeti Gulia	2019	Review paper which provides Comparison among Big data tools and techniques.	Differentiation between Hadoop, Spark, and MongoDB is highlighted concerning various parameters.	[22]

Table 2
Comparative analysis of Big data reliability techniques

S. No	Technique	Inter-dependencies	Difference	Parameter estimation	Mean value functions	Cumulative no. of faults derived	Expected number of faults derived
1.	Stochastic Differential Equation	Big data, Network traffic, Cloud computing	Use 3 kinds of Brownian motion to show interdependencies between Cloud, Big data, and Network	Maximum likelihood.	Inflection S-shaped SRGM	yes	yes
2.	AHP and Wiener Process	Big data, Network traffic, Cloud computing	Use the method of reliability assessment in terms of AHP using 3V's of Big data	Maximum likelihood	Inflection S-shaped SRGM	yes	yes
3.	Jump Diffusion with Wiener Process	Big data, Mobile Clouds, Cloud computing	Jump term is added in the stochastic equation of reliability assessment to incorporate Big data factor	Maximum likelihood and Genetic algorithms (for jump-diffusion parameters)	Inflection S-shaped SRGM	No	No
4.	Hazard rate with Stochastic Differential Equation	Big database and Cloud software.	Component-based reliability assessment	Using distribution function of hazard rate model.	Inflection S-shaped SRGM	No	No
5.	Clustering and Hazard Rate	Big database and Cloud software	K-mean Clustering is applied on fault data obtained from Big data, Cloud, and server software.	Maximum likelihood	Not required	No	No
6.	Neural Network and Clustering	Big database and Cloud software	Software fault data is applied to the input of NN and learning data is derived using normalized Mean Square errors obtained through clustering of fault data.	Not required	Not required	No	No
7.	Neural Network and Jump Diffusion	Big database and Cloud software	Reliability assessment based on the component ratio of database software per unit time on Cloud	Maximum likelihood and Genetic algorithms (for jump-diffusion parameters)	Delayed-S shaped SRGM	No	No
8.	Development of hybrid models	Weibull model, NHPP models, Exponential model	A hybrid model is developed	Using Maximum likelihood, G.A, and MATLAB 2014a	No	No	No

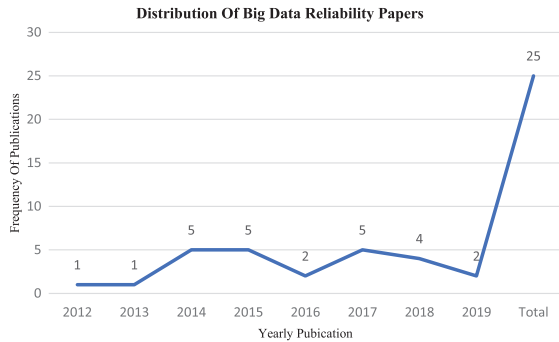


Fig. 2. Analysis of Big data reliability.

Although there are many papers published related to Big data, discussing its applications, techniques, tools, and analytics used but we have taken only those which discuss the reliability of Big data and classify them according to hardware, software, and data reliability.

ability. As shown in Fig. 3. The overall percentage of papers based on hardware reliability is 16%, 28% discussed the issues related to data reliability, and 56% of papers emphasized software reliability in Big data.

In Fig. 4 Software reliability models are compared (X-axis represents the paper numbers taken from Table 1) based on different factors or parameters like Maximum Likelihood Estimation, Stochastic Equation used, inflection S-Shaped SRGM and fault dataset. It was found that out of 56% papers on software reliability in Big data, approximately 35% papers used Maximum Likelihood Function for parameter estimation and Stochastic Differential Equation to determine reliability. To calculate the mean value function 64% papers used inflection S-shaped SRGM and to evaluate software reliability 71% papers used fault data of Hadoop and Spark.

Figure 5 is the graphical representation of the comparative view of the different techniques used

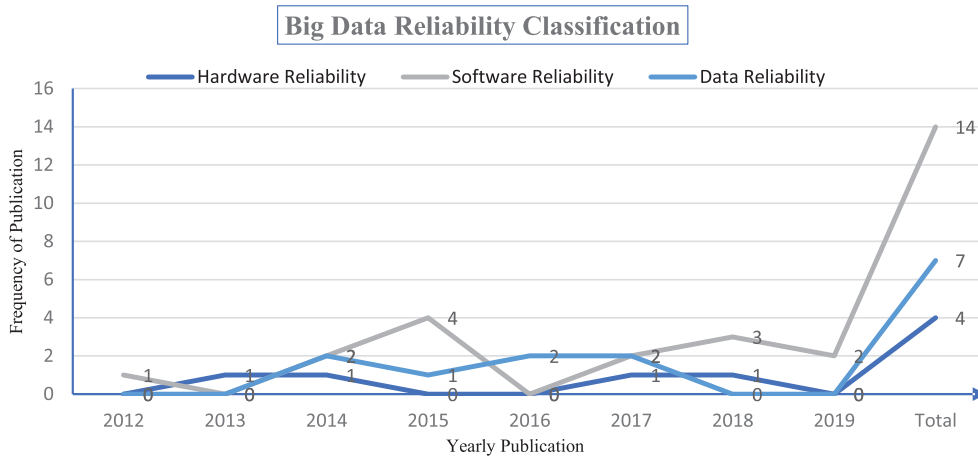


Fig. 3. Big data reliability classification.

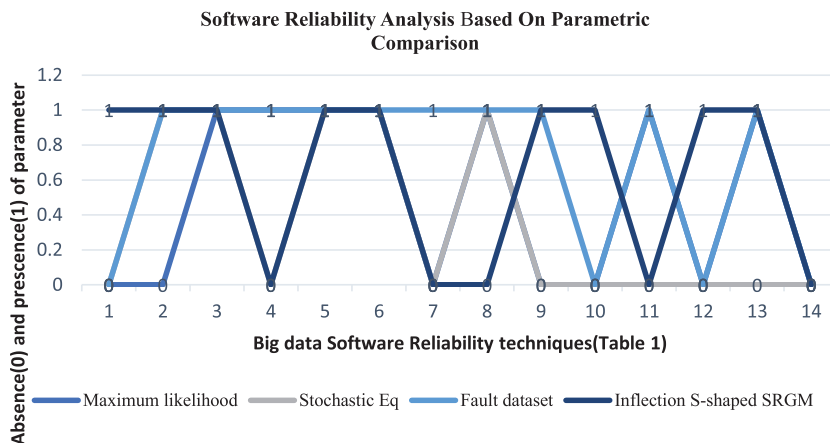


Fig. 4. Big data software reliability analysis.

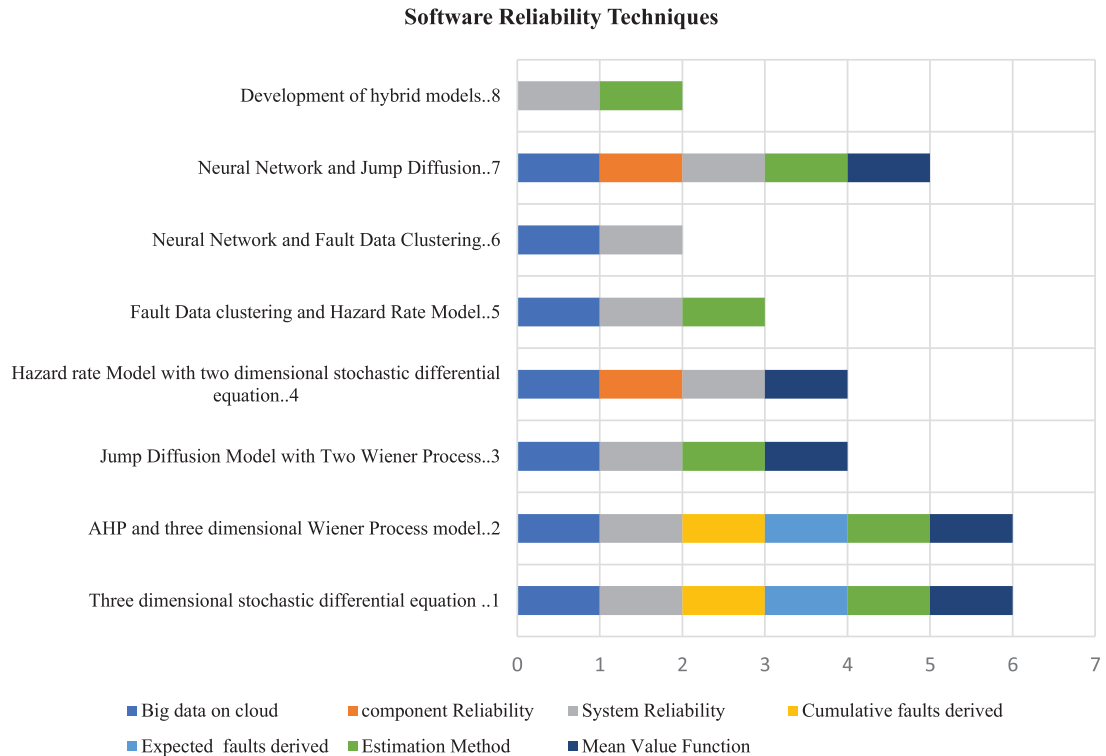


Fig. 5. Comparison of software reliability techniques for Big data.

(Y-axis represents the techniques represented by serial number in Table 2) to calculate software reliability in Big data. Main techniques used in major papers were identified and a comparison is made among them depending upon the various factors shown in the graph. Most papers calculate the system-wide reliability whereas some papers also calculated the component reliability of Big database and Soft computing software.

Expected and cumulative faults were derived in few techniques using three-dimensional Stochastic Equation for software reliability. Majority of techniques calculated the mean value function using inflection and delayed S-shaped SRGM. Some techniques used only Maximum Likelihood Estimation for parameter evaluation while others used LSE either with Genetic algorithm or some other methods for determining the overall parameter.

6. Conclusion

In this paper, various research papers based on Big data reliability are studied and their classification is done based on various important parameters.

The Paper also summarizes certain new techniques required to handle Big data. Additionally, few commonly used software reliability models were also explained. The Paper also provides a comprehensive insight into newly developed reliability models of Big data. It is difficult to determine the indirect effect of Big data in reliability computation along with fault data that is why only a few effective measures have been proposed based on the various dimensions to incorporate the multi factors in various papers.

The study indicates that most papers use Stochastic Differential Equation to determine the reliability of the system while considering the multiple interactions either as the additional dimension or a jump value or by using hazard rate function. Maximum Likelihood Method and inflection S-shaped SRGM is employed by most techniques for parameter estimation and for calculating the mean value, respectively. A hybrid model for reliability assessment was also developed which is derived using NHPP, Weibull, and Exponential models giving a better estimation of reliability than most of the traditional models for Big fault data.

While comparing these reliability techniques it came into notice that some models calculate the

reliability of the whole system on the cloud, considering the interactions between Network traffic and Cloud software with the Big data software while others calculate the component reliability of software modules of a system separately. These Big data techniques have been compared and provide a meticulous understanding of what type of mathematical models one needs to develop for a specific platform, which methods were employed to calculate the value of reliability parameters, what type of soft computing techniques can be used for determining the value of parameters, what type of datasets were being used, how the developed model is being utilized.

The motivation of this paper is to provide a common base for all the researchers working in the field of Big data reliability. Researchers can be benefitted from the provided summary and can develop their model by considering the methods and techniques mostly employed while analyzing the reliability in Big data scenarios.

This paper provides a comparative review of the techniques used for Big data reliability with the belief that it will support the understanding of current developments and future research prospects in this direction. Further work required the analysis of all these models for a specific data set while adjusting certain parameters for uniformity, using new techniques to calculate the value of the parameter of these overly complex equations, and comparing which one gives the more accurate prediction. Existing soft computing techniques can also be utilized to optimize the result of these models or a new one can be developed for a specific model which gives more accurate reliability prediction.

