iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** XII **Month of publication:** December 2023

DOI: https://doi.org/10.22214/ijraset.2023.57626

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue XII Dec 2023- Available at www.ijraset.com

Software Bug Prediction Using Machine Learning Approach

Dr Adarsh T.K[1], Sinchana R[2], Kulsum Pasha.C[3], Uday.R[4]
[1]*Hod Professor,* [2, 3, 4]*Students, ISE Department, T John Institute of Technology*, Visvesvaraya Technological University

Abstract: ***Software defect prophecy work focuses on the number of faults remaining in a software system. A prophecy of the number of remaining scars in an audited artefact can be used for decision timber. An accurate prophecy of the number of scars in a software product during system testing contributes not only to the operation of the system testing process but also to the estimation of the product's demanded conservation. amiss software modules beget software failures, increase development and conservation costs, and drop customer satisfaction. It strives to meliorate software quality and testing effectiveness by constructing predictive models from law attributes to enable a timely identification of fault-prone modules. In this design, we will explore machine knowledge ways for software defect prophecy . This helps the formulators to descry software scars and correct them. Unsupervised ways may be used for defect prophecy in software modules, more so in those cases where defect labels are not available.***

Keyword: Machine Learning algorithms, Naive Bayes (NB) classifier, Decision Tree (DT) classifier, Artificial Neural Networks (ANNs) classifier, Bug Predicting Tools.

I. INTRODUCTION

The existence of software bugs affects dramatically on software reliability, quality and maintenance cost. Achieving bug-free software also is hard work, even the software applied carefully because most time there is hidden bugs. In addition to, developing software bug prediction model which could predict the faulty modules in the early phase is a real challenge in software engineering. Software bug prediction is an essential activity in software development. This is because predicting the buggy modules prior to software deployment achieves the user satisfaction, improves the overall software performance. Moreover, predicting the software bug early improves software adaptation to different environments and increases the resource utilization. Various techniques have been proposed to tackle Software Bug Prediction (SBP) problem. The most known techniques are Machine Learning (ML) techniques. The ML techniques are used extensively in SBP to predict the buggy modules based on historical fault data, essential metrics and different software computing techniques. In this paper, three supervised ML learning classifiers are used to evaluate the ML capabilities in SBP. The study discussed Naive Bayes (NB) classifier, Decision Tree (DT) classifier and Artificial Neural Networks (ANNs) classifier. The discussed ML classifiers are applied to three different datasets obtained from [1] and [2] works.

II. LITERATURE REVIEW

[1] This paper introduces a software bug prediction model leveraging machine learning (ML) algorithms, specifically Naive Bayes (NB), Decision Tree (DT), and Artificial Neural Networks (ANNs). By analyzing historical data, these supervised ML algorithms are employed to forecast future software faults. The evaluation of the model demonstrated its effectiveness, achieving a high accuracy rate. Additionally, a comparison with other existing approaches was conducted, revealing that the ML-based model outperformed alternative methods. In essence, the results suggest that employing ML algorithms is a promising strategy for software bug prediction, exhibiting superior performance compared to other approaches. [2] This study proposes Con-build for model building, Con-update for model updating, and ConEA for model evaluation in order to address software bug prediction in continuous software development. Con-update directs model reuse or update using changing distributional properties, while Con-build redefines training data selection based on bug prediction data distribution. ConEA rethinks effort-aware evaluation by taking file evolution into account for buggy probabilities. Trials conducted on 120 versions across six open-source systems demonstrate useful advantages. This research calls for a more thorough re-examination of software analysis models in continuous development, encompassing effort estimation and issue triage, in addition to its primary focus on bug prediction.

[3] In their research, they conducted a comprehensive review of existing literature on software bug prediction and machine learning to gain insights into constructing prediction models. Examining 31 key studies, they identified six predominant machine learning techniques.

Not only did they analyse the techniques used, but they also assessed the commonly employed datasets, metrics, and performance measures during model development. Their focus narrowed down to two frequently used public datasets, with object-oriented metrics being the preferred choice. Performance evaluation involved both graphical and numerical measures.

