J. Integr. Sci. Technol. 2025, 13(2), 1026

Article

Journal of Integrated
SCIENCE& TECHNOLOGY

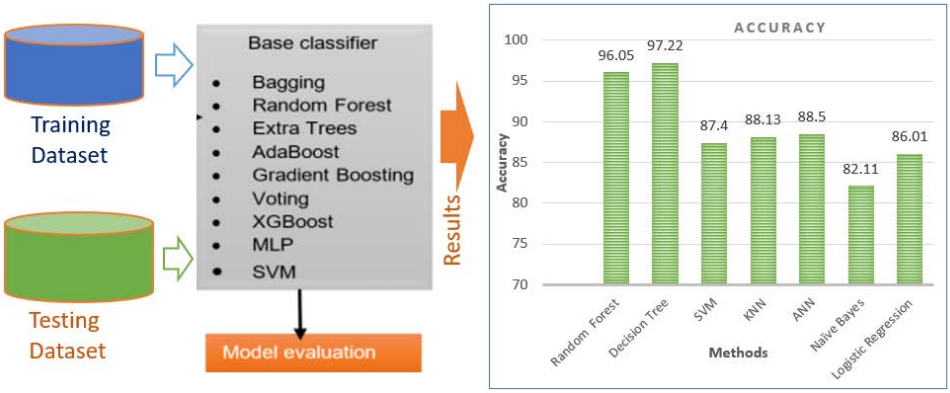
Prediction and classification of software reliability using ensemble learning

Getachew Mekuria Habtemariam[1], Sudhir Kumar Mohapatra[2]*, Hussien Worku Seid[1], Srinivas Prasad[3], Tarini Prasad Panigrahy[4], Prasanta Kumar Bal[4]

[1]Addis Ababa Science and Technology University, Addis Ababa, Ethiopia. [2]Sri Sri University, Cuttack, Odisha, India. [3]GITAM University, Vishakhapatnam, Andhra Pradesh, India. [4]GITA Autonomous College, Bhubaneswar, India.

Received on: 11-Sep-2023; Accepted and Published on: 02-Sep-2024

ABSTRACT



Software reliability plays a pivotal role in determining the overall system reliability and is an inescapable factor when assessing the integrity of any software product. When it comes to creating mission-critical software like software for space exploration, the health sector, scientific calculation, the aerospace industry, etc., where the need for high reliability is paramount, we encounter numerous challenges that need to be effectively addressed. Accurate prediction of software reliability ensures software quality, which ultimately builds the confidence of the customer in the software they are using. Machine learning, particularly the ensemble method, is very important to solve these prediction problems. This research develops an ensemble learning technique for software reliability prediction. Ensemble methods, which are a combination of more individual ML models are used in this research. Bagging, Boosting, and stacking techniques are applied for classification and prediction. Prediction is used to predict the failure time of the software based on the Mean Time Between Failures (MTBF). Musa, J.D's benchmark dataset on Software reliability is used for prediction. The classification is used for classifying the software for the presence of defects or not. NASA dataset is used for classification. The designed model achieves 94% prediction and 97% classification accuracy.

Keywords: Software reliability prediction, Machine learning, ensemble learning, Education Software

INTRODUCTION

Software plays a crucial role in the education sector by enhancing teaching and learning processes, improving administrative tasks, and facilitating communication and collaboration among students, teachers, and administrators. The areas of education where software is used are Learning Management Systems (LMS), Educational Apps, Virtual Learning Environments, Multimedia Tools, Simulations and Virtual Laboratories, Assessment and Grading, Administrative Systems, Collaboration and Communication, Personalized Learning, Data Analysis and Reporting, etc. Overall, software applications and tools have revolutionized the education sector, providing opportunities for flexible learning, individualized instruction, and enhanced collaboration, while also simplifying administrative tasks and providing valuable data for analysis and improvement. The increasing demand forqualitative, error-free, reliable software requires a quality check by the developing companies. Quality of the software can be possible by using software reliability prediction at the time of development of the software.

The reliability of software is stated as "The ability of the software to perform its required function under stated conditions for a stated period of time". Through fast improvement as well as expanding the intricacy of a product, the unwavering quality of the product is difficult to accomplish. Among the most vital aspects as well as characteristics of software soundness is reliability. Software reliability, as defined by ANSI,[1] is "The probability of failure-free operation of a computer program for a specified period in a specified environment."[2] The model has been used to predict and

*Corresponding Author: Sudhir Kumar Mohapatra, Sri Sri University, Cuttack, India
Tel: +91-9556878743; Email: sudhir.mohapatra@srisriuniversity.edu.in

Cite as: *J. Integr. Sci. Technol., 2025, 13(2), 1026.*
DOI: 10.62110/sciencein.jist.2025.v13.1026

estimate the number of software errors in this work.[2,3] Additionally, classification was carried out in this work to assign the error to a desired output class. The principal objective of software reliability modeling is to determine an expected interval time between successive failures or the likelihood of a system malfunction within a stated period frame.[4,5]

Compared to statistical methods, machine learning (ML) methods have proven to be more accurate at predicting outcomes and can be used to predict and classify software failures with greater precision. Computers are able to evolve, predict, and classify system behaviour based on failure data from the past and the present thanks to an approach known as machine learning (ML). This approach is focused on learning automatically. As a result, it is quite natural to be able to quantitatively determine which techniqueinclines to be successful for a particular malfunction dataset as well to what degree.[6-9]