References

- [1] N. Elgendy and A. Elragal, "Big data analytics: A literature review paper," in *Industrial conference on data mining*, Cham, pp. 214–227, 2014.
- [2] D.P. Achariya and K. Ahmed, A survey on Big data analytics: challenges, open research issues, and tools, *International Journal of Advanced Computer Science and Applications* **7**(2) (2016), 511–518.
- [3] A. Kamilaris, A. Kartakoullis and F.X. Prenafeta-Boldu, A review on the practice of Big data analysis in agriculture, *Computers and Electronics in Agriculture* **143** (2017), 23–37.
- [4] W. Zou, W. Jing, G. Chen, Y. Lu and H. Song, A survey of Big data analytics for smart forestry, *IEEE Access* **7** (2019), 46621–46636.
- [5] A.R. Al-Ali, I.A. Zualkernan, M. Rashid, R. Gupta and M. Alikarar, A smart home energy management system using IoT and Big data analytics approach, *IEEE Transactions on Consumer Electronics* **63**(4) (2017), 426–434.
- [6] Y. Zhang, T. Huang and E.F. Bompard, Big data analytics in smart grids: A review, *Energy Informatics* **1**(1) (2018), 8.
- [7] I. Adjerid and K. Kelley, Big data in psychology: A framework for research advancement, *American Psychologist* **73**(7) (2018), 899.
- [8] S. Chauhan, N. Agarwal and A. Kumar, Addressing Big data challenges in smart cities: A systematic literature review, *info* **18**(4) (2016), 73–90.
- [9] L. Hu, K. Liu, Y. Diao, X. Meng and W. Sheng, Operational reliability evaluation method based on Big data technology, 2016 International Conference on Cyber-Enabled Distributed *Review* **23**(5) (1983), 903–943.
- [10] T.M. Choi, S.W. Wallace and Y. Wang, Big data analytics in operations management, *Production and Operations Management* **27**(10) (2018), 1868–1883.
- [11] V. Chang, A proposed social Network analysis platform for Big data analytics, *Technological Forecasting and Social Change* **130** (2018), 57–68.
- [12] N. Venkatesh, Comparative analysis of Big data, Big data analytics: Challenges and trends, *International Research Journal of Engineering and Technology (IRJET)* **5** (2018), 1948–1964.
- [13] U. Narayanan, V. Paul and S. Joseph, Different analytical techniques for Big data analysis: A review, in *International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS-2017)*, Chennai, pp. 372–382, 2017.
- [14] A. Gandomi and M. Haider, Beyond the hype: Big data concepts, methods, and analytics, *International Journal of Information Management* **35**(2) (2015), 137–144.
- [15] J. Zakir, T. Seymour and K. Berg, Big data analytics, *Issues in Information Systems* **16**(2) (2015), 81–90.
- [16] I.M. Ali, Y.Y. Jusoh, R. Abdullah, R. Nor, H. Nor and L.S. Affendey, Measuring the performance of Big data analytical process, *Journal of Theoretical and Applied Information Technology* **97**(14) (2019), 3796–3808.
- [17] A. Hudaib and M. Moshref, Survey in software reliability growth models: Parameter estimation and models ranking, *International Journal of Computer Systems* **5**(5) (2018), 11–25.
- [18] J.G. shantikumar, Software reliability models: A Computing and Knowledge Discovery (CyberC), Chengdu, 2016, pp. 341–344, doi: 10.1109/CyberC.2016.71.
- [19] M. Spichkova, H.W. Schmidt, II. Yusuf, I.E. Thomas, S. Androulakis, G.R. Meyer, Towards modelling and implementation of reliability and usability features for research-oriented Cloud computing platforms, in *Communication in Computer and Information Science*, **703** (2016), 158–178, cham, springer.
- [20] Y. Tamura and S. Yamada, Optimisation analysis for reliability assessment based on Stochastic Differential Equation modeling for open source software, *International Journal of Systems Science* **40**(4) (2009), 429–438.
- [21] R. Nachiappan, B. Javadi and R.N. Calhe, Cloud storage reliability for Big data applications: A state of the art survey, *Journal of Network and Computer Applications* **97** (2017), 35–47.
- [22] R. Ratra and P. Gulia, Big data tools and techniques: A roadmap for predictive analytics, *International Journal of Engineering and Advanced Technology (IJEAT)* **9**(2) (2019), 4986–4992.
- [23] P. Gupta, A. Sharma and R. Jindal, Scalable machine-learning algorithms for Big data analytics: A comprehensive review, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **6**(6) (2016), 194–214.

- [24] J. Fox, Applied regression analysis, linear models and methods, Sage Publications, 1997.
- [25] W. Medhat, A. Hassan and H. Korashy, Sentiment analysis algorithms and applications: A survey, *Ain Shams Engineering Journal* **5**(4) (2014), 1093–1113.
- [26] M. Srinivas and L.M. Patnaik, Genetic Algorithms: A survey, *Computer* **27**(6) (1994), 17–26.
- [27] D.E. Goldberg, Genetic algorithm, Pearson education India, 2006.
- [28] J. Qiu, Q. Wu, G. Ding, Y. Xu and S. Feng, A survey of machine learning for Big data processing, *EURASIP Journal on Advances in Signal Processing* **2016**(1) (2016), 67.
- [29] Z. Lv, H. Song, P. Basanta, A. Steed and M. Jo, Next-generation Big data analytics: State of the Art, Challenges, and Future Research Topics, *IEEE transactions on Industrial Informatics* **13**(4) (2017), 1891–1899.
- [30] A. L'Heureux, K. Grolinger, H.F. Elyamany and M.A. Capretz, Machine Learning with Big data : Challenges and approaches, *IEEE Access* **5** (2017), 7776–7797.
- [31] R.H. Hariri, E.M. Fredericks and K.M. Bowers, Uncertainty in Big data analytics: survey, opportunities, and challenges, *Journal of Big data* **6**(1) (2019), 44.
- [32] E.D. Karnin, A simple procedure for pruning back-propagation trained neural Networks, *IEEE Transactions on Neural Network* **1**(2) (1990), 239–242.
- [33] I. Lakshmanan and S. Ramasamy, An Artificial Neural-Network approach to software reliability growth modeling, *Procedia Computer Science* **57** (2015), 695–702.
- [34] X. Chen and X. Lin, Big data Deep Learning: Challenges and perspective, *IEEE Access* **2** (2014), 514–525.
- [35] S. Arora and I. Chana, A survey of clustering techniques for Big data analysis, in *5th International Conference-Confluence The Next Generation Information Technology Summit*, Noida, pp. 59–65, 2014.
- [36] A.K. Jain and R.C. Dubes, Algorithms for clustering data., Prentice-Hall Inc, 1988.
- [37] P.N. Misra, Software reliability analysis, *IBM Systems Journal* **22**(3) (1983), 262–270.
- [38] G.J. Schick and R.W. Wolverson, An analysis of computing Software reliability models, *IEEE Transactions on Software Engineering* **SE-4**(2) (1978), 104–120.
- [39] M. Prasad, L. Flowrence and C.V. Srikrishna, Overview of software reliability models, *International Journal of Engineering and Management Research (IJEMR)* **3**(5) (2013), 11–15.
- [40] A.L. Goel, Software reliability models: Assumptions, limitations, and applicability, *IEEE Transactions on Software Engineering* **SE-11**(12) (1985), 1411–1423.
- [41] Z. Jelinski and P. Moranda, Software reliability research, *Statistical Computer Performance Evaluation*, pp. 465–484, 1972.
- [42] A.L. Goel and K. Okumoto, Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Transactions on Reliability* **R-28**(3) (1979), 206–211.
- [43] G.J. Pai, A survey of software reliability models. *arXiv preprint arXiv 1304.4539*, 2013.
- [44] J.D. Musa, Software reliability measurement, *Journal of Systems and Software* **1** (1979), 223–241.
- [45] B. Littlewood and J.L. Verrall, A Bayesian reliability model with a stochastically monotone failure rate, *IEEE Transactions on Reliability* **R-23**(2) (1974), 108–114.
- [46] D.D. Hanagal and N.N. Bhalerao, Modeling on generalized extended inverse Weibull software reliability growth model, *Journal of Data Science* **17**(3) (2019), 575–592.
- [47] M. Xie and B. Bergman, On modeling reliability growth for software, *IFAC Identification and Systems* **21**(9) (1988), 567–570.
- [48] M. Ohba, S. Yamada, K. Takeda and S. Osaki, S-shaped software reliability growth curve: how good is it? Proceedings IEEE COMPSAC 82, Chicago, 1982, pp. 38–4.
- [49] M. Ohba and S. Yamada, S-shaped software reliability growth models, *In International Colloquium on Reliability and Maintainability* **4** (1984), 430–436.
- [50] M. Ohba, Inflection S-shaped software reliability growth model, in *Stochastic Models in Reliability Theory*, Heidelberg, 1984.
- [51] P.K. Kapur and R.B. Garg, Optimum release policy for an inflection s-shaped software reliability growth model, *Microelectronics Reliability* **31**(1) (1991), 39–41.
- [52] S. Yamada, M. Ohba and S. Osaki, S-shaped reliability growth modeling for software error detection, *IEEE Transactions on Reliability* **R-32**(5) (1983), 475–484.
- [53] S. Yamada, M. Ohba and S. Osaki, S-shaped software reliability growth models and their applications, *IEEE Transactions on Reliability* **R-33**(4) (1984), 289–292.
- [54] M. Ohba, Software reliability analysis models, *IBM Journal of Research and Development* **28**(4) (1984), 428–443.
- [55] R.K. Sharma, A. Kumar and S. Bajaj, Analysis of various software reliability models and proposing a new model of software reliability for embedded systems, *International Journal of Innovative Research in Computer Science & Technology (IJRCST)* **5**(3) (2017), 287–290.
- [56] Y. Tamura, K. Miyaoka and S. Yamada, Reliability analysis based on three-dimensional stochastic differential equation for Big data on Cloud computing, pp. 863–867, 2014.
- [57] Y. Tamaru and S. Yamada, Reliability analysis based on AHP and software reliability models for big data on cloud computing, *International Journal of Statistics – Theory and Applications* **1**(1) (2014), 43–49.
- [58] R.W. Saaty, The analytic hierarchy process—what it is and how it is used, *Mathematical modelling* **9**(3–5) (1987), 161–176.
- [59] Y. Tamura and S. Yamada, “Software reliability assessment tool based on fault data clustering and hazard rate model considering Cloud computing with Big data,” in *2015 4th International Conference on Reliability, Info-com Technologies and Optimization (ICRITO) (Trends and Future Directions)*, Noida, pp. 1–6, 2015.
- [60] Y. Tamura and S. Yamada, Reliability analysis based on a jump diffusion model with two wiener processes for cloud computing with big data, *Entropy* **17**(12) (2015), 4533–4546.
- [61] Y. Tamura and S. Yamada, Software reliability analysis considering the fault detection trends for big data on cloud computing, in *Industrial Engineering, Management Science and Applications*, Berlin, Heidelberg, Springer **349** (2015), 1021–1030.
- [62] Y. Tamura, Y. Nobukawa and S. Yamada, “A method of reliability assessment based on Neural Network and fault data clustering for Cloud with Big data,” in *2015 2nd International Conference on Information Science and Security (ICISS)*, Seoul, pp. 1–4, 2015.
- [63] Y. Tamura T. Takeuchi and S. Yamada, Software reliability and cost analysis considering service user for Cloud with Big data, *International Journal of Reliability, Quality, and Safety Engineering* **24**(2) (2017), 1–14.
- [64] R. Cao and J. Gao, “research on reliability evaluation of Big data system,” in *2018 the 3rd IEEE International Conference on Cloud Computing and Big data*

- Analysis*, Chengdu, 2018 pp. 261–265, doi: 10.1109/ICC-CBDA.2018.8386523(11).
- [65] P. Govindasamy and R. Dillibabu, Development of software reliability models using a hybrid approach and validation of the proposed models using Big data, *The Journal of Supercomputing* **76** (2018), 1–2.
- [66] S. Yamada, J. Hishitani and S. Osaki, Software-reliability growth with a Weibull test-effort: a model and application, in *IEEE Transactions on Reliability* **42**(1) (1993), 100–106.
- [67] R. Miller, Exponential order statistic models of software reliability growth, *IEEE Transactions on Software Engineering* **SE-12**(1) (1986), 12–24.
- [68] R. Kumar, S. Kumar and S.K. Tiwari, A study of software reliability on Big data Open-Source Software, *International Journal of System Assurance Engineering and Management* **10** (2019), 242–250.
- [69] X. Han, L. Tian, M. Yoon and M. Lee, A Big data model supporting information recommendation in Social Networks, in *2012 Second International Conference on Cloud and Green Computing*, Xiangtan, pp. 810–813, 2012.
- [70] Y. Tamura and S. Yamada, “Fault identification and reliability assessment tool based on deep learning for fault Big data,” *Journal of Software Networking* **2017**(1) (2018), 161–167.
- [71] S. Yaremchuk and V. Kharchenko, Big data and similarity-based software reliability assessment: The technique and applied tools, in *The 9th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT 2018*, Kyiv, 2018, pp. 485–490, doi: 10.1109/DESSERT.2018.8409182.
- [72] R. Tkachenko and I. Izonin, Model and principles for the implementation of neural-like structures based on Geometric Data Transformations, in *International Conference on Computer Science, Engineering and Education Applications ICCSEEA 2018: Advances in Computer Science for Engineering and Education* **754** (2019), 578–587.

Ranking of Reliability Models based on Accurate Estimation and Weighted Function

Shalini Sharma

School of Computing Science and
Engineering
Galgotias University, Greater Noida
shalini.sharma@rediffmail.com

Naresh Kumar

School of Computing Science and
Engineering
Galgotias University, Greater Noida
Naresh.dhull@gmail.com

Kuldeep Sing Kaswan

School of Computing Science and
Engineering
Galgotias University, Greater Noida
kaswankuldeep@gmail.com

Abstract—Many software reliability models have been developed and are in use extensively. Because of advancements in technology, no single model gives optimal results for all applications; therefore, selecting the best model for a particular application always remains an area of interest. This paper uses different computational methods using various weighted functions to determine models' ranking. A set of thirteen comparison criteria using observed value and an estimation value of models has been formulated. We assess the model's suitability in estimating the fault by calculating model rank based on various factors. We developed a ranking model that takes into account the weightage of the model ranks in all individual comparison criteria and incorporates the estimation capability of a model through a weighted estimation function.

Keywords— *Ranking Algorithm, Comparison Criteria, Hybrid Models, Weighted Rank Method*

I. INTRODUCTION

In recent times software has become an integral part of our lives. Software quality determined by its reliability is specified as (IEEE Std. 1633-2016) the probability of failure-free operation of software under specific conditions for a limited time. Software reliability growth models were developed to determine the quality of developed software and predict the failure phenomenon. This paper investigated the most commonly used two or three-parameter Non-homogeneous Poisson process models and selected the best fit model based on their ranking. The software failure rate can be estimated by collecting fault data as a function of time. To guarantee failure-free software, one must use various tools and techniques to keep track of fault, determine its cause, and correct it in order to avoid further defects. The quality of the software is determined using various well-known models using multiple assumptions in determining the reliability and has been proposed earlier. Best model selection required that the models be classified and filtered based on their suitability for the product environment, techniques used, and basic assumptions used. The model best suited for the application after all the scrutiny is selected. If there is no such model, one needs to develop a tailored model specifically for the requirements. We propose a ranking method that considers the model's performance based on pre-set comparison criteria and evaluates the model's prediction capability in comparison with the observed values.