While their findings suggest that machine learning techniques can effectively predict bugs, their practical applications remain limited. They identified challenges in constructing prediction models, highlighting the need for more in-depth studies to yield robust results. Their study concludes with recommendations for future research based on the insights derived from their analysis.

[4] This work covers five research issues and presents an experimental model for software fault prediction using the NASA MDP dataset. Interestingly, 13 classifiers were examined and the results showed that J48, IBk, Bayesian, and random tree were the most effective, while regression classification produced the least favourable outcomes. The impact of data pre-processing approaches is examined in the study, and it is discovered that Propositionalization works best for the defect prediction dataset. An investigation of the SMOTE algorithm is carried out to solve class imbalance, assessing the impact of two parameters and recent cases on prediction accuracy. Furthermore, the research suggests using the Adaptive Synthetic Minority Over-sampling (ASMO) method to improve SMOTE. Experimental findings show that the ASMO algorithm produces better True Positive rates and Area Under the Curve (AUC) when compared to standard SMOTE.

[5] This study employs three supervised machine learning algorithms Logistic Regression, Naïve Bayes, and Decision Tree—to predict software bugs based on historical data. Validation is achieved through random forest ensemble classifiers and K-Fold cross-validation, ensuring effectiveness across various scenarios. The proposed work addresses the need for improved bug prediction in software development, emphasizing the limitations of individual classifiers and the significance of variable selection, feature engineering, and reductions. The preference for the random forest algorithm stems from its ensemble nature, leveraging multiple decision trees for enhanced accuracy. Future directions include exploring artificial neural networks for increased prediction accuracy and developing forecasting models for bug occurrences. While initial thoughts included Artificial Neural Network (ANN), the decision to settle with machine learning algorithms was influenced by achieving 100% accuracy with existing datasets, with potential consideration for ANN with expanded datasets.