Ensemble learning methods have gained significant importance in solving prediction problems in various domains, including software reliability.[10] In software dependability reliability, the aim is to evaluate the quality and reliability of software products. This assessment is crucial in building customer confidence in the software they are procuring. By accurately predicting software reliability, organizations can ensure high-quality software that meets customer expectations. Ensemble learning leverages the diversity of different models to achieve better predictive performance.[11] It combines the individual predictions of multiple models through voting, averaging, or weighting mechanisms. This approach helps mitigate the limitations of individual models and enhances the overall reliability of the predictions. The ensemble learning-based model developed in this article holds promise for improving software reliability prediction. By utilizing diverse models and combining their predictions, the model can provide accurate and robust estimations of software failures. This, in turn, contributes to enhancing software quality and instilling customer confidence.

LITERATURE REVIEW

This section describes how machine learning approaches are used by researchers for better software reliability. A study has been done by Sabnis et al.[12] to compare the different technologies, and they used machine learning methods to estimate the defect level of the software. They have taken various methods like SVM, ANN, NB and RF where ANN shows good results as compared to others. ANN classifiers have the best accuracy about 65.5% among all other machine learning technologies. Another study has been done by Jindal et al.[13] in which a heuristics test of different Machine learning and Deep Learning methods on univariate software failure time stamp data was used to find the best approach for software reliability. The main objective of this study was to predict Software reliability using various machine learning methods. After choosing the algorithm, that algorithm is trained to determine the failure. A total of 101 data samples have been taken for the testing purpose and from that 2 attributes have been shown. Here four model have been trained for the purpose of predicting the software reliability and also reported their individual performances. ANN has been taken as the baseline model and found that the LSTM model performs well. Banga et al.[14] introduced an approach which is used to find the most relevant parameters that affects the software reliability. In this research, a hybrid approach is used to predict the fault of software with the help of machine learning.[15] The study proposed a method to detect the quality of software with the help of matrices. The information provided by the matrices is important to detect the failure earlier which is very important in the field of software. In the experiment part, they have taken eight different types of classifiers using metrics which have been collected from freely available projects PROMISE data repository.

Yaghoobi et al. (2021)[16] gave two multiple-criteria decision-making methods for contrasting and selecting the most suitable Software Reliability Growth Model (SRGM) for a specific dataset. The methods determine an evaluation for every SRGM based on the weight estimates and compute a weight for every analytical criterion in terms of the level of diversity. The simplicity, criterion weighting, and incorporation of numerous descriptive and predictive properties of a framework in the framework selection procedure are advantages of the techniques.

Sudharson et al. (2019)[17] stated that in order to get dependability in software results by assessing faults during examination, software reliability is a crucial quantitative attribute. To find product faults, time-dependent software reliability models are used, but they are useless in environments that are constantly changing. The researcher uses machine learning techniques for software reliability prediction.[18]

Li et. al.[19] research is on the reliability of the object-oriented program. For the first time, they come up with special features for OOP. Soft computing and machine learning models are proposed and deduced by the researcher for software testing and quality assurance.[20-23]

Luo et al.(2023)[24] proposed a reliability growth model based on non-homogeneous poison distribution. The result confirmed the model is effective in fault fitting and prediction. Chen et. al. (2023)[25] study is on open-source software. The idea is nowadays a lot of open-source software is used and its reliability is vital for practical use. The researcher used a modified diffusion model for it. The model can be used to determine the optimal release time of the software. Liu et. al.(2022)[26] came up with a reliability growth model based on an uncertain differential equation. The author also proposed a new method (MESBRGM) using uncertainty theory. The model comparison with other models shows promising results in performance and accuracy.[27]

Table 1. Literature Review Summary

Author	Title	Performance	Dataset	Findings
Sabnis et al.[12]	A Study on Machine Learning Techniques Based Software Reliability Assessment.	65.5% accuracy	NA	ANN classifiers have the best accuracy among all machine-learning technology
Jindal et al.[13]	Comparative Analysis of Software Reliability Prediction Using Machine Learning and Deep Learning	Mean Absolute Error 1.5639 which is very less	Software Failures Dataset[*]	LSTM model performs well.

Banga et al.[14]	Implementation of machine learning techniques in software reliability: A framework.	78% accuracy	NA	A hybrid new approach to fault prediction based on a machine learning algorithm.
Reddivari et al.[15]	Software quality prediction: an investigation based on machine learning.	AUC of 0.75	UIMS and QUES [**]	decision tree-based prediction techniques perform well.
Yaghoobi et al. (2021) [16]	Selection of optimal software reliability growth model using a diversity index	85% accuracy	NA	Statistical models are used
Sudharson et al. (2019) [17]	A novel machine learning approach for software reliability growth modelling with pareto distribution function	85% accuracy	NA	Soft computing methods are used

The major limitation of all the existing work is the accuracy of the model. The highest accuracy achieved by any model is 86%, which is very less for the industrial use of the model. The existing models either use prediction or classification. The main objectives of our model are