The paper is structured as: Section II gives a brief introduction of literature related to reliability modeling. Section III defines various comparison criteria formulas used to determine the individual criterion ranking of the reliability model. Section IV explains our methodology for determining rank. Section V used the developed method using fault data

and derived the rank of considered models. Section VI concludes the paper.

II. LITERATURE REVIEW

The reason for software failures is the fault remained in software; detection and removal of these inherent faults is desired for the successful operation of any automated system [1]

In the last 40 years, researchers developed many reliability models to aid decisions relating to software release in the market. Models were classified into various categories by various researchers. Broadly models can be classified into two categories static and dynamic. A static model uses software metrics to estimate the number of defects (or faults) in the software, whereas a dynamic model uses the past failure discovery rate during software execution overtime to count the number of failures [2]. Some well-known Software Reliability Growth Models (SRGM) are Goel-Okumoto (GO): first Non-Homogeneous Poisson Process Model (NHPP), Generalized Goel: A generalization of GO model, Yamada Delayed S-shaped model, Yamada Inflection S-shaped model, Logistic growth curve model, and Yamada exponential model. Each model gives a good result for a specified data set, but no one can be selected as the best model for all types of datasets.

Another classification divides the models into five categories: Early Prediction Models, Software Reliability Growth Models, Architecture-Based Models, Hybrid Black Box Models, and Hybrid White Box Models.

Software fault prediction helps in maintaining the quality of the software product [3]. Early Prediction Models extrapolates the characteristics of software from the requirement phase to the testing phase to predict the software behavior during operation [4], SRGM concludes software failure during the testing phase [5], Input domain model utilizes the input properties to determine the probability of correct estimation from properly executed test cases [6], Architecture-based models derive the reliability estimate by combining the reliability estimate of different software modules [4], Hybrid Black Box model is a combination of the input domain model and SRGMs features. In contrast, Hybrid White Box Models use specific characteristics of white-box models and black box models [7].

SRGM based models are further classified into NHPP models and failure rate models [6], whereas architecture-based models are divided into state-based models, additive models, and path-based models [7].

The mean value function (MF) and Intensity function (IF) of some well-known models are tabulated in TABLE I.

TABLE I. NHPP MODELS

Models	$m(t)$	$\lambda(t)$
Goel-Okumoto Model [8]	$x(1 - e^{-yt})$	xye^{-yt}
Generalized Goel [4]	$x(1 - e^{-yt^z})$	$xyz t^{z-1} e^{-yt^z}$
Gompert Model [2]	xze^{-yt}	$xy \ln(k) z e^{-yt} e^{-yt}$
Logistic Growth Curve [2]	$\frac{x}{(1 + ze^{-yt})}$	$\frac{xyze^{-yt}}{(1 + ze^{-yt})^2}$
Yamada Delayed S-Shaped [9]	$x\{1 - (1 + yt)e^{-yt}\}$	$xy^2 t e^{-yt}$
Yamada Inflection S-Shaped [9]	$\frac{x(1 - e^{-yt})}{1 + ze^{-yt}}$	$\frac{xye^{-yt}(1 + z)}{(1 + ze^{-yt})^2}$
Yamada exponential [10]	$x\{1 - e^{-zw(1-e^{-yt})}\}$	$xzwy e^{-zw(1-e^{-yt})-yt}$
Musa-Okumoto [2]	$x \log(1 + yt)$	$\frac{xy}{1 + yt}$
Modified Duane [4]	$a(1 - (\frac{b}{b+t})^c)$	$acb^c(b+t)^{1-c}$
Yamada imperfect debugging model1 [2]	$\frac{ab(e^{pt} - e^{-bt})}{p + b}$	$\frac{ab(pept + e^{-bt})}{p + b}$
Yamada imperfect debugging model2 [2]	$a(1 - e^{-bt})(1 - \frac{p}{b}) + pat$	$abe^{-bt(1-\frac{p}{b})} + pa$

III. COMPARISON CRITERIA

Many researchers [11],[7],[12],[13],[14],[15] used comparison criteria set to determine the ability of a model to give effective results. The comparison criteria set we used to evaluate models quantitatively is mentioned below.

A. Mean Value (Bias):

Bias gives deviation between estimated and observed values [14,15]. Smaller mean values are desired for best prediction as it signifies the lesser deviation between the observed and predicted values. If there are positive and negative values in the data, then all these terms cancel out each other while taking the summation, giving the wrong result. In such cases, it is advisable to use the Mean Absolute Error/Deviation (MAE/MAD) instead of Mean for better prediction.

$$\sum_{i=1}^n (Eval(i) - Oval(i))/n$$

The deviation of square errors between observed values and estimated values is being normalized by the

B. Mean Square Error (MSE):

MSE gives the summative square of observed and estimated values deviation [13]. It gives the best fit line corresponding to the data set, and smaller values of MSE indicate better prediction.

$$\sum_{i=1}^n \frac{(Oval(i) - Eval(i))^2}{n}$$

C. Mean Absolute Deviation (MAD):

MAD gives absolute deviation between observed and estimated values [13]. A smaller value means lesser variation.

$$\sum_{i=1}^n \frac{|Oval(i) - Eval(i)|}{n}$$

D. Predictive Ratio Risk (PRR)

PRR gives a summative difference between the estimated and observed values compared to calculated values [17].

$$\sum_{i=1}^n \frac{(Eval(i) - Oval(i))}{Eval(i)}$$

E. Noise

Noise gives the sum of the difference between intensity function at time t and intensity function at time t-1, concerning intensity function value at time t-1 [16].

$$\sum_i^n \frac{\lambda(t_i) - \lambda(t_{i-1})}{\lambda(t_{i-1})}$$

F. Root Mean Square Error (RMSE)

RMSE gives the square root of Mean Square Error [13]. Lower the value better is the prediction capability of a model.

$$\sum_{i=1}^n \frac{(Oval(i) - Eval(i))^2}{n}$$

G. Relative Absolute Error (RAE)

RAE gives the summative absolute difference between the observed and estimated values concerning the sum of the absolute difference between the estimated value and the mean estimated value of the model [13]. For a model to be a better estimator, this ratio must be close to zero.

$$\frac{\sum_{i=1}^n |Oval(i) - Eval(i)|}{\sum_{i=1}^n |Eval(i) - \bar{Eval}|}$$

H. Root Relative Squared Error (RRSE)

RRSE gives the square root of the summative squared difference between the estimated and observed value compared to the summative squared difference between the observed value and the mean of practical value [13].

squared deviation of the estimated values from the mean value of the model. Smaller values of RRSE give a better fitting model.

$$\sqrt{\frac{\sum_{i=1}^n (Eval(i) - Oval(i))^2}{\sum_{i=1}^n (Oval(i) - \bar{Oval})^2}}$$

I. Mean Magnitude of Relative Error (MMRE)

MMRE gives MAD divided by observed values [13]. Ideally, for a perfect fit, the value of absolute error should be zero. While accessing the model for a good fit, this value must be closer to zero.

$$\frac{1}{n} \sum_{i=1}^n \left| \frac{Oval(i) - Eval(i)}{Oval(i)} \right|$$

J. Predictive Power (PP)

PP gives the sum of the square difference between the estimated and observed values regarding the observed value [13].

$$\sum_{i=1}^n \left(\frac{Eval(i) - Oval(i)}{Oval(i)} \right)^2$$

K. Coefficient Of Determination (R²)

R² gives deviation from the sum of the squared difference between the observed value and estimated value concerning the sum of the squared deviation of the practical value from its mean value [18]. A high R² value near 1 represents a good fit.

$$1 - \frac{\sum_{i=1}^n (Oval(i) - Eval(i))^2}{\sum_{i=1}^n (Oval(i) - \bar{Oval(i)})^2}$$

L. Accuracy Of estimation (AE)

AE gives the difference between observed and estimated cumulated values concerning observed cumulative value [18]. Near zero values of cumulative difference provides a good fit, so smaller values of AE are desirable for better estimation.

$$\frac{O(a) - E(a)}{O(a)}$$

M. Theil's Statistics (TS)

TS gives the percentage of average deviation concerning the summative square of observed values [17]. Smaller values of these statistics provide a good fit.

$$\sqrt{\frac{\sum_{i=1}^n (Eval(i) - Oval(i))^2}{\sum_{i=1}^n Oval(i)^2}} \times 100\%$$

IV. RANKING METHOD

we developed a methodology to rank the models based on a weighted ranking method. For this, we first evaluate each model's criteria values and arrange them in matrix form called

Criteria Matrix. Then criteria values were ranked, assigned weights, and estimation functions were calculated to evaluate the rank of a model.

A. Criteria Matrix

Each element a_{ij} that belongs to this matrix represents the j^{th} criteria value of an i^{th} model. If there are n criteria and m models to rank, then we can represent this matrix as:

$$A = \text{Mat} [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

B. Individual Criteria Ranking Matrix

Let us call this matrix R. It consists of ranking all m models according to each n criteria. We can represent it as

$$R = \text{Mat} [r_{ij}] = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \dots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{bmatrix}$$

C. Weight Matrix

Next, we assign the weights to all these ranks, Then the model having a consistent lower position gives a low value. The model with the lowest sum of weight is considered the best model compared to other models against considered criteria for evaluation. The highest weight, seven, is assigned to rank 11, and the lowest weight, 0.0001, is given to rank 1 in a gradually increasing manner according to the rank number as shown in TABLE II.

TABLE II. WEIGHT VECTOR ASSIGNED TO RANKS

R1	R2	R3	R4	R 5	R 6	R 7	R 8	R 9	R 10	R 11
W1	W2	W3	W4	W 5	W 6	W 7	W 8	W 9	W 10	W 11
0.00 01	0.00 05	0.0 01	0.0 05	1	2	3	4	5	6	7

Weight Matrix for each model is calculated by substituting the rank of the model by corresponding rank weight as:

$$W = \text{Mat} [w_{ij}] = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \dots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}$$

D. Weighted Criteria Matrix

The weighted rank value of each model was obtained by multiplying the rank of each model with its weight for each criterion.

$$WR = \text{Mat}[rw_{ij}] = \text{Mat}[r_{ij}] * \text{Mat}[w_{ij}]$$

$$\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \dots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{bmatrix} *$$

$$\begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \dots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}$$

E. Weighted Estimation Function

The weighted estimation function is calculated based on the difference in estimation and observed values. We considered the deviation from 0 (for the same value) to 10, giving each model a weight vector of 11 values. Calculated deviation determines which model provides the estimation with values closer to observed values.

F. Rank Evaluation

The model, which is the best estimator and has a low rank in most criteria, is considered the best model. All the models were ranked based on this assumption by dividing the row summation of the weighted criteria matrix by weighted estimation function of that particular model as:

$$R_j = \frac{\sum_{i=1}^n WR_{ij}}{\sum WF_j}, \text{ where } j = 1 \text{ to } m$$

V. ILLUSTRATED EXAMPLE

We used the Fault data set, and parameters value used by Rajpal Garg et al. [15], shown in TABLE III and TABLE IV. We used thirteen comparison criteria discussed above to rank eleven different SRGMs having two or three parameters.

TABLE III. FAULT DATASET

Time (In weeks)	Observed Faults	Cumulative Faults
1	20	20
2	25	45
3	14	59
4	20	79
5	8	87
6	12	99
7	20	119
8	5	124
9	2	126
10	7	133
11	1	134
12	8	142
13	16	158
14	6	164
15	3	167
16	2	169
17	1	170
18	2	172
19	4	176
20	1	177
21	20	197

TABLE IV. PARAMETERS VALUE

MODELS	Parameter1	Parameter2	Parameter3
GO	2.157630e+2	1.08e-1	-
YMD	1.907960e+2	2.96e-1	-
MO	1.130030e+2	2.30e-1	-
GG	1.853600e+2	8.00e-4	3.1005
GMPZ	1.917870e+2	2.42e-1	5.97e-2
LGC	1.883490e+2	3.32e-1	7.21
YMI	2.033070e+2	1.55e-1	5.24e-1
PZIF	1.907950e+2	2.96e-1	1.00e-5
MD	2.375810e+2	4.0437e+1	4.096
YIDM1	128	1.89e-1	2.47e-2
YIDM2	128	1.91e-1	3.26e-2

The criteria values for each of the thirteen criteria reflected in TABLE V were calculated using MATLAB-2020b.

Based on the values of these criteria, each model's ranking for individual measures is determined and shown in TABLE VI.

Next, we assigned the weights so that the weightage given to low ranks when added up for different criteria for a specific model remains small compared to others. The weight matrix is evaluated according to the weight vector and shown in Table VII.

The weight matrix is then multiplied by the rank matrix to get a weighted rank-sum of a model as shown in TABLE VIII. This matrix is evaluated to ensure that models with consistent performance for all criteria must be given low rank.

Even though the Weighted matrix gives the rank according to the consistent performance of a model against set criteria, but it may happen that the two models still have equal weighted grades because of specific rank combinations. The best-ranked model may not be the best estimator for the same set of failure data and parameters.