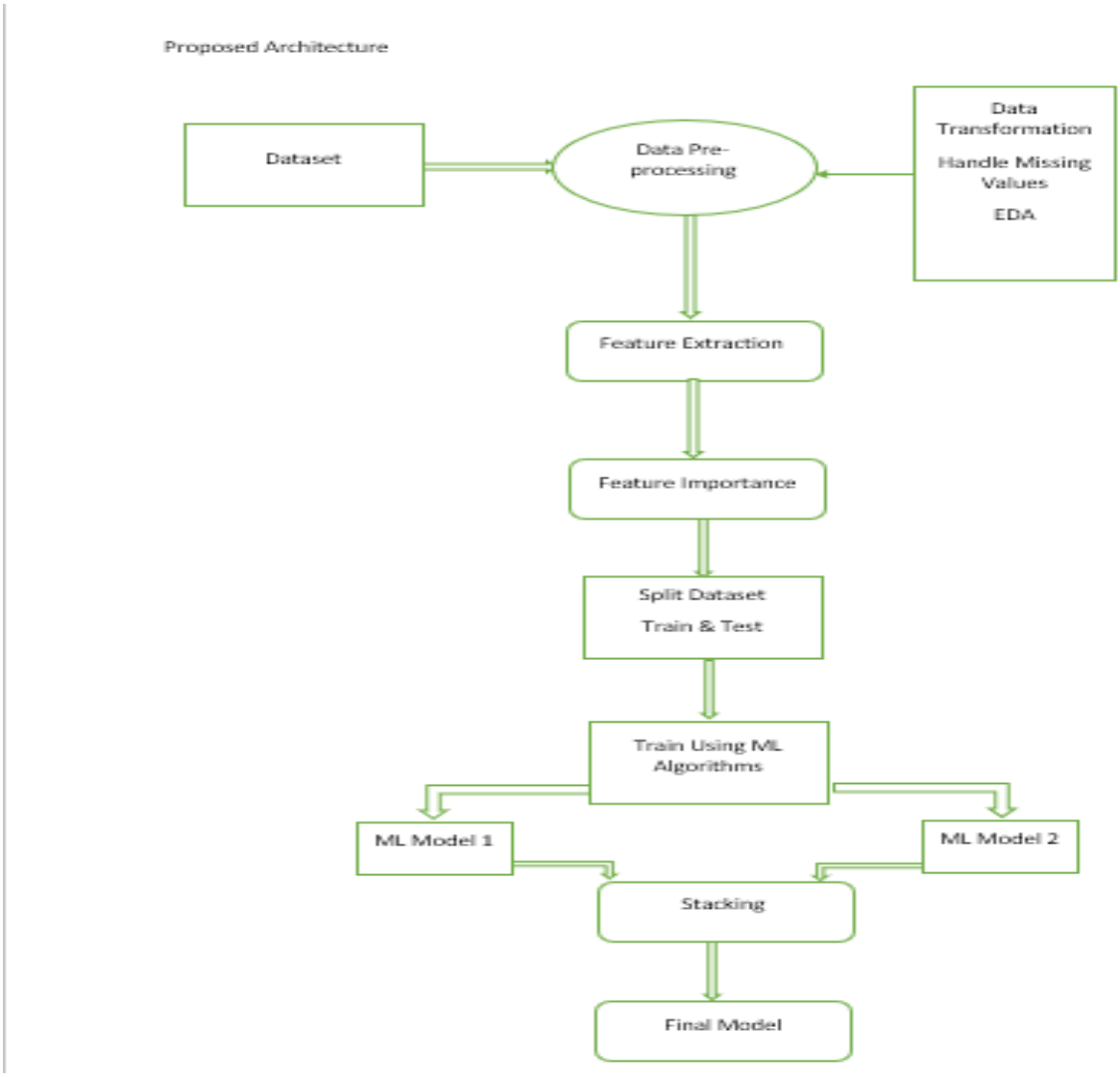
TABLE 1.1

SI.NO	References taken from other research/work		
	Title	Author's	Work
6	Automatic Software Bug Prediction Using Adaptive Artificial Jelly Optimization With Long Short-Term Memory.	R. Siva, Kaliraj, B. Hariharan, N. Premkumar	Proposed Automatic Software Bug Prediction Using Adaptive Artificial JellyOptimization WithLong Short-Term Memory
7	Artificial Intelligence for Software Engineering: An Initial Review on software Bug Detection and Prediction.	Julnar Ahmed Fadhil, Koh Wei, Kew Si Na	Artificial Intelligence bug prediction and proposed with the services
8	Software Bug Prediction using Machine Learning Approach.	Meenakshi, Dr. Satwinder Singh	Software bud prediction using historical data
9	Software Engineering Approach to Bug Prediction Model using Machine Learning as a Service (MLaaS)	Uma Subbiah, Muthu Ramachandran, Zaigham Mahmood	Software Engineering Approach to Bug Prediction Models using Machine Learning proposed using Service (MLaaS)
10	Using Defect Prediction to Improve the Bug Detection Capability of search-Based Software Testing	Anjana Perera	Defect Prediction to Improve the Bug Detection Capability of Search-Based Software Testing

III. METHODOLOGY

The methodology for software bug prediction encompasses a multifaceted approach, leveraging three distinct machine learning techniques: Support Vector Classification (SVC), Random Forest (RF), Nu-Support Vector Classification (NuSVC), and Multi-Layer Perceptron (MLP). The initial phase involves the meticulous collection of three real testing/debugging datasets from credible sources, ensuring a diverse representation of software fault scenarios. Data pre-processing procedures follow, emphasizing the normalization of numerical features and addressing class imbalances to foster unbiased model training. Feature selection techniques are applied to identify and retain pivotal software metrics, optimizing the models' predictive capabilities. Subsequently, the three selected machine learning classifiers SVC, RF, and NuSVC are individually trained on the pre-processed and balanced datasets, utilizing historical fault data and essential metrics. The MLP, as a neural network-based approach, provides an additional layer of complexity to the predictive modelling process. Ensemble learning, specifically stacking, is employed to synergistically combine the strengths of individual classifiers, culminating in the construction of a final bug prediction model. This model is then deployed for predicting future software faults, and a comprehensive analysis assesses the efficacy of each classifier, with a particular emphasis on evaluating the impact of ensemble learning. The methodology concludes by suggesting avenues for future research, including the exploration of other machine learning techniques and the incorporation of additional software metrics to further refine predictive accuracy and model generalization.