1. Both prediction and classification-based ensemble models.
2. Achieving high prediction and classification accuracy.

DESIGNED MODEL

Developing reliable software for critical business applications is a significant challenge in the software industry today. Several factors that contribute to this challenge are the complexity of the software, security, scalability, reliability and fault tolerance, testing and quality assurance. Addressing these challenges requires industry standards and methodologies (such as agile or DevOps), prioritising software quality, and leveraging automated testing and deployment pipelines to ensure reliable software development for critical business applications. Ensemble methods have been applied for reliability, estimating and classifying the amount of defects present in software. A principal objective of software reliability modelling is determining the likelihood of a software malfunction data specified period interval, if no anticipated time duration among consecutive breakdowns. For this work, ML methods utilized for predicting software reliability prediction and classification which are SVM, KNN, Random Forest, Decision Tree, Linear Regression, Logistic Regression, Bayesian Ridge Regression, Lasso Regression, ElasticNet Regression, ANN, Naïve Bayes algorithm, SVR, bagging, boosting & stacking. The mentioned machine learning methodologies are used together on two different datasets in this research.

Prediction:

In order to predict software reliability, the dataset of a successive failure of the software is used and different above-mentioned ML techniques are used for prediction of the failure time of the software reliability dataset. The value is predicted based on the Mean Time Between Failure (MTBF) using a single feature dataset representing the meantime between failures in chronological order. To perform prediction, the cumulative MTBF (equation 1) is calculated for bagging and hence, the subsequent errors arepredicted. In prediction, a set of hypotheses is combined to give better accuracy and improved results.

MTBF= (Total operation time-total breakdown time)/(Number of break downs)……..1

Classification:

In a classification problem, based on certain parameters of a module in software like cyclomatic complexity, significate complexity, blueprint complexity as well as number of lines etc., the module of a software needs to be classified as whether it would have one or more reported defects or not. Using Ensemble learning for the classification problem is based on the different types of classifiers mentioned above.

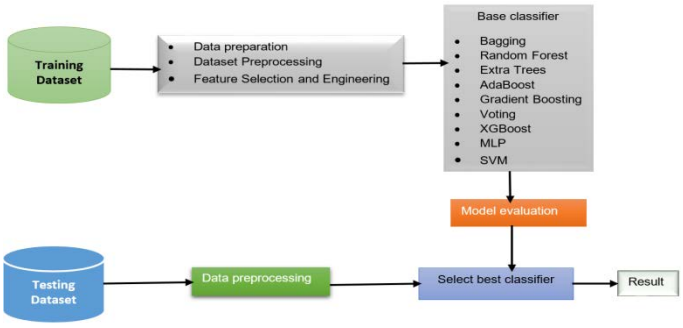


Figure 1. The representation of the designed Model

The proposed model is presented in Figure 1. The individual models are trained using training data. The ensemblermodel then combines the individual model prediction. In the proposed model Bagging, Boosting, and Stacking ensemble techniques are used. Once the model is trained then the model is tested using the test data. The data are split using a 10-fold cross-validation method. The hyperparameters of the individual models are given in Table 2.

Table 2. Hyperparameter of individual models

Base Model	Hyperparameter	Values
Ridge	Alpha	10^range(−5, 0)a
LASSO	Alpha	10^range(−5, 0)
Elastic Net	Alpha	10^range(−5, 0)
	l1_ratio	10^range(−5, 0)
Bayesian Ridge	alpha_1	10^range(−5, 0)
	alpha_2	10^range(−5, 0)
SVM	C	linspace(0.01, 5, 20)c
	Gamma	range(0.01, 0.5, 0.05)
	Kernel	{linear, poly, rbf}
KNN	n_neighbors	range(2, 11)
Regression tree	max_depth	range(4, 23)
Bagging	n_estimators	{100, 200, 500}
	max_samples	{0.7, 0.8, 0.9, 1.0}
Random Forest	n_estimators	{100, 200, 500}
	max_depth	range(4, 10)
Neural network	Alpha	linspace(0.0001, 0.5, 20)
	learning_rate_init	linspace(0.0001, 0.5, 20)
	Activation	{identity, logistic, tanh, relu}

RESULT AND DISCUSSION

The proposed model consists of 3 ensemble models. The three different models are Bagging, Boosting and Stacking. This article on Software reliability prediction by applying an ensembling learning approach and three different ensembling machine learning approaches which are used to forecast and classify the dataset. In which bagging, boosting & stacking is implemented using Python programming language given software reliability prediction & classification dataset. To implement the ensemble model with the selected software tools, a machine with a processor of Intel(R) Core(TM) i7-5200U CPU @ 2.20GHz 2.20 GHz and 8 GB RAM memory capacity is used. It is also tested in Google Colab for comparing the computational speed with 1gbps internet speed. In software reliability prediction dataset given a single row of failure time of the software indexed in a file, where a sample with replacement technique is used to build another dataset and train the model on that dataset and after that the prediction is done using the original dataset and similarly, different rounds are used for the iteration purpose to enhance the accurateness of the ensembling model. While predicting an output, different machine learning regression models are applied to predict an output and then find the error rate in the model. All the output value is stored in an xlsx file and tableau software is used to visualize the data or the output that we got from the Ensemble method for software reliability prediction. The experiment process is presented in Figure 2.