TABLE V. COMPARISON CRITERIA'S VALUE

MODEL S	BIAS	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	R ²	AE	TS
GO	-0.46	24.37	2.97	60.76	22.96	4.94	0.92	0.76	0.71	0.00	0.42	0.07	0.49
YMD	-0.33	33.59	3.54	773.71	16.09	5.80	0.81	0.90	0.66	0.06	0.19	0.05	0.57
MO	-0.27	23.70	3.12	19.71	22.62	4.87	1.23	0.75	0.88	0.04	0.43	0.04	0.48
GG	-0.39	95.80	6.05	2890560.00	12.12	9.79	1.18	1.52	1.41	0.06	-1.30	0.06	0.96
GMPZ	-12.34	315.07	12.34	786.07	14.46	17.75	3.33	2.75	1.78	0.49	-6.56	1.88	1.75
LGC	-1.20	41.41	3.95	2229.61	12.31	6.44	1.14	1.00	0.74	0.00	0.01	0.18	0.63
YMI	12.34	289.00	12.78	11.12	21.15	17.00	2.95	2.63	4.55	0.21	-5.93	1.88	1.67
PZIF	-0.33	33.59	3.54	773.45	16.09	5.80	0.81	0.90	0.66	0.93	0.19	0.05	0.57
MD	14298.55	332608000.00	14298.55	20.98	36485.31	18237.54	3.15	2824.27	3415.26	11.61	7976513.00	2177.44	1794.39
YIDM1	-6.53	102.35	6.53	4272453.00	2.62	10.12	270.97	1.57	0.69	0.02	-1.45	0.99	1.00
YIDM2	-0.49	23.23	3.04	20.03	22.38	4.82	1.16	0.75	0.81	37.25	0.44	0.08	0.47

TABLE VI. RANKING OF DIFFERENT MODELS BASED ON A SPECIFIC CRITERION

MODELS	BIAS	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	Rsquare	AE	TS
GO	5	3	1	5	10	3	3	3	4	2	3	5	3
YMD	2	4	4	7	6	4	1	4	1	6	4	2	4
MO	1	2	3	2	9	2	7	2	7	4	2	1	2
GG	4	7	7	10	2	7	6	7	8	5	7	4	7
GMPZ	10	10	9	8	4	10	10	10	9	8	10	10	10
LGC	7	6	6	9	3	6	4	6	5	1	6	7	6
YMI	9	9	10	1	7	9	8	9	10	7	9	9	9
PZIF	3	5	5	6	5	5	2	5	2	9	5	3	5
MD	11	11	11	4	11	11	9	11	11	10	11	11	11
YIDM1	8	8	8	11	1	8	11	8	3	3	8	8	8
YIDM2	6	1	2	3	8	1	5	1	6	11	1	6	1

TABLE VII. WEIGHT MATRIX

MODELS	BIAS	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	Rsquare	AE	TS
GO	1.0000	0.0010	0.0001	1.0000	6.0000	0.0010	0.0010	0.0010	0.0050	0.0005	0.0010	1.0000	0.0010
YMD	0.0005	0.0050	0.0050	3.0000	2.0000	0.0050	0.0001	0.0050	0.0001	2.0000	0.0050	0.0005	0.0050
MO	0.0001	0.0005	0.0010	0.0005	5.0000	0.0005	3.0000	0.0005	3.0000	0.0050	0.0005	0.0001	0.0005
GG	0.0050	3.0000	3.0000	6.0000	0.0005	3.0000	2.0000	3.0000	4.0000	1.0000	3.0000	0.0050	3.0000
GMPZ	6.0000	6.0000	5.0000	4.0000	0.0050	6.0000	6.0000	6.0000	5.0000	4.0000	6.0000	6.0000	6.0000
LGC	3.0000	2.0000	2.0000	5.0000	0.0010	2.0000	0.0050	2.0000	1.0000	0.0001	2.0000	3.0000	2.0000
YMI	5.0000	5.0000	6.0000	0.0001	3.0000	5.0000	4.0000	5.0000	6.0000	3.0000	5.0000	5.0000	5.0000
PZIF	0.0010	1.0000	1.0000	2.0000	1.0000	1.0000	0.0005	1.0000	0.0005	5.0000	1.0000	0.0010	1.0000
MD	7.0000	7.0000	7.0000	0.0050	7.0000	7.0000	5.0000	7.0000	7.0000	6.0000	7.0000	7.0000	7.0000
YIDM1	4.0000	4.0000	4.0000	7.0000	0.0001	4.0000	7.0000	4.0000	0.0010	0.0010	4.0000	4.0000	4.0000
YIDM2	2.0000	0.0001	0.0005	0.0010	4.0000	0.0001	1.0000	0.0001	2.0000	7.0000	0.0001	2.0000	0.0001

TABLE VIII. WEIGHTED RANK MATRIX

MODELS	BIAS	MSE	MAD	PRR	NOISE	RMSE	RAE	RRSE	MMRE	PP	R ²	AE	TS
GO	5.000	0.003	0.000	5.000	60.000	0.003	0.003	0.003	0.020	0.001	0.003	5.000	0.003
YMD	0.0010	0.0200	0.0200	21.0000	12.0000	0.0200	0.0001	0.0200	0.0001	12.0000	0.0200	0.0010	0.0200
MO	0.0001	0.0010	0.0030	0.0010	45.0000	0.0010	21.0000	0.0010	21.0000	0.0200	0.0010	0.0001	0.0010
GG	0.0200	21.0000	21.0000	60.0000	0.0010	21.0000	12.0000	21.0000	32.0000	5.0000	21.0000	0.0200	21.0000
GMPZ	60.0000	60.0000	45.0000	32.0000	0.0200	60.0000	60.0000	60.0000	45.0000	32.0000	60.0000	60.0000	60.0000
LGC	21.0000	12.0000	12.0000	45.0000	0.0030	12.0000	0.0200	12.0000	5.0000	0.0001	12.0000	21.0000	12.0000
YMI	45.0000	45.0000	60.0000	0.0001	21.0000	45.0000	32.0000	45.0000	60.0000	21.0000	45.0000	45.0000	45.0000
PZIF	0.0030	5.0000	5.0000	12.0000	5.0000	5.0000	0.0010	5.0000	0.0010	45.0000	5.0000	0.0030	5.0000
MD	77.0000	77.0000	77.0000	0.0200	77.0000	77.0000	45.0000	77.0000	77.0000	60.0000	77.0000	77.0000	77.0000
YIDM1	32.0000	32.0000	32.0000	77.0000	0.0001	32.0000	77.0000	32.0000	0.0030	0.0030	32.0000	32.0000	32.0000
YIDM2	12.0000	0.0001	0.0010	0.0030	32.0000	0.0001	5.0000	0.0001	12.0000	77.0000	0.0001	12.0000	0.0001

Weights gradually decrease by two points for each increased variation from 20 to 1.

To overcome this problem, we computed a weighted function based on estimation values, and undermentioned Table X gives the weighted function values of different models. Zero value is replaced by 0.0001 for computation purposes.

The estimation values for all the models were calculated and formatted to number values without any decimal part. The Estimation function is then calculated, giving the highest weight to zero deviation and lowest to ten point deviation.

The rank of a model based on set criteria and estimation capabilities is then calculated by dividing the weighted sum value by the weighted estimation function value. The models are then arranged according to their rank in Table XI, where the lowest rank value specifies the best model for selected fault data.

TABLE IX. WEIGHTED ESTIMATION FUNCTION

MODELS	Estimation Function
GO	255
YMD	230
MO	249
GG	184
GMPZ	73
LGC	219
YMI	48
PZIF	230
MD	0.0001
YIDM1	160
YIDM2	249

TABLE X. MODEL'S RANKING

S. No.	Models	Rank
1	GO	2
2	YMD	1
3	MO	3
4	GG	7
5	GMPZ	9
6	LGC	6
7	YMI	10
8	PZIF	4
9	MD	11
10	YIDM1	8
11	YIDM2	5

VI. CONCLUSION

Various researchers developed ranking models based only on the specific criteria set in the past. This paper specified a comparison criterion to determine the model's rank. The comparison criteria consist of thirteen criteria generally used in the literature to determine the performance and ranking of models. We developed a ranking methodology by specifying various matrices. Criteria Matrix and Rank Matrix represent the criteria values and rank of eleven models against each measure. Weight matrix assigns different weights to models rank in such a way that models showing good consistent performance must be assigned a low value. The weighted estimation function matrix gives the weighted sum of the number of deviations of model estimation with observed value, ranging from 0 to 19 for each model. The summation of the weighted rank of each model for all thirteen criteria is then divided by the weighted estimation function of that specific model to give us the best model, which has the lowest rank and better estimation capabilities than other models.

No model is optimal for all applications, and we can select the best-suited model by specifying criteria tailored to our requirements. The rank calculation methodology adopted in this paper involves simple calculations and is easy to use. The model can be modified by including more relevant values into the criteria set, and weights can be assigned depending on measures set values regarding the problem. The only limitation of our methodology is the inclusion of reliability models up to three parameters, and one can also generalize the method for reliability models with more than three parameters.

REFERENCES

- [1] S. Khurshid, A. K. Shrivastava, and J. Iqbal, "Effort based software reliability model with fault reduction factor, change point and imperfect debugging," *Int. J. Inf. Technol.*, vol. 13, no. 1, pp. 331–340, 2021, doi: 10.1007/s41870-019-00286-x.
- [2] M. Anjum, M. A. Haque, and N. Ahmad, "Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value," *Int. J. Inf. Technol. Comput. Sci.*, vol. 5, no. 2, pp. 1–14, 2013, doi: 10.5815/ijitcs.2013.02.01.
- [3] D. Sharma and P. Chandra, "A comparative analysis of soft computing techniques in software fault prediction model development," *Int. J. Inf. Technol.*, vol. 11, no. 1, pp. 37–46, 2019, doi: 10.1007/s41870-018-0211-3.
- [4] M. Xie, "Software Reliability Models for Practical Applications," pp. 211–214, 1995, doi: 10.1007/978-0-387-34848-3_32.
- [5] S. S. Gokhale, P. N. Marinos, and K. S. Trivedi, "Important milestones in software reliability modeling," *Seke*, pp. 345–352, 1996.
- [6] S. S. Gokhale, W. E. Wong, J. R. Horgan, and K. S. Trivedi, "An analytical approach to architecture-based software performance and reliability prediction," *Perform. Eval.*, vol. 58, no. 4, pp. 391–412, 2004, doi: 10.1016/j.peva.2004.04.003.
- [7] K. Sharma, R. Garg, C. K. Nagpal, and R. K. Garg, "Selection of optimal software reliability growth models using a distance based approach," *IEEE Trans. Reliab.*, vol. 59, no. 2, pp. 266–276, 2010, doi: 10.1109/TR.2010.2048657.
- [8] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Trans. Reliab.*, vol. R-28, no. 3, pp. 206–211, 1979, doi: 10.1109/TR.1979.5220566.
- [9] S. Yamada and S. Osaki, "Software Reliability Growth Modeling: Models and Applications," *IEEE Trans. Softw. Eng.*, vol. SE-11, no. 12, pp. 1431–1437, 1985, doi: 10.1109/TSE.1985.232179.
- [10] S. Yamada, H. Ohtera, and H. Narihisa, "Software Reliability Growth Models with Testing-Effort," *IEEE Trans. Reliab.*, vol. 35, no. 1, pp. 19–23, 1986, doi: 10.1109/TR.1986.4335332.
- [11] P. Govindasamy and R. Dillibabu, "Development of software reliability models using a hybrid approach and validation of the proposed models using big data," *J. Supercomput.*, vol. 76, no. 4, pp. 2252–2265, 2020, doi: 10.1007/s11227-018-2457-8.
- [12] K. Pillai and V. S. Sukumaran Nair, "A model for software development effort and cost estimation," *IEEE Trans. Softw. Eng.*, vol. 23, no. 8, pp. 485–497, 1997, doi: 10.1109/32.624305.
- [13] C. Y. Huang and S. Y. Kuo, "Analysis of incorporating logistic testing-effort function into software reliability modeling," *IEEE Trans. Reliab.*, vol. 51, no. 3, pp. 261–270, 2002, doi: 10.1109/TR.2002.801847.
- [14] K. C. Chiu, Y. S. Huang, and T. Z. Lee, "A study of software reliability growth from the perspective of learning effects," *Reliab. Eng. Syst. Saf.*, vol. 93, no. 10, pp. 1410–1421, 2008, doi: 10.1016/j.ress.2007.11.004.
- [15] R. Garg, K. Sharma, R. Kumar, and R. K. Garg, "Performance analysis of software reliability models using matrix method," *World Acad. Sci. Eng. Technol.*, vol. 71, pp. 31–38, 2010, doi: 10.5281/zenodo.1330575

Hybrid Software Reliability Model for Big Fault Data and Selection of Best Optimizer Using an Estimation Accuracy Function

Ms. Shalini Sharma¹, Dr. Naresh Kumar², Dr. Kuldeep Sing Kaswa³

¹School of Computing Science and Engineering

Galgotias University, Greater Noida, India

Email Id: shalini.sharma@rediffmail.com

²School of Computing Science and Engineering

Galgotias University, Greater Noida, India

Email Id: Naresh.dhull@gmail.com

³School of Computing Science and Engineering

Galgotias University, Greater Noida, India

Email Id: kaswankuldeep@gmail.com

Abstract— Software reliability analysis has come to the forefront of academia as software applications have grown in size and complexity. Traditionally methods have focused on minimizing coding errors to guarantee analytic tractability. This causes the estimations to be overly optimistic when using these models. However, it is important to take into account non-software factors, such as human error and hardware failure, in addition to software faults to get reliable estimations. In this research, we examine how big data systems' peculiarities and the need for specialized hardware led to the creation of a hybrid model. We used statistical and soft computing approaches to determine values for the model's parameters, and we explored five criteria values in an effort to identify the most useful method of parameter evaluation for big data systems. For this purpose, we conduct a case study analysis of software failure data from four actual projects. In order to do a comparison, we used the precision of the estimation function for the results. Particle swarm optimization was shown to be the most effective optimization method for the hybrid model constructed with the use of large-scale fault data.

Keywords. Software Reliability, Hybrid models, Parameter valuation methods, Soft computing optimization

I. INTRODUCTION

The failure-free operation of software under predetermined times and conditions is referred to as software reliability. Before a piece of software is released, its reliability is checked, which if not managed properly can result in software failure making reliability a crucial factor in software development. If not managed effectively, software defects can result in software failures making reliability prediction a crucial task. There are many models to handle reliability that are available in the literature [2-15], but when big data analytical systems were utilized these models didn't provide accurate predictions [3]. Software reliability prediction has significantly improved when using a hybrid reliability model rather than a classic one. The performance of models depends heavily on the kind of parameter evaluation technique, making it difficult to choose the optimal strategy among the statistical and soft computing approaches.