IV. PROPOSED ARCHITECTURE

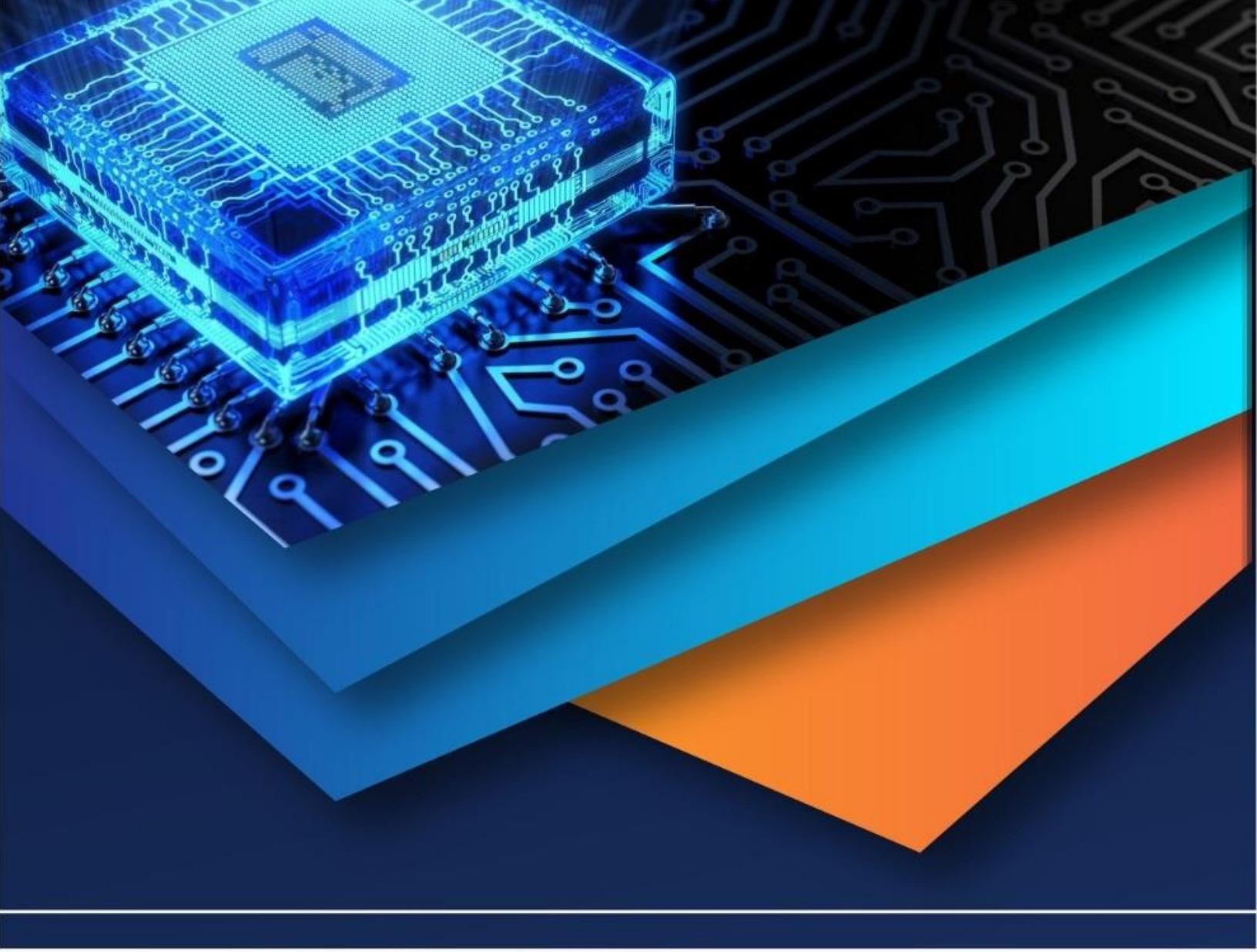


V. CONCLUSION

Software bug prediction is a technique in which a prediction model is created in order to predict the future software faults based on historical data. Various approaches have been proposed using different datasets, different metrics and different performance measures. This paper evaluated the using of machine learning algorithms in software bug prediction problem. Three machine learning techniques have been used, which are SVC, RF, NuSVC and MLP. The evaluation process is implemented using three real testing/debugging datasets. Results reveal that the ML techniques are efficient approaches to predict the future software bugs. The comparison results showed that the RF classifier has the best results over the others. Moreover, experimental results showed that using stacked approach provides a better performance for the prediction model than other approaches. As a future work, we may involve other ML techniques and provide an extensive comparison among them. Furthermore, adding more software metrics in the learning process is one possible approach to increase the accuracy of the prediction model.

REFERENCES

[1] Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, Fatima Alsarayrah, “Software Bug Prediction using Machine Learning Approach”, (IJACSA),Vol.9, No. 2, 2018
[2] Song Wang ,Junjie Wang, Jaechang Nam, Nachiappan Nagappan, “Continuous Software Bug Prediction”, ESEM ,2021
[3] Syahana Nur’Ain Saharudin, Koh Tieng Wei and Kew Si Na, “Machine Learning Techniques for Software Bug Prediction: A Systematic Review”, Journal of Computer Science Vol.16 (11):1558.1569, 2020
[4] Syed Rizwan, Wang Tiantian, Su Xiaohong, Salahuddin, “Empirical Study on Software Bug Prediction”, DOI: 10.1145,2017
[5] S. Delphine, M. Farida Begam, M. Floramangry, “Software Bug Prediction Using Supervised Machine Learning Algorithms”, IEEE, 978-1-5386, 2019