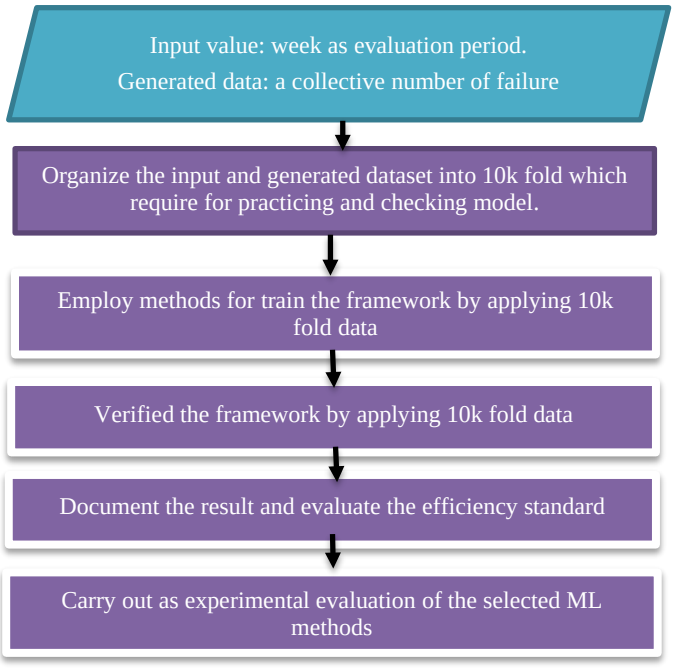


Figure 2. The process adopted. forSoftware reliability prediction

Dataset:

In this section, it is described the two datasets which are applied in our proposed work. For the prediction of the time of failure, the Musa dataset was used. It is a benchmarked dataset containing 101 observations of the pair (T, Y*t*) pertaining to software failure. T represents the Tth modification corresponding to the time of failure Y*t*. The data set is publicly available in the Mohanty et. al. research article [28].

Table 3: The Musa dataset

T	Y_t	T	Y_t	T	Y_t	T	Y_t
0	5.7683	26	8.5941	52	10.0998	78	14.7824
1	9.5743	27	11.0399	53	12.6078	79	14.8969
2	9.105	28	10.1196	54	7.1546	80	12.1399
3	7.9655	29	10.1786	55	10.0033	81	9.7981
4	8.6482	30	5.8944	56	9.8601	82	12.0907
5	9.9887	31	9.546	57	7.8675	83	13.0977
6	10.1962	32	9.6197	58	10.5757	84	13.368
7	11.6399	33	10.3852	59	10.2994	85	12.7206
8	11.6275	34	10.6301	60	10.6604	86	14.192
9	6.4912	35	8.3333	61	12.4972	87	11.3704
10	7.901	36	11.315	62	11.3745	88	12.2021
11	10.2679	37	9.4871	63	11.9158	89	12.2793
12	7.6839	38	8.1391	64	9.575	90	11.3667
13	8.8905	39	8.6713	65	10.4504	91	11.3923
14	9.2933	40	6.4615	66	10.5866	92	14.4113
15	8.3499	41	6.4615	67	12.7201	93	8.3333
16	9.0431	42	7.6955	68	12.5982	94	8.0709
17	9.6027	43	4.7005	69	12.0859	95	12.2021
18	9.3736	44	10.0024	70	12.2766	96	12.7831
19	8.5869	45	11.0129	71	11.9602	97	13.1585
20	8.7877	46	10.8621	72	12.0246	98	12.753
21	8.7794	47	9.4372	73	9.2873	99	10.3533
22	8.0469	48	6.6644	74	12.495	100	12.4897
23	10.8459	49	9.2294	75	14.5569		
24	8.7416	50	8.6971	76	13.3279		
25	7.5443	51	10.3534	77	8.9446		

For classification, NASA (http://mdp.ivv.nasa.gov) dataset is used. The data set consists of 22 static metrics of the software. The details of the feature are presented in Table 4.

Table 4: The NASA dataset details

R.no	Metrics	Explanation
1	*loc*	McCabe's code of line count
2	*v(g)*	McCabe "cyclomatic complexity"
3	*ev(g)*	McCabe "essential complexity"
4	*iv(g)*	McCabe "design complexity"
5	*n*	Halstead over-all operands+ operators
6	*v*	Halstead "volume"
7	*l*	Halstead "program length"
8	*d*	Halstead "difficulty"
9	*i*	Halstead "intelligence"
10	*e*	Halstead "effort"
11	*b*	Halstead “value”
12	*t*	Halstead's time estimator
13	*lOCode*	Halstead's line count
14	*lOComment*:	Halstead's count of lines of comments
15	*lOBlank*:	Halstead's count of blank lines
16	*lOCodeAndComment*	IO lines
17	*uniq_Op*:	unique operators
18	*uniq_Opnd*:	unique operands
19	*total_Op*:	total operators
20	*total_Opnd*:	total operands
21	*branchCount*:	of the flow graph
22	*prediction* :	{false, true} module has one or more reported defects or not

Bagging

The prediction models or the regression methods those are used in this research are Linear Regression, Decision Tree regression, Ridge regression, Lasso regression, ElasticNet regression, Random Forest regression, Support vector regression and others. After predicting the dataset and finding the error; the error value is very less in each model. We can also say that the error is tends to zero.

For each of the regression method the error rate is very less and for some regression like the logistic regression and support vector regression the error rate is bit high. The comparison between actualfailure interval values versus various regression model and the bagging model is shown in Figures 3, 4 & 5.

Figure 3. Regression model comparison between Ridge regression, lasso regression, Decision Tree regressor and the Actual time interval.

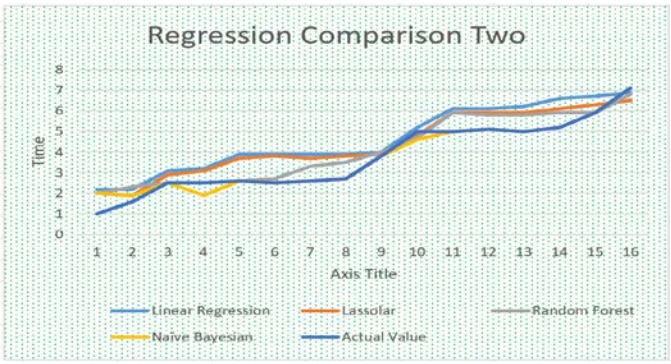


Figure 4. Regression model comparison between Random Forest regression, linear regression, Bayesian Ridge regressor, LassorLars Regressor and the Actual time interval.

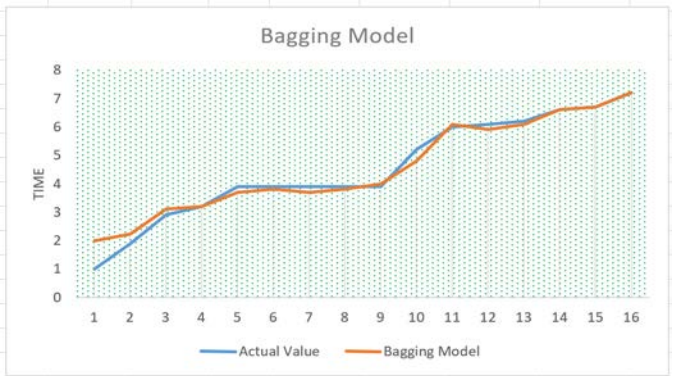


Figure 5. Actual failure time interval and Bagging model failure time interval comparison.

Similarly, in the bagging approach for the classification 7 different models or machine learning classifiers are used to classify the sample with replaced dataset and then the original dataset is used to get the output class and then finding the final output voting approach is used to find the final output value. While displaying the output in the bagging approach accuracy of the model is printed also the confusion matrix and maximum value, minimum value, mean value and final bagging accuracy is printed which gives a clear idea about the classification of each model. The visualization of the accuracy of each model while where the iterator range is 7 is visualized below in Figure 6.

Similarly, while we print the confusion matrix of each model there are four possible values where 2 out of 4 options are correct and others are incorrect. The accuracy value of each model depends upon the confusion matrix of each model and the accuracy can be calculated by using the confusion matrix. If there is a plot between each value of the confusion matrix in each model, then the line graph is shown as in Figure 7.

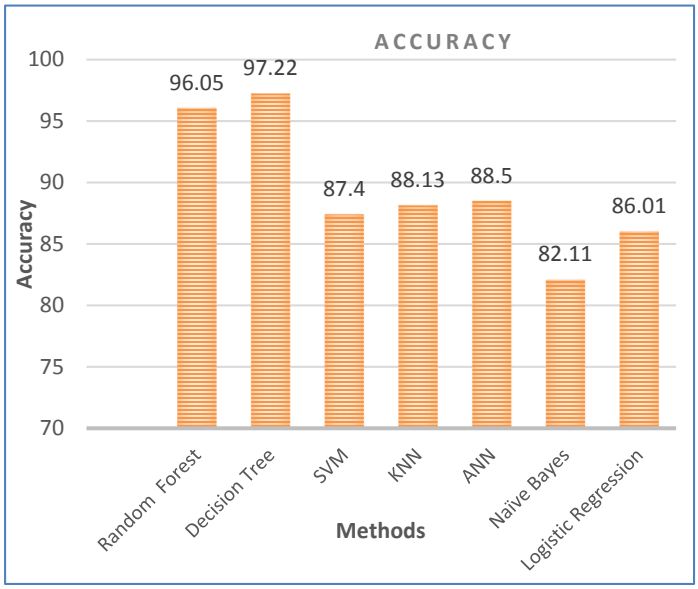


Figure 6. Accuracy of the different machine learning classification model

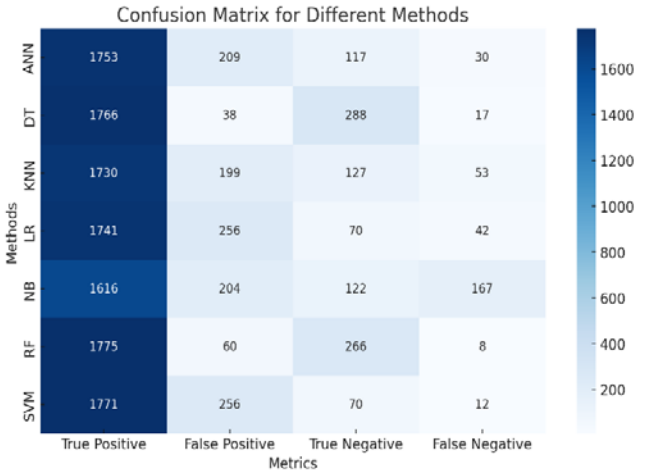


Figure 7. Confusion matrix visualization of different machine learning models

Boosting:

Adaboost algorithm is used to implement boosting over software reliability prediction to enhance the accuracy of a dataset where for each model we are creating multiple instances of the boosting where each instance will act as a neuron; the weight will be initialized in very small amount and then output will be predicted and again the weight will be adjusted and similar thing continues. At last the model gives us the individual accurateness of every model as well as aboosted accuracy of each model for each iteration. And we can also observe the following graph to observe the individual accuracy of each model for each iteration and boosting accuracy of each model for each iteration in the Figures 7, 8, 9 & 10.

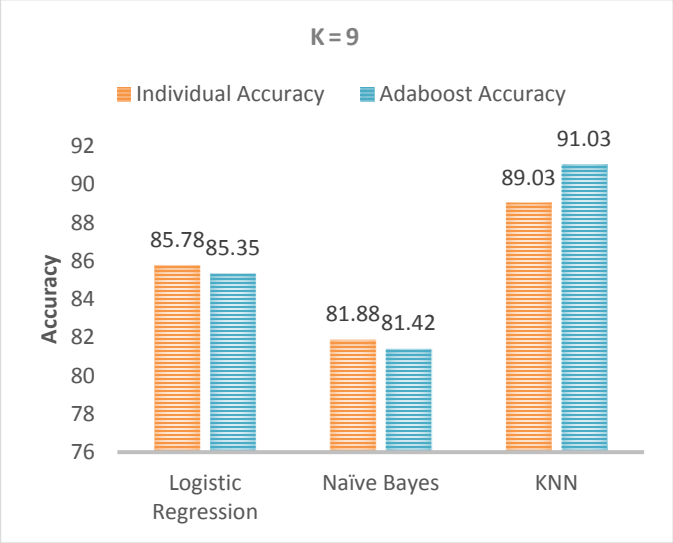


Figure 8. Comparison between boosting accuracy and individual accuracy of 3 models for iterative range k=9

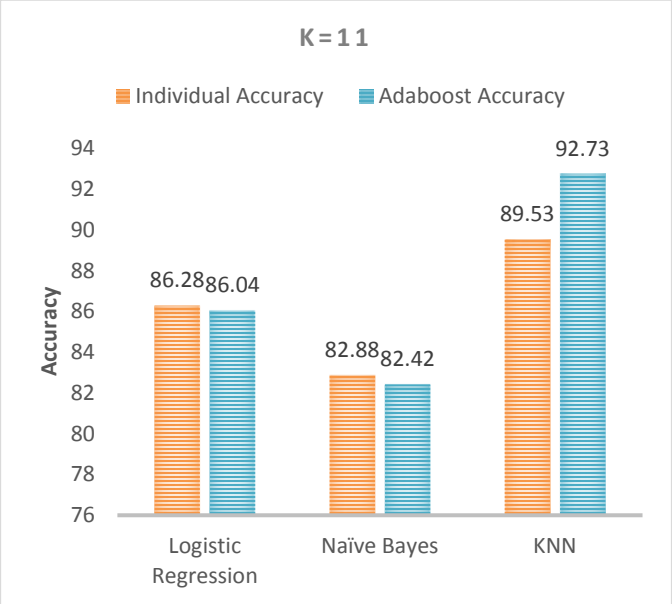


Figure 9. Comparison between boosting accuracy and individual accuracy of 3 models for iterative range k=11

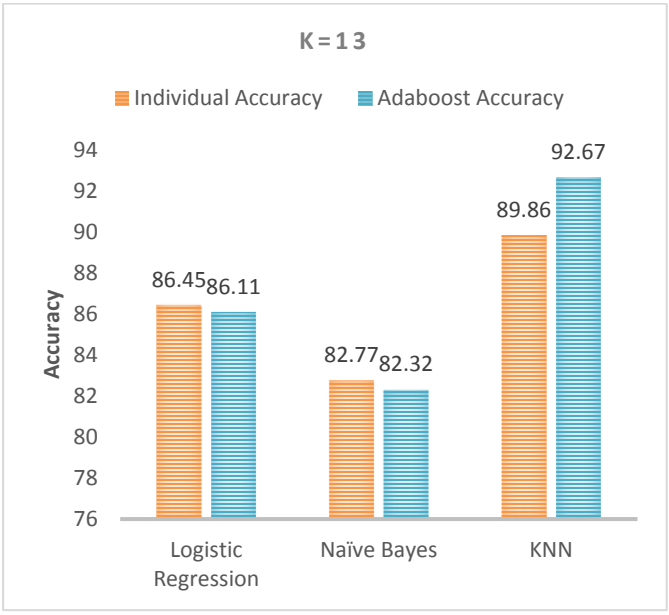


Figure 9. Comparison between boosting accuracy and individual accuracy of 3 models for iterative range k=13

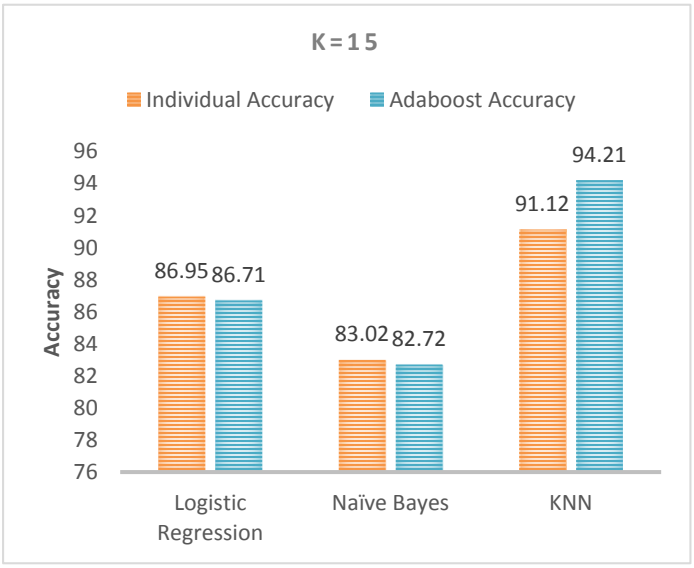


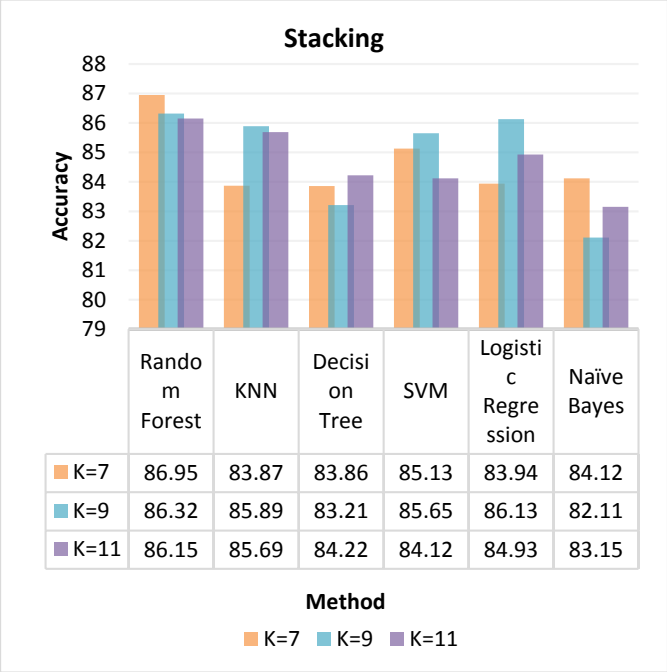
Figure 10. Comparison between boosting accuracy and individual accuracy of 3 models for iterative range k=15

As we can see from the above graph that for different value of K the result is different and for some k the accuracy is higher in increasing k.

Stacking:

In stacking, to learn a machine learning techniques over a working out dataset, afterwards the current dataset is produced through these models. The current dataset is utilized as an input for the combiner machine learning techniques.

In stacking, sub-models produce different predictions. All these sub model's prediction is combined to generate a new dataset and then that is used for the combiner model. For different iteration of stacking different models are used in stacking and the accuracy value of the stacking is represented in the Figure 11.



	Random Forest	KNN	Decision Tree	SVM	Logistic Regression	Naïve Bayes
K=7	86.95	83.87	83.86	85.13	83.94	84.12
K=9	86.32	85.89	83.21	85.65	86.13	82.11
K=11	86.15	85.69	84.22	84.12	84.93	83.15

Figure 11. Comparison between stacking accuracy of different machine learning model for different iterative range

Thus, from the above graph we conclude that for different value of k in different model different model gives different accuracy. And from the above graph we can clearly see the variance of the accuracy for different models for different k.

Conclusion

Various machine learning prediction and classification techniques are applied in ensemble learning methods for software reliability prediction. As we have realized that in bagging approach of the classification the output is classified in different class and then voting method is used to get the final output. In prediction problem we are getting very less error rate for prediction. In boosting model, we have taken weak machine learning model and then by using AdaBoost algorithm the accuracy of the model is improved. In stacking we are combining different base learner algorithm and then predicting the dataset and finally we are using combiner algorithm to predict the output. In bagging method of classification, we can see that for k=7 the decision tree gives a highest performance among all other machine learning model. In boosting method, for different value of k the output of KNN model is improved and the accuracy value is increased by maximum 1%. For stacking for different value of k i.e. 7,9 or 11 KNN model gives the highest accuracy among all.The ununiform result with a varying value of K is a challenge of this model. The researcher planning to do more study on the result and to derive a consistent lag value(k), which uniformly impacting on the classification result. The model is trained and tested using a benchmark dataset i.e. Musa, NASA, but it needs to be tested using the real dataset from the industry. The researcher also planned to apply this model in a software company and observe the prediction and actual output.

Conflict of Interest Statement

Authors declare that there is no conflict of interest for this work as no financial help was received for this work.

References

1. A. Quyoum, M.-U.-D. Dar, S.M.K. Quadri. Improving Software Reliability using Software Engineering Approach- A Review. *Int. J. Comput. Appl.* **2010**, 10 (5), 41–47.
2. S.H. Aljahdali, K.A. Buragga. Employing four ANNs Paradigms for Software Reliability Prediction: an Analytical Study. *ICGST AIML J* **2008**, 8 (Ii), 1687–4846.
3. N. Karunanithi, D. Whitley, Y.K. Malaiya. Prediction of software reliability using connectionist models. *IEEE Trans. Softw. Eng.* **1992**, 18 (7), 563–574.
4. K.K. Sharma, A. Sinha, A. Sharma. An analytical study on testing metrics for software applications. *J. Integr. Sci. Technol.* **2023**, 11 (3), 517.
5. J.H. Lo. Predicting software reliability with support vector machines. In 2nd International Conference on Computer Research and Development, ICCRD 2010; **2010**; pp 765–769.
6. D. Kumari, K. Rajnish. Investigating the effect of object-oriented metrics on fault proneness using empirical analysis. *Int. J. Softw. Eng. its Appl.* **2015**, 9 (2), 171–188.
7. N. Yadav, V. Yadav, D.A.P.J. Abdul. Software reliability prediction and optimization using machine learning algorithms: A review. *J. Integr. Sci. Technol.* **2023**, 11 (1), 457.
8. R. Malhotra, A. Kaur, Y. Singh. Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines; *Int. J. System Assurance Engin. Management*, **2010**, 1, 269-281.
9. R. Agrawal, O. Sharma, N.O. Aljehane, R.F. Mansour. Soft Computing: Goals, importance and various problem-solving techniques. *J. Integr. Sci. Technol.* **2023**, 11 (3), 522.
10. V. Shrivastava, A.K. Chaturvedi. A review on intrusion detection system for distributed network based on Machine Learning. *J. Integr. Sci. Technol.* **2024**, 12 (2), 739.
11. S. Chorey, N. Sahu. Rapid Recover Map Reduce (RR-MR): Boosting failure recovery in Big Data applications. *J. Integr. Sci. Technol.* **2023**, 12 (3), 773.
12. P.S. Sabnis, S. Joshi, J. Naveenkumar. A Study on Machine Learning Techniques based Software Reliability Assessment. In 4th International Conference on Inventive Research in Computing Applications, ICIRCA 2022 - Proceedings; *IEEE*, **2022**; pp 687–692.
13. A. Jindal, A. Gupta, Rahul. Comparative Analysis of Software Reliability Prediction Using Machine Learning and Deep Learning. In Proceedings of the 2nd International Conference on Artificial Intelligence and Smart Energy, ICAIS 2022; *IEEE*, **2022**; pp 389–394.
14. M. Banga, A. Bansal, A. Singh. Implementation of Machine Learning Techniques in Software Reliability: A framework. In 2019 International Conference on Automation, Computational and Technology Management, ICACTM 2019; *IEEE*, **2019**; pp 241–245.
15. S. Reddivari, J. Raman. Software quality prediction: An investigation based on machine learning. In Proceedings - 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science, IRI 2019; *IEEE*, **2019**; pp 115–122.
16. T. Yaghoobi. Selection of optimal software reliability growth model using a diversity index. *Soft Comput.* **2021**, 25 (7), 5339–5353.
17. D. Sudharson, D. Prabha. A novel machine learning approach for software reliability growth modelling with pareto distribution function. *Soft Comput.* **2019**, 23 (18), 8379–8387.
18. J. Lou, Y. Jiang, Q. Shen, et al. Software reliability prediction via relevance vector regression. *Neurocomputing* **2016**, 186, 66–73.
19. W. Li, S. Henry. Object-oriented metrics that predict maintainability. *J. Syst. Softw.* **1993**, 23 (2), 111–122.
20. G.M. Habtemariam, S.K. Mohapatra, H.W. Seid, D.B. Mishra. A Systematic Literature Review of Predicting Software Reliability Using

Machine Learning Techniques. In EAI/Springer Innovations in Communication and Computing; **2022**; pp 77–90.
21. S.K. Mohapatra, A.K. Mishra, S. Prasad. Intelligent Local Search for Test Case Minimization. *J. Inst. Eng. Ser. B* **2020**, 101 (5), 585–595.
22. R. Sharma, A. Saha. Optimization of object-oriented testing using firefly algorithm. *J. Inform. Optim. Sci.*, **2017**, 38(6), 873-893.
23. D. Getachew, S.K. Mohapatra, S. Mohanty. A Heuristic-Based Test Case Prioritization Algorithm Using Static Metrics. In EAI/Springer Innovations in Communication and Computing; Springer International Publishing, Cham, **2022**; pp 45–58.
24. H. Luo, L. Xu, L. He, L. Jiang, T. Long. A Novel Software Reliability Growth Model Based on Generalized Imperfect Debugging NHPP Framework. *IEEE Access* **2023**, 11, 71573–71593.
25. K.J. Chen, C.Y. Huang. Using Modified Diffusion Models for Reliability Estimation of Open Source Software. *IEEE Access* **2023**, 11, 51631–51646.
26. Z. Liu, R. Kang. Imperfect Debugging Software Belief Reliability Growth Model Based on Uncertain Differential Equation. *IEEE Trans. Reliab.* **2022**, 71 (2), 735–746.
27. Z. Liu, S. Wang, B. Liu, R. Kang.. Change point software belief reliability growth model considering epistemic uncertainties. *Chaos, Solitons & Fractals*, **2023**, 176, 114178.
28. R. Mohanty, V. Ravi, M.R. Patra. Application of Machine Learning Techniques to Predict Software Reliability. *Int. J. Appl. Evol. Comput.* **2010**, 1 (3), 70–86.