Hundreds of software reliability growth models have been proposed by scholars in the past (SRGM). An SRGM is a mathematical model that accurately predicts experimental data

and is created by tracking the probability density function or growth curve [15]. The failure pattern of the system is used by SRGM to assess the software's reliability. These failure patterns and data trends are used for reliability estimation. Non-homogeneous Poisson Process (NHPP) models and failure rate models are the two categories under which SRGMs are categorized.

The software is treated as an internal structure with interactions from the external world in NHPP models, known as black-box models. The defect information gathered during testing is used to assess the model's parameters and reliability aspects. The primary need for a software model with a good fit is precise parameter evaluation. The parameter values of dependability models are frequently determined using statistical methods like LSE and MLE. LSE is a minimization strategy that reduces the minimal sum of the squared deviation between the estimated and observed value, which is evaluated by fitting a regression line using data points that satisfy the LSE property. In contrast, MLE determines the parameter value that optimizes the function. MLE is a necessity for the chi-square test Bayesian approaches, modeling of random effects, and

inference with missing data since it is sufficient, consistent, efficient, and parameter invariant.

When the density function is complex, nonlinear, and involves a large number of factors, it is difficult to determine parameter values. In such circumstances, we must numerically determine the parameter value that minimizes LSE or maximizes MLE using optimization techniques such as Newton, quasi-Newton, Gauss-Newton, and Levenberg-Marquardt. Due to the nonlinearity and complexity of the model function, it was difficult to estimate the parameter's value because the model didn't provide a single value for a collection of parameters for several guess values. Additionally, with more parameters, it became too difficult to check all possible permutations of estimate values for an ideal answer. Utilizing soft computing techniques to resolve reliability analysis optimization issues has been popular recently. For parameter evaluation, a variety of bio-inspired techniques such as Particle Swarm Optimization, Cuckoo Search, Grey Wolf, Ant Colony, Artificial Bee Colony, and many more are used alone or in combination with genetic algorithms, neural networks, exponential logistic techniques, and traveling salesman problem.

In this study, we created a hybrid model in which resulting faults were not only because of coding errors but also due to induced error in the software brought on by hardware failure or human error. For nonlinear prediction models, a non-homogeneous reliability model (NHPP) is developed. The suggested methodology combines three NHPP models to assess programming faults and other caused software defects. NHPP models heavily rely on time, fault data, and error rate function to assess dependability and employ mean value function to quantitatively characterize the failure phenomenon. These models perform parameter evaluation through the use of least square estimation (LSE) or maximum likelihood estimation (MLE) methodologies. The primary goal is to assess the value of the mathematical model's unknown parameters, which gives minimum error in the output. The optimal outcome is not always achieved using statistical approaches, despite their being the most popular methodologies. When dealing with complex nonlinear equations with many unknown parameters, soft computing approaches are preferred to statistical methods for parameter estimation.

In this paper, we used GA, SA, and PSO methods for parameter evaluation using MATLAB 2022b. Experiments were performed using four datasets and seven comparison criteria. Comparison is carried out to determine the best optimizer for the developed model using Big Data.

The Paper is organized as follows. Section II discusses the literature survey of the hybrid models used for big data. In Section III literature survey of developed models for big data is

discussed. In section IV various techniques applied in this paper to obtain results were briefly discussed. Section V discusses the development of a hybrid model along with the methodology used. Section VI presents the experimental work and obtained results. Finally, Section VII concludes the paper.

II. LITERATURE SURVEY

Big data processing requires a new class of specialized hardware and software due to its unique properties. Simple software processing errors, data errors, and hardware issues lead to erroneous, compromised results, and subpar performance [1].

Big data reliability is further divided into three categories: hardware reliability, software reliability, and data reliability. The system may gather big data from a variety of sources, including sensors, IoT devices, cell phones, scanners, CVS, sensors, social media, logs, census data, RDBMS, etc. When content is ingested from a variety of sources, its accuracy and completeness are depicted as data reliability. Program failures can be caused by a misreading of the specifications, insufficient testing, a coding error, or improper software usage. The probability of errors grows in line with the exponential expansion in data volume. Therefore, in order to get accurate results, a reliability assessment of the data in use is required. Contrarily, hardware reliability results from an inadequately built system, which eventually causes performance to decline and fails to reach the required standards. Operational and architectural components make up hardware reliability. Operational reliability uses trustworthy statistical techniques to forecast equipment failure in the field and system performance using large data from modern reliability that is obtained through IoT or sensor devices attached to the system. Architectural reliability deals with the quick data retrieval from large capacity storage medium necessary for voluminous data, making it a crucial component in the successful execution of a big data project. Since data analysis is being done to support decision-making, we must employ a reliability model to cross-check for any potential flaws that could have caused the system to produce inaccurate predictions. Researchers created a variety of models to handle the reliability of big data by accounting for the external interactions of large data for precise prediction.

Han, Tian, Yoon & Lee (2012) [17] after analyzing massive amounts of data from social networks, created a big data model using Big Table and Map Reduce. Meeker and Hong in 2013 [18] identified a number of applications that make use of field reliability data, such as warranty, degradation, lifetime, and recurrence field data, and investigated opportunities to assess reliable statistical techniques for predicting the performance of systems in the field. Chang Liu et al. (2014) [19] proposed the use of the Boneh-Lynn-Shacham (BLS) signature and the Multiple Huffman Table (MHT) for authorized audit in

a public auditing system. Tamura, Miyaoka & Yamada (2014) [20] used a three-dimensional stochastic differential equation (3D-SDE) to assess the reliability of open software systems on the cloud, assuming irregular and time-dependent fault reporting during the operating phase. The model accounts for mistakes resulting from interactions between network software, open software, and big data software as well as the software fault factor, the big data factor, and the network factor. Tamura & Yamada, (2014) [21] presented a model for 3-D stochastic differential reliability. Based on three key features of big data, parameters were evaluated using the Analytical Hierarchy approach. Kwon, Lee & Shin in 2014 [22] considered that IT capabilities like data quality management and data consumption experience have a substantial impact on the desire to acquire big data analytics, and better-quality management boosts data usage regardless of its source. Li et al. (2014) [23] proposed a model to resolve conflicts between numerous big data sources with diverse structures by using an optimized framework termed CRH of two variables: truths and source reliability, where truth is defined as the value responsible for the minimum possible departure from multi-source inputs and the weights denote the degree of reliability. Tamura and Yamada (2015) [24] presented a hazard rate and clustering method for huge data situations using cloud computing that is based on SRM. They concentrated on the operation phase reliability of cloud computing with big data and created an Application for Reliability Assessment (AIR) employing cluster analysis of fault data. Tamura and Yamada (2015) [25] offered a different way to assess the dependability of cloud computing. By employing a jump-diffusion model with stochastic differential equations and two-dimensional Weiner processes, they anticipated software costs. They also define an optimal maintenance issue using a sample path while taking the level of noise into account. OSS reliability assessment (RA) methodologies were created using big data to measure component and system-level reliability. Tamura and Yamada (2015) [26] employed a hazard rate model composed of stochastic equations for RA using a data set that included cumulative faults and temporal gaps between failures. An SRM based on k-means clustering and neural networks was proposed by Tamura, Nobukawa, and Yamada, 2015 [27]. Utilizing cluster analysis findings on fault datasets gathered from databases, including Hadoop and NoSQL, and cloud applications, like Eucalyptus and OpenStack, NN was utilized to estimate cumulative faults. By integrating the human error effects with traditional PRA techniques and administrative considerations, Pence et al., 2015 [28] proposed a theoretical methodology based on Socio-technical Risk Analysis "SoTeRia" for quantifying the organizational mechanism for performance shaping factors (PSF) in human resources. A quality assessment methodology for big data was developed by

Cai and Zhu in 2015 [29] using a feedback mechanism that specifies common quality components and their corresponding indicators. A model for trustworthy network data mining was created by Li, He, and Ma in 2016 [30] utilizing an updated PageRank algorithm. Hu, Liu, Diao, Meng, and Sheng 2016 [31] proposed a model to leverage big data to identify the operational reliability issue of the power distribution system. The model was evaluated using NN after applying parallel index rule mining to analyze the influential aspects connected to the reliability index. A framework and model focused on research were developed by Spichkova et al. in 2016 [32] employing the cloud computing platform's usability and reliability capabilities. Researchers who are exploring huge data and enormous computations can use the model. The chimney platform has undergone testing in the fields of structural biology and physics. A maintenance issue and approach were put out by Tamura and Yamada in 2017 [33] to assess component reliability on the cloud platform. Through the use of a Genetic Algorithm (GA) and NN for RA, model parameters were assessed. Yan, Meng, Lu & Li (2017) [34] put forth a method as a framework for structuring large data (gathered from diverse sources with heterogeneous information) with attention to spatiotemporal factors. To make the production process clear, they simulate a variety of invisible elements, including those linked to energy conservation and preventive maintenance. In order to study the best parallel recovery strategies for replication and random and shifted 45 multi-way de-clustering data layouts, Wang, Wu, and Wang (2017) [35] built a reliability model. Nachiappan, Javadi, Calherios & Matawie (2017) [36] investigate the replication and Erasure approaches in cloud storage for massive data and list their difficulties. A text classifier was created by Xiang, Du, Ma, & Fan in 2017 [37] to assess the validity of online hotel reviews. Tamura and Yamada (2017) [38] created a deep learning-based RA model for open-source software. They created a tool that uses fault datasets to access OSS reliability. The suggested technique uses deep learning and a hazard rate model to accomplish reliability estimation. The current modeling and reliability analysis developments with reference to complicated dimension structures were explored and reviewed in 2018 by Hong, Zhang, and Meeker [39]. In 2018 Cao and Gao [40] created an SRM for big data systems utilizing fault tree analytics (FTA). FTA assesses the overall system dependability, serves as a benchmark for quality assurance, and reviews a module qualitatively to assess its likelihood of failing. Yaremchuk and Kharchenk (2018) [41] established a similarity model to spot software plagiarism and identify copies, as well as a number of other techniques and tools to spot the augmentation of program reliability. Three hybrid dependability models were created by Govindasamy and Dillibabu (2018) [3] by integrating previously existing NHPP models. With the aid of comparison

criteria, models were validated. In order to evaluate the parameters, MLE and GA were utilized. In 2022, Wang, Zhang, and Yang [42] proposed a RAM based on the presumption that the rate of fault introduction for Open System Software (OSS) will gradually decline. considering fault severity levels (CFSL) in OSS was the subject of a hazard rate model published by Yanagisawa, Tamura, Anand, & Yamada (2022) [43]. Their study's goal was to create a Hazard rate model for two types of fault data in the Bug Tracking System using covariate vectors and CFSL adaptive to baseline hazard function.

III. OVERVIEW OF APPLIED SOFTWARE RELIABILITY MODELS AND PARAMETER ESTIMATION TECHNIQUES

NHPP models in three to create the hybrid model, Duane, Exponential, and PZIFD were used. The parameters were then assessed using statistical techniques MLE and LSE, and they were then further improved using soft computing techniques GA, SA, and PSO utilizing seven criteria values. The majority of SRGMS were built around a non-homogeneous poisoning process (NHPP). NHPP models are actual processes that make use of stochastic methodologies to estimate the reliability of a system utilizing appropriate interpretations from software testing and debugging. The mean value function (MVf) follows the Poisson distribution and the total number of defects is denoted by $m(t)$. Different MVFs can be used to create various NHPP models, or the current MVF of NHPP models can be modified. Successful modeling of a software failure process is only possible after examining the testing process's variables. The MVF and intensity function (InF) of software reliability models utilized in hybrid model development are shown in Table 1 below.

Table I. NHPP models used in Hybrid model development

Models	Mean Value Function	Intensity Function
Exponential Model [47]	$m(t) = at$	$\lambda(t) = a$
Duane Model (DM) [48]	$m(t) = at^b - a$ $> 0, b > 0$	$\lambda(t) = abt^{(b-1)}$
Wang-Zhang Imperfect Fault Detection Model (PZIFD) [49]	$m(t) = a e^{-bt} (1 + (b + d)t + bdt)$	$\lambda(t) = ae^{-bt} [bt(b - d) + d(b^2t^2 - 1)]$

A. Parameter Estimation Techniques

The correct value of the parameters is a prerequisite for both the reliability estimation and the model validation. Large datasets and statistical methods provide a numerical estimation of the parameter values, which is then used to assess the model's goodness of fit. Two of the most used statistical techniques for parameter estimation in reliability modeling are MLE and LSE. Because of qualities like consistency, sufficiency, and parameter invariance, MLE is a favored standard approach that is frequently utilized. LSE is typically assessed using linear regression, a sum of squared errors, a root mean square deviation, and R^2 and is a viable option for linear models with medium or small samples (proportion variance). By minimizing the discrepancy between the observed and estimated values, LSE evaluates the parameters.

B. Least Square Estimation

As a result of its lower bias or faster normalcy method, the LSE excelled in predicting small data sets. The relative inaccuracy between observed and estimated values is what LSE attempts to reduce. Calculating a model's intensity function value or rate of error yields the estimated values of the model.

$$LSE = \sum_{i=1}^n ((\lambda(t)_{Eval} - \lambda(t)_{val})^2 \tag{1}$$

C. Maximum Likelihood Estimation

The MLE technique selects parameter values that maximize the loglikelihood function, and the likelihood function L is obtained by utilizing the intensity function (t) as the input $\lambda(t)$

$$L = \prod_{i=1}^n \lambda(t_i) \tag{2}$$

and loglikelihood function is given as (Pham,2006)

$$\text{Log } L = \log (\prod_{i=1}^n \lambda(t_i)) \tag{3}$$

$$= \sum_{i=1}^n \log (\lambda(t_i) - m(t_n)) \tag{4}$$

The preceding equation is differentiated m times for each parameter with respect to time for a model with m parameters, producing m equations to solve. These equations are then solved using algorithms for “ m ” equations. As a result, the accuracy of the parameter now hinges on how well these approaches can avoid local minima and determine appropriate values.

A growing trend in parameter value optimization is soft computing. Numerous researchers have recently used a variety of soft computing techniques, including GA, NN, Fuzzy Logic, Support Vector Machines, Particle Swarm, Ant Colony, Gray Wolf, Cuckoo Search, Sparrow Search, and Artificial Bee Colonies, among others, for software engineering optimization

and reliability assessment. Soft computing techniques take advantage of uncertain, approximative, incomplete, and imprecise behavior to produce useful, economical dependable models. We utilized the techniques listed below to optimize the parameter value.

D. Genetic Algorithm

GA is a simulation of the process of species creation that is based on natural selection in biology. Chromosomes make up a population, and GA creates new populations by choosing chromosomes from the present generation and creating new ones through mutation while employing a fitness function [44]. The creation of new chromosomes continued until a halting threshold was reached. GA is irrational in nature and uses the past to solve a problem.

E. Particle Swarm Optimization

PSO is a population-based metaheuristic optimization algorithm that draws inspiration from nature. It starts out as a swarm of randomly distributed particles that represent a potential solution. Each particle calculates its position based on its velocity, prior location, and adjacent objects, which helps shape the swarm's overall behavior [16]. Every particle in the search space is an alternative solution that has fitness for the objective function. While classical PSO converges quickly, it is simple to get stuck in local minima for complex problems, which results in premature convergence. PSO is more adaptable than GA and has the ability to manage and balance both local and global search space exploration.

F. Simulated Annealing

When there are several local optima, SA is another optimization method that can be used to locate global optima. The term "annealing" is derived from thermodynamics, which describes heating and cooling metal to change its physical characteristics due to internal structural changes. Metal keeps its new structure and characteristics after cooling. Instead of material energy, SA provides the problem's objective function that needs to be maximized. To replicate the heating and cooling processes, the temperature is handled as a variable in SA. When the temperature is high, the algorithm accepts more solutions more frequently, enabling it to exit any local optimum. As the temperature drops, it becomes more likely that one will accept a suboptimal answer, compelling the algorithm to concentrate on the region of the search space where the near-optimal solution can be located. For huge, complicated problems with many local optimums, SA is an effective method that uses a slow cooling process to locate the solution that is close to the ideal.

G. Comparison Criteria

We used five comparison criteria as an objective function in GA, PSO, and SA methods to optimize the value of the parameter. These five criteria values were taken from the literature [3] and tabulated in Table II.

Table II. Comparison Criteria

Criteria	Formula
Mean Square Error (MSE)	$\sum_{i=1}^n \frac{(Oval(i) - Eval(i))^2}{n}$
Mean Absolute Deviation (MAD)	$\sum_{i=1}^n Oval(i) - Eval(i) $
Root Mean Square Error (RMSE)	$\sqrt{\sum_{i=1}^n \frac{(Oval(i) - Eval(i))^2}{n}}$
Mean Magnitude Of Relative Error (MMRE)	$\frac{1}{n} \sum_{i=1}^n \left \frac{Oval(i) - Eval(i)}{Oval(i)} \right $
Predictive Power (PP)	$\left(\frac{Eval(i) - Oval(i)}{Oval(i)} \right)^2$

IV. HYBRID RELIABILITY MODEL

A good model is regarded as dependable if it is straightforward, widely applicable, and successfully forecasts future failures. NHPP models are useful for figuring out a model's combined software and hardware reliability. By integrating the MVFs or InF of two (or more) NHPP models, one can create an NHPP hybrid model. The generated hybrid model's MVF is represented by the resulting combined MVF (or InF). So, by merging different NHPP models with significant parameter-set, we may create a hybrid model that provides a decent match but requires more computations and has lower dependability confidence. We created a hybrid model by merging the PZIFD model, the Duane model, and the exponential model to cover pure software faults, hardware-induced software errors, and manual-induced software bugs. Given below is the hybrid model's combined MVF and InF with its six parameters (a, b, d, x, y, and p).

$$m(t) = a - a e^{-bt} (1 + (b + a)t + bdt^2) + \lambda t^x + pt \quad (5)$$

$$\lambda(t) = \alpha e^{-\alpha t} [bt(b - d) + d(b^2t^2 - 1)] + \alpha y t^{(y-1)} + p \quad (6)$$

A. Methodology

In this study, we constructed a hybrid model and used five different approaches to identify its parameters: two statistical and three soft computing-based. To optimize the parameter value utilizing GA, SA, and PSO, we use five comparison criteria. We evaluate the intensity function using all parameter values in order to find the optimal optimizer for the

hybrid model. We then compute a weighted estimation function [1]. Weights ranged from 20 to 1 and were gradually reduced by 2 points using a ten-point error deviation. When there is no error and there is a 10 percent difference between the observed and experimented values, weight 20 is assigned. The following is the weighted estimation function:

$$F=20e^0+15e^1+16e^2+14e^3+12e^4-10e^5+8e^{-6}+6e^7+4e^8+2e^9+1e^{10} \quad (6)$$

7. EXPERIMENT WORK AND RESULT DISCUSSION

Two statistical approaches (MLE and LSE) and three soft computing techniques (GA, SA, and PSO) were used in MATLAB 2022b to evaluate the parameters of the hybrid model. To optimize the parameters, five criteria values were employed as the goal function across all of the soft computing approaches. We used four different data sets to find the optimal optimizer for the hybrid model by analyzing fault data from a large analytical system tabulated in Table III. The first three datasets (DS1, DS2, and DS3) are from a Big Data system, whereas the fourth (DS4) is a failure dataset from a medium-sized project.

Table III. Dataset Used

Dataset	Project
DS1	Software Analytical project [7]
DS2	Fault data sets were collected from the Apache Storm project (STORM) [6].
DS3	Fault data sets collected from Apache Chemistry OpenCMIS project (OpenCMIS) [6].
DS4	data set (2008) collected during the testing process of a middle-size software project.[5]

Parameters of the hybrid model using statistical evaluation using MLE and LSE are shown below in Table IV.

Table IV. Parameters evaluated using Statistical Methods

DS1						
MLE	0.5	0.5	0.5	0.6	-0.3	-1380.9
LSQ	0.5112	0.4872	0.5075	0.7524	1.3978	2.7791
DS2						
MLE	0	0.2033	0.5161	0.0266	-6.4033	0.0303
LSQ	2.0346	0.1572	2.8603	0.1001	0.5559	5.4312
DS3						
MLE	0	0.3024	0.5191	0.0466	-7.4203	0.05
LSQ	1.9223	0.3857	1.1211	2.196	0.6543	0.1997
DS4						
MLE	0.5	0.5	0.5	0.5946	0.4035	-50.1346
LSQ	0.5261	0.3488	0.5225	2.0034	1.0563	5.723

Parameter values using GA, SA, and PSO using seven comparison values are tabulated below from Table V to Table VII for DS1 to DS4 respectively.

Table V. Parameter optimization using GA for DS1 to DS4

GA(DS1)						
MLE	4.10	0.50	24.74	12.64	-2.05	-0.02
LSE	69.07	0.13	-0.23	-861.34	0.11	12.29
MSE	0.94	0.13	9.70	16.52	0.45	8.19
MAD	5.00	8.47	33.71	0.34	1.66	3.11
RMSE	4.72	0.15	0.88	0.12	1.82	7.37
MMRE	12.54	8.99	16.53	0.46	0.21	3.97
PP	39.29	12.21	4.88	8.13	0.57	1.18
GA(DS2)						
MLE	3.01	2.76	-9.09	-9.95	-1.22	-0.09
LSE	39.53	9.06	3.25	-13.09	0.61	6.65
MSE	-2.57	5.17	-5.79	0.99	1.28	-1.61
MAD	10.00	9.99	27.97	-0.56	0.48	0.08
RMSE	-33.86	8.59	-7.70	-6.69	0.67	2.81
MMRE	5.33	-2.18	2.08	-8.78	-5.90	-0.96
PP	-7.40	-7.63	9.43	-9.26	9.67	3.02
GA(DS3)						
MLE	6.55	9.17	-9.99	2.89	0.48	-1.01
LSE	-1.66	4.46	3.90	6.54	0.13	0.77
MSE	-12.53	8.57	-16.72	10.32	-0.21	0.81
MAD	-3.28	8.14	-33.50	12.45	-0.08	0.25
RMSE	3.23	2.27	-2.29	-0.32	0.23	0.68
MMRE	4.02	0.72	-5.49	1.65	2.58	9.50
PP	-3.50	8.99	-9.56	9.02	5.01	-5.09
GA(DS4)						
MLE	-9.55	0.87	243.42	0.05	-0.07	0.04
LSE	-44.90	0.10	-1.10	-1330.40	-0.30	7.10
MSE	7.97	1.05	8.77	-6.57	-4.47	5.74
MAD	-5.71	0.06	3.95	0.61	-5.54	-4.14
RMSE	10.64	0.98	4.80	-38.35	-0.66	5.10
MMRE	9.81	1.04	7.29	-16.35	-2.05	1.76
PP	-11.92	0.97	-8.14	-17.56	-3.20	1.38

Table VI. Parameter optimization using SA for DS1 to DS4

SA(DS1)						
MLE	8.00	43.22	25.86	30.83	124.67	57.41
LSE	24.90	7.14	0.30	0.90	1.37	2.63
MSE	2.68	0.16	2.43	8.25	0.39	8.80
MAD	58.81	19.06	0.22	1.88	0.10	6.69
RMSE	111.35	46.84	35.73	0.20	1.46	9.55
MMRE	22.58	11.37	56.11	1.29	0.32	3.95
PP	2.25	49.75	10.29	14.91	0.27	2.11
SA(DS2)						
MLE	3.01	2.76	-9.09	-9.95	-1.22	-0.09
LSE	0.57	0.16	15.12	8.38	0.67	2.87
MSE	40.13	49.96	13.39	0.10	1.71	0.10
MAD	39.77	22.93	0.64	0.11	0.11	0.11
RMSE	3.36	51.89	22.13	0.27	1.49	0.14
MMRE	0.50	0.50	0.50	0.50	0.50	0.50
PP	0.50	0.50	0.50	0.50	0.50	0.50
SA(DS3)						
MLE	80.14	44.21	146.13	68.20	235.11	104.40
LSE	19.34	0.29	0.18	10.49	0.18	0.10
MSE	2.83	107.92	52.43	0.68	0.89	0.17
MAD	29.77	43.62	20.35	0.10	0.11	0.10
RMSE	0.50	0.50	0.50	0.50	0.50	0.50
MMRE	0.50	0.50	0.50	0.50	0.50	0.50

PP	0.50	0.50	0.50	0.50	0.50	0.50
SA(DS4)						
MLE	23.32	7.20	24.47	85.76	134.16	3.74
LSE	6.09	0.11	9.76	11.26	0.75	5.70
MSE	9.76	53.24	27.93	47.33	0.51	0.10
MAD	55.89	48.67	38.94	123.37	0.27	0.27
RMSE	37.53	55.12	66.70	52.01	0.40	3.10
MMRE	8.29	83.50	31.26	150.63	0.10	0.11
PP	29.70	58.07	4.05	72.79	0.19	0.12

Table VII. Parameter optimization using PSO for DS1 to DS4

PSO(DS1)						
MLE	419.90	1000.00	786.20	475.30	119.40	0.10
LSE	813.10	300.75	751.05	0.10	0.10	9.63
MSE	945.43	219.03	928.80	0.10	0.10	9.63
MAD	929.80	1000.00	988.60	0.10	0.10	6.00
RMSE	108.84	659.51	951.67	192.33	0.10	0.10
MMRE	362.99	801.54	443.62	22.21	0.10	3.78
PP	410.46	214.89	455.85	41.01	0.10	2.30
PSO(DS2)						
MLE	3.01	2.76	-9.09	-9.95	-1.22	-0.09
LSE	60.90	0.10	0.10	61.37	0.10	0.10
MSE	203.82	977.47	612.44	11.70	0.10	0.10
MAD	607.72	16.57	88.70	0.10	0.10	0.10
RMSE	0.10	0.10	41.31	71.11	0.10	0.10
MMRE	992.05	594.74	897.44	501.57	415.51	500.68
PP	797.76	78.95	529.52	99.82	117.05	610.10
PSO(DS3)						
MLE	1000.00	1000.00	531.80	632.00	230.40	1000.00
LSE	193.87	452.40	432.44	0.10	0.10	0.64
MSE	974.94	652.89	0.33	12.85	0.10	0.10
MAD	0.76	8.72	107.78	0.10	0.10	0.10
RMSE	1000.00	984.10	0.10	12.90	0.10	0.10
MMRE	510.90	4.44	925.50	757.34	118.85	827.10
PP	575.04	738.63	596.80	254.29	965.39	63.06
PSO(DS4)						
MLE	0.10	966.20	835.60	1000.00	112.70	952.90
LSE	475.60	0.10	0.10	0.10	0.10	0.10
MSE	611.16	464.80	970.08	191.25	0.10	5.31
MAD	290.37	471.35	539.47	341.57	0.10	1.70
RMSE	857.80	981.00	49.85	191.25	0.10	5.31
MMRE	824.46	959.24	468.38	150.92	0.10	0.10
PP	973.01	998.95	750.69	176.92	0.10	0.10

A Comparison between Evaluation methods for all datasets is carried out to determine the good performance of a method across all datasets as shown below in Fig. 1.

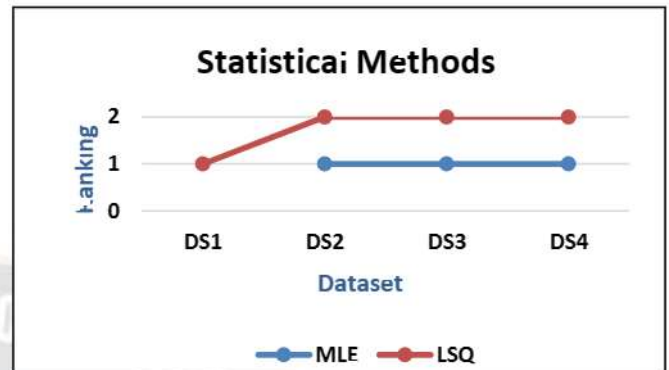


Figure 1. Comparison between statistical methods.

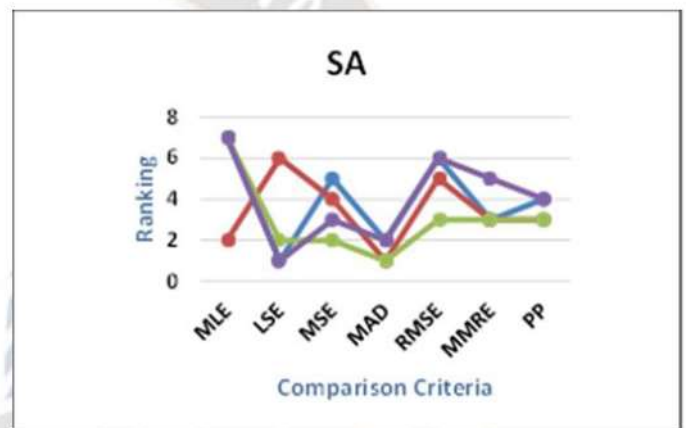


Figure 2. Ranking of GA values evaluated using comparison criteria

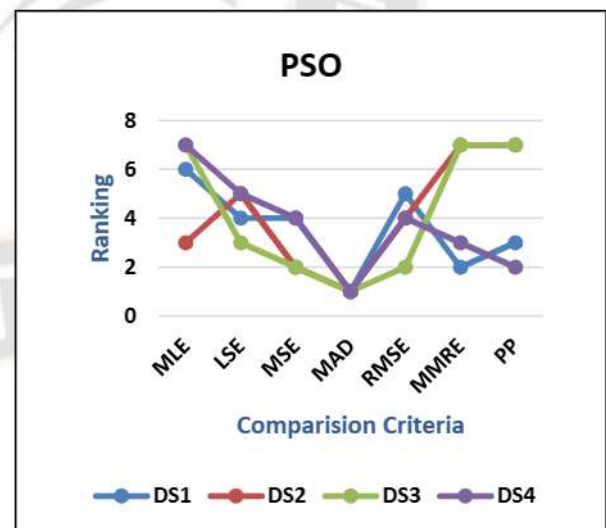


Figure 3. Ranking of PSO values evaluated using comparison criteria

It is clear from the Fig. 1, that between MLE and LSE MLE gives good results as compared to LSE for datasets DS2, DS3, and DS4. Whereas LSE is better than MLE for DS1. Fig. 2 and Fig 3. show the performance of GA and PSO for all datasets and

GA gives better performance for DS1 while SA performs better for DS3.

We also determine the performance of each method for individual datasets to compare which criteria when used as an objective function is giving the best result. Also calculated is the highest value of the estimation function for each evaluation

and method and compared to determine the method giving accurate predictions. The results of the comparisons are shown in the charts of Fig. 4. Following observations were made regarding estimation function values evaluated using various methods for all datasets.

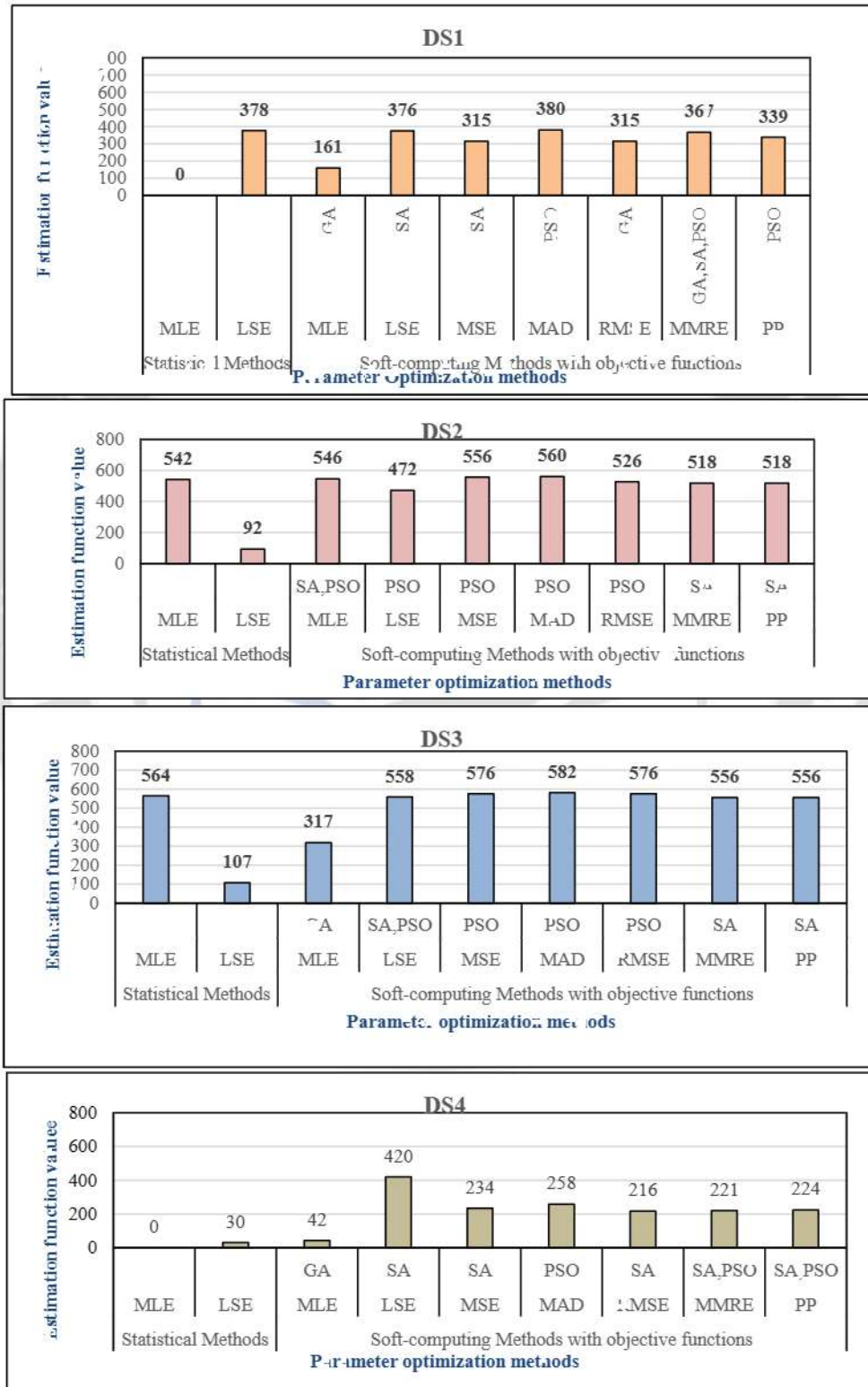


Figure 4. Max values of estimation function for datasets DS1 to DS4

DS1: Figure demonstrates that the PSO using MAD as an objective function gives the highest value of the estimation function giving more accurate prediction as compared to other discussed methods. The second-best prediction is given by LSE using the “Levenberg-Marquardt” algorithm for optimization.

DS2: PSO is the best evaluator for DS2 having both the first and second-best estimation function score using MAD and MSE respectively.

DS3: Again, PSO is a better evaluator than other methods for DS3 Giving both the first and second highest estimation function value for MAD and (MSE, RMSE) respectively.

DS4: SA gives way better performance than any other evaluator for DS4 which is not of fault data of a big data system. PSO with MAD has the second-highest estimation function score.

When we consider all method's highest estimation scores as shown in Fig. 5, we find that PSO using MAD as an objective function gives the best performance for all big datasets. So, we can safely say according to the depicted results that PSO is the best method to be used for the big data hybrid reliability model. Moreover, if we consider the individual performance of each method based on their estimation accuracy, we can infer that the estimation capability of MLE is better for DS3 as compared to other datasets. LSE gives the best prediction for DS1 compared to DS2, DS3, and DS4. Giving first and second highest estimation accuracy is PSO using MAD, MSE, and RMSE for DS3. The third position is bagged by statistical MLE using the “trust-region dogleg algorithm” as an optimizer. SA and PSO both are at position four for DS3 using LSE as an objective function. SA with MMRE and PP for DS3 is at number five and MLE as an objective function for both PSO and SA gets the last position.

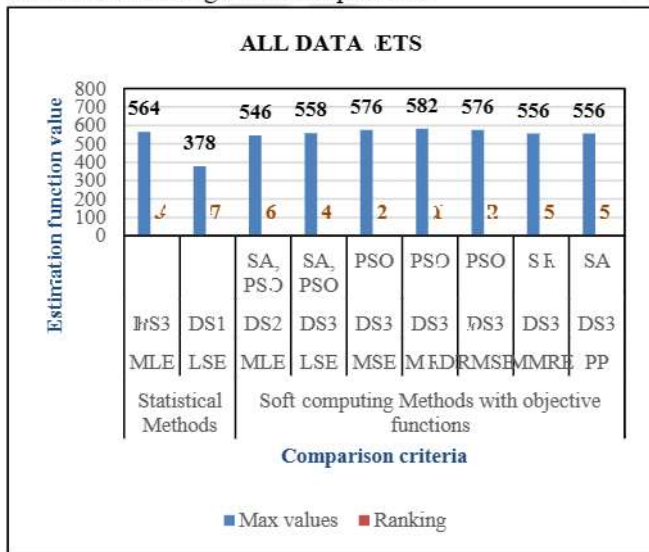


Figure 5. Max values of estimation function in considered methods.

The model is validated by comparing it to two other models Yamada Delayed and Yamada imperfect debugging model number 1. Shown below in Fig. 6 are the intensity

function plot of our developed Hybrid model (HM) with these traditional models for datasets DS1, DS2, DS3, and DS4.

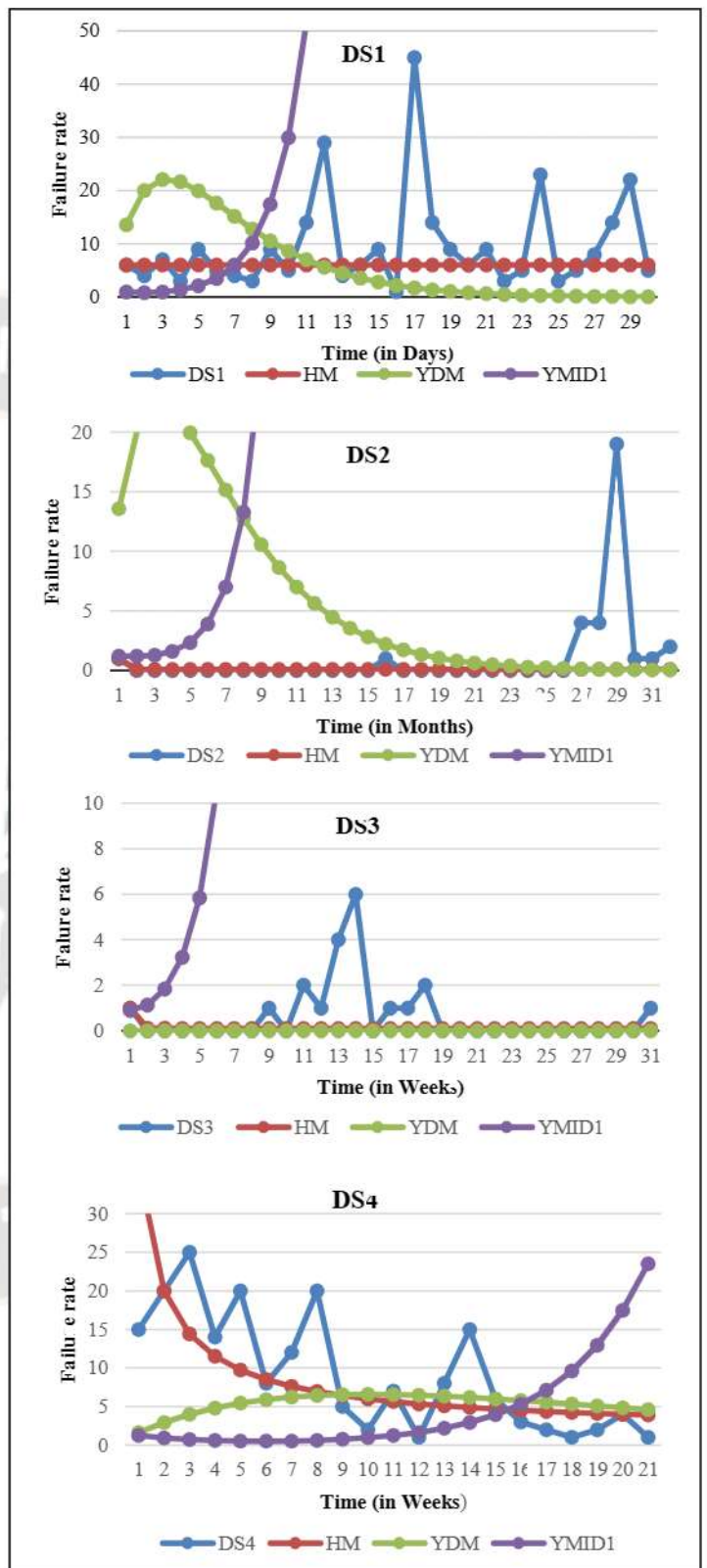


Figure 6. Intensity function comparison of Hybrid Model, YDM and YDIM1.

It is evident from the graphs that the intensity function plot of the hybrid model gives better prediction than the other two

well-known traditional models for all datasets. Thus, we can safely say that the developed hybrid model is a better predictor than traditional models for big data systems.

VI. CONCLUSION:

The study presents a hybrid model that was built to forecast the dependability of software. The parameters of the model were determined with the help of four datasets through the use of a variety of statistical and soft computing approaches. To optimize the parameter values, a total of five different criteria value formulas, together with LSE and MLE, were used as objective functions. There was a total of four datasets utilized in the process of determining the best optimizer with criteria value, and it was discovered that PSO with MAD is the best estimator for DS1, DS2, and DS3, while DS4 received second place for its performance. Based on the results of the experiment, we are able to draw the conclusion that PSO is the technique that is most suited for hybrid models that deal with Big Data analytical systems and uses MAD as its objective function for predictions. Moreover, the comparison of the developed model with traditional models confirms its claim of better predictor than existing models for big fault data.

REFERENCES

- [1]. S. Sharma, N. Kumar, and K. S. Kaswan, "Ranking of Reliability Models based on Accurate Estimation and Weighted Function," *Proc. - 2021 3rd Int. Conf. Adv. Comput. Commun. Control Networkin*, ICAC3N 2021, pp. 1679–1685, 2021, doi: 10.1109/ICAC3N53548.2021.9725534.
- [2]. A. L. Guel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Trans. Reliab.*, vol. R-28, no. 3, pp. 206–211, 1979, doi: 10.1109/TR.1979.5220566.
- [3]. P. Govindasamy and R. Dillibabu, "Development of software reliability models using a hybrid approach and validation of the proposed models using big data," *J Supercomput*, Springer vol. 76, no. 4, 2020.
- [4]. S. Osaki, "S-Shaped Software Reliability Growth Models and Their Applications," *IEEE Trans. Reliab.*, vol. R-33, no. 4, pp. 289–292, 1984, doi: 10.1109/TR.1984.5221826.
- [5]. S. Yamada, M. Ohba, and S. Osaki, "S-Shaped Reliability Growth Modeling for Software Error Detection," *IEEE Trans. Reliab.*, vol. R-32, no. 5, pp. 475–484, 1983, doi: 10.1109/TR.1983.5221735.
- [6]. S. Yamada and S. Osaki, "Software Reliability Growth Modeling: Models and Applications," *IEEE Trans. Softw. Eng.*, vol. SE-11, no. 12, pp. 1431–1437, 1985, doi: 10.1109/TSE.1985.232179.
- [7]. J. D. Musa and K. Okumoto, "A Logarithmic Poisson execution time model for software reliability measurement". In *Proceedings of the 7th international conference on Software engineering*, pp. 230–238, 1984.
- [8]. A. Wood, "Software-reliability growth model: primary-failures generate secondary-faults under imperfect debugging and . *IEEE Transactions on Reliability*, 43, 3, 408 (September 1994)," *Microelectron. Reliab.*, vol. 36, no. September, p. 446, 1996, [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/0026271496819585>.
- [9]. K. Ohishi, H. Okamura, and T. Dohi, "Gompertz software reliability model: Estimation algorithm and empirical validation," *J. Syst. Softw.*, vol. 82, no. 3, pp. 535–543, 2009, doi: 10.1016/j.jss.2008.11.840.
- [10]. S. Yamada, J. Hishitani, and S. Osaki, "Software-Reliability Growth with a Weibull Test-Effort: A Model & Application," *IEEE Trans. Reliab.*, vol. 42, no. 1, pp. 100–106, 1993, doi: 10.1109/24.210278.
- [11]. S. Yamada, K. Tokuno, and S. Osaki, "Imperfect debugging models with fault introduction rate for software reliability assessment," *Int. J. Syst. Sci.*, vol. 23, no. 12, pp. 2241–2252, 1992, doi: 10.1080/00207729208949452.
- [12]. A. Iannino and J. D. Musa, *Software Reliability*, vol. 30, no. C. 1990.
- [13]. C. Y. Huang, M. K. Lyu, and S. Y. Kuo, "A unified scheme of some nonhomogeneous poisson process models for software reliability estimation," *IEEE Trans. Softw. Eng.*, vol. 29, no. 3, pp. 261–269, 2003, doi: 10.1109/TSE.2003.1183936.
- [14]. E. Pham, L. Nordmann, and X. Zhang, "A general imperfect-software-debugging model with s-shaped fault-detection rate," *IEEE Trans. Reliab.*, vol. 48, no. 2, pp. 169–175, 1999, doi: 10.1109/24.784276.
- [15]. S. Yamada, H. Ohtera, and H. Narihisa, "Software Reliability Growth Models with Testing-Effort," *IEEE Trans. Reliab.*, vol. 35, no. 1, pp. 19–23, 1986, doi: 10.1109/TR.1986.4335332.
- [16]. C. Jin and S. W. Jin, "Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization," *Appl. Soft Comput. J.*, vol. 40, pp. 283–291, 2016, doi: 10.1016/j.asoc.2015.11.041.
- [17]. X. Han, L. Tian, M. Yoon, and M. Lee, "A big data model supporting information recommendation in social networks," *Proc. - 2nd Int. Conf. Cloud Green Comput. 2nd Int. Conf. Soc. Comput. Its Appl. CGC/SCA 2012*, pp. 810–813, 2012, doi: 10.1109/CGC.2012.125.
- [18]. Y. Hong, M. Zhang, and W. Q. Meeker, "Big data and reliability applications: The complexity dimension," *J. Qual. Technol.*, vol. 50, no. 2, pp. 135–149, 2018, doi: 10.1080/00224065.2018.1438007.
- [19]. C. Liu *et al.*, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2234–2244, 2014, doi: 10.1109/TPDS.2013.191.
- [20]. Y. Tamura, K. Miyaoka and S. Yamada, "Reliability analysis based on three-dimensional stochastic differential equation for big data on cloud computing," *2014 IEEE International Conference on Industrial Engineering and Engineering Management*, 2014, pp. 863–867, doi: 10.1109/IEEM.2014.7058761.

- [21]. Y. Tamura and S. Yamada, "Reliability Analysis Based on AHP and Software Reliability Models for Big Data on Cloud Computing," vol. 1, no. 1, pp. 43–49, 2014.
- [22]. Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and T. Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 1187–1198, 2014, doi: 10.1145/2588555.2610509.
- [23]. O. Kwon, N. Lee, and B. Shin, "Data quality management, data usage experience and acquisition intention of big data analytics," *Int. J. Inf. Manage.*, vol. 34, no. 3, pp. 387–394, 2014, doi: 10.1016/j.ijinfomgt.2014.02.002.
- [24]. Y. Tamura and S. Yamada, "Software reliability assessment tool based on fault data clustering and hazard rate model considering cloud computing with big data," *2015 4th Int. Conf. Reliab. Infocom Technol. Optim. Trends Futur. Dir. ICRITO 2015*, vol. d, pp. 1–6, 2015, doi: 10.1109/ICRITO.2015.7359208.
- [25]. Y. Tamura and S. Yamada, "Reliability analysis based on a jump diffusion model with two wiener processes for cloud computing with big data," *Entropy*, vol. 17, no. 7, pp. 4533–4546, 2015, doi: 10.3390/e17074533.
- [26]. Y. Tamura and S. Yamada, "Software reliability analysis considering the fault detection trends for big data on cloud computing," *Lect. Notes Electr. Eng.*, vol. 349, pp. 1021–1030, 2015, doi: 10.1007/978-3-662-47200-2_106.
- [27]. Y. Tamura, Y. Nobukawa, and S. Yamada, "A method of reliability assessment based on neural network and fault data clustering for cloud with big data," *2015 IEEE 2nd Int. Conf. InformationScience Secur. ICISSE 2015*, pp. 4–7, 2016, doi: 10.1109/ICISSE.2015.7370965.
- [28]. J. Pence *et al.*, "Quantifying organizational factors in human reliability analysis using the big data-theoretic algorithm," *Int. Top. Meet. Probabilistic Saf. Assess. Anal. PSA 2015*, vol. 2, pp. 650–659, 2015.
- [29]. L. Cai and Y. Zhu, "The challenges of data quality and data quality assessment in the big data era," *Data Sci. J.*, vol. 14, pp. 1–10, 2015, doi: 10.5334/dsj-2015-002.
- [30]. J. Li, Y. He, and Y. Ma, "Research of network data mining based on reliability source under big data environment," *Neural Comput. Appl.*, vol. 28, pp. 327–335, 2017, doi: 10.1007/s00521-016-2349-x.
- [31]. L. Hu, K. Y. Liu, Y. Diao, X. Meng, and W. Sheng, "Operational reliability evaluation method based on big data technology," *Proc. - 2016 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2016*, pp. 341–344, 2017, doi: 10.1109/CyberC.2016.71.
- [32]. M. Spichkova, H. W. Schmidt, I. I. Yusuf, I. E. Thomas, S. Androurakis, and G. R. Meyer, "Towards modelling and implementation of reliability and usability features for research-oriented cloud computing platforms," *Commun. Comput. Inf. Sci.*, vol. 703, pp. 158–178, 2016, doi: 10.1007/978-3-319-56390-9_8.
- [33]. Y. Tamura, T. Takeuchi, and S. Yamada, "Software Reliability and Cost Analysis Considering Service User for Cloud with Big Data," *Int. J. Reliab. Qual. Saf. Eng.*, vol. 24, no. 2, pp. 1–14, 2017, doi: 10.1142/S0218539317500097.
- [34]. J. Yan, Y. Meng, L. Lu, and L. Li, "Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance," *IEEE Access*, vol. 5, no. c, pp. 23484–23491, 2017, doi: 10.1109/ACCESS.2017.2765544.
- [35]. J. Wang, H. Wu, and R. Wang, "A new reliability model in replication-based big data storage systems," *J. Parallel Distrib. Comput.*, vol. 108, pp. 14–27, 2017, doi: 10.1016/j.jpdc.2017.02.001.
- [36]. R. Nachiappan, B. Javadi, R. N. Calheiros, and K. M. Matawie, "Cloud storage reliability for Big Data applications: A state of the art survey," *J. Netw. Comput. Appl.*, vol. 97, pp. 35–47, 2017, doi: 10.1016/j.jnca.2017.08.011.
- [37]. Z. Xiang, Q. Du, Y. Ma, and W. Fan, "Assessing reliability of social media data: lessons from mining TripAdvisor hotel reviews," *Inf. Technol. Tour.*, vol. 18, no. 1–4, pp. 43–59, 2018, doi: 10.1007/s40558-017-0098-z.
- [38]. Y. Tamura and S. Yamada, "Fault Identification and Reliability Assessment Tool Based on Deep Learning for Fault Big Data," *Softw. Netw.*, vol. 2017, no. 1, pp. 161–167, 2017, doi: 10.13052/jsn2445-9739.2017.008.
- [39]. Y. Hong, M. Zhang, and W. Q. Meeker, "Big data and reliability applications: The complexity dimension," *J. Qual. Technol.*, vol. 50, no. 2, pp. 135–149, 2018, doi: 10.1080/00224065.2018.1438007.
- [40]. R. Cao and J. Gao, "Research on reliability evaluation of big data system," *2018 3rd IEEE Int. Conf. Cloud Comput. Big Data Anal. ICCCBDA 2018*, pp. 261–265, 2018, doi: 10.1109/ICCCBDA.2018.8386523.
- [41]. S. Yaremchuk and V. Kharchenko, "Big data and similarity-based software reliability assessment: The technique and applied tools," *Proc. 2018 IEEE 9th Int. Conf. Dependable Syst. Serv. Technol. DESSERT 2018*, no. May, pp. 485–490, 2018, doi: 10.1109/DESSERT.2018.8409182.
- [42]. J. Wang, C. Zhang, and J. Yang, "Software reliability model of open source software based on the decreasing trend of fault introduction," *PLoS One*, vol. 17, no. 5 May, pp. 1–18, 2022, doi: 10.1371/journal.pone.0267171.
- [43]. T. Yanagisawa, Y. Tamura, A. Anand, and S. Yamada, "A Software Reliability Model for OSS Including Various Fault Data Based on Proportional Hazard-Rate Model," *Am. J. Oper. Res.*, vol. 12, no. 01, pp. 1–10, 2022, doi: 10.4236/ajor.2022.121001.
- [44]. T. Minohara and Y. Tohma, "Parameter estimation of hypergeometric distribution software reliability growth model by genetic algorithms," *Proc. Int. Symp. Softw. Reliab. Eng. ISSRE*, no. i, pp. 324–329, 1995, doi: 10.1109/issre.1995.497673.
- [45]. A. Choudhary, A. S. Baghel and O. P. Sangwan, "An efficient parameter estimation of software reliability growth models using gravitational search algorithm", *International Journal of System Assurance Engineering and Management*, 8(1), 79-88,2017.
- [46]. S. Sharma, N. Kumar, and K. S. Kaswan, "Big data reliability: A critical review," *J. Intell. Fuzzy Syst.*, vol. 40, no. 3, pp. 5501–5516, 2021, doi: 10.3233/JIFS-202503.

- [47]. R. Miller, Exponential order statistic models of software reliability growth, *IEEE Transactions on Software Engineering* SE-12(1) (1986), 12–24.
- [48]. M. Xie, "Software Reliability Models for Practical Applications," pp. 211–214, 1995, doi: 10.1007/978-0-387-34848-3_32.
- [49]. H. Pham, "System software reliability", Springer Science & Business Media, 2007.