[6] R. Siva , Kaliraj S , B. Hariharan, N. Premkumar, “ Automatic Software Bug Prediction Using Adaptive Artificial Jelly Optimization With Long Short-Term Memory”,Wireless Personal Communications,132:1975-1998,2023
[7] Julanar Ahmed Fadhil, Koh Tieng Wei and Kew Si Na, “Artificial Intelligence for Software Engineering”: An Initial Review on Software Bug Detection and Prediction, Journal of Computer Science, Vol.16 (12):1709-1717
[8] Meenakshi, Dr. Satwinder Singh, “Software Bug Prediction using Machine Learning Approach (IRJET)”, Vol. 06(04):2395-0056
[9] Uma Subbiah, Muthu Ramachandran and Zaigham Mahmood, “Software Engineering Approach to Bug Prediction Models using Machine Learning as a Service (MLaaS) ”,Special Session on Software Engineering for Service and Cloud Computing, SE-CLOUD 2018
[10] Anjana Perera, “Using Prediction to Improve the bug Detection Capability of Search Based Software Testing”, Virtual Event, 2020
[11] Tracy Hall, Sarah Beecham, David Bowes, David Gray, and Steve Counsell, “A systematic literature review on fault prediction performance in software engineering”. IEEE Transactions on Software Engineering 38, 6 (2011), 1276–1304
[12] Akmel, F., Birihanu, E., & Siraj, B. (2017), “A literature review study of software defect pre-diction using machine learning techniques” Int. J. Emerg. Res. Manag. Technol, 6(6), 300-306
[13] Pandey, S. K., Mishra, R. B., & Triphathi, A. K, “ Software bug prediction prototype using Bayesian network classifier: A comprehensive model” Procedia Computer Science, 132, 1412–1421
[14] Ambros, M. Lanza, and R. Robbes, "An Extensive Comparison of Bug Prediction Approaches", In Proc. IEEE Seventh Working Conf. Mining Software Repositories, pp. 31-41, 2010
[15] Kumar, L. S, “Effective fault prediction model developed using least square support vector machine (LSSVM)”. Journal of Systems and Software, 137, 686-712.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY