International Journal of Scientific Modern Research and Technology
(Volume: 11, Issue: 2, Number: 3) Paper ID: IJSMRT-23110203
ISSN: 2582-8150

Software Bug Prediction using Support Vector Machine with Independent Component Analysis

Vishal Kumar Singh[1], Neelesh Rai[2]
[1] Research Scholar, [2]Head and Assistant Professor
[1, 2]Department of CSE, MITS, Bhopal, India

Abstract: Predicting software flaws—known as software bug prediction (SBP)—is crucial to the software industry as a whole. This is so because it is possible to increase software quality, reliability, efficiency, and cost by anticipating software issues at an earlier phase. Although various methods have been presented in the literature, establishing a reliable bug prediction model remains a formidable challenge. Successful software development relies heavily on error prediction. Predicting when bugs may appear is now a crucial discussion point throughout development and upkeep. As a result, it is crucial to foresee software defects at early phases of the SDLC. The goal here is to figure out how to foresee errors so that we can make software that is high-quality, stable, efficient, and affordable. Bugs are a big problem for large, complicated software development projects. Here, SVM-ICA is used to the task of analyzing huge datasets for software bug prediction in order to provide more precise findings that are functional in a wide range of contexts. Each model's efficacy is assessed, and then cross validation is carried out, with the findings being visualized thereafter. At last, we compared the proposed SVM-ICA model to several other ML contenders. F1-Score, precision, recall, and accuracy of predictions all rise by 15.39%.
Key Terms: SVM-ICA, Accuracy, Precision, Recall, F1-Score.

***How to cite this article**: Vishal Kumar Singh, Neelesh Rai, "Software Bug Prediction using Support Vector Machine with Independent Component Analysis", Published in International Journal of Scientific Modern Research and Technology (IJSMRT), ISSN: 2582-8150, Volume-11, Issue-2, Number-3, June-2023, pp.11-16, URL: www.ijsmrt.com/wp-content/uploads/2023/08/IJSMRT-23110203.pdf*

IJSMRT-23110203

1. INTRODUCTION

A developer might make a mistake during coding that would prevent a given feature from becoming fully functional. This blunder is the malfunction-causing flaw in the program. The problem encountered by Grace Murray Hopper, a pioneer in the field of computers, is the inspiration for the term "bug." Working on an electromechanical computer, he ran across a problem that slowed it down. In the process of troubleshooting, he discovered a moth lodged deep inside the machine. Since then, the term "bug" has come to be used for any computer-related problem. There is a direct correlation between the number of problems in software and the amount of time and money spent fixing it. In most cases, even when the program is deployed meticulously, there are still defects that might create problems. In addition, it is a significant task in software engineering to create a bug prediction model that can identify problematic modules at an early stage. Predicting where bugs may appear in software is a crucial part of the software development process. Predicting the faulty components of a piece of software before releasing it

ISSN: 2582-8150

may increase user happiness and boost the program's overall performance. Early problem prediction also enhances software's ability to adjust to new settings and makes better use of available resources.

II. PREVIOUS WORK

There has been a rise in the adoption of automated approaches for software defect prediction (SDP), most often those based on machine learning (ML). To capture the semantic information presented in bug tracking systems, current ML-based techniques need manually extracted features, which is tedious and time-consuming. Using deep learning (DL) methods, experts can now automatically extract and learn from high-dimensional, complicated datasets. The goal of this research was to take stock of where DL algorithms are in relation to SDP and rigorously catalog, summarize, and synthesize the existing literature on the topic. (Görkem Giray; Kwabena Ebo Bennin; mer Köksal; nder Babur; Bedir Tekinerdogan; 2023).

Predicting when a bug will appear is a difficult problem in software engineering and programming language study. The difficulty here is identifying the flawed code in a reliable way. There have been several attempts to solve the difficult challenge of fault prediction model development. Many issues have been addressed and resolved as a result of recent advancements in machine learning technology, notably the growth of deep learning approaches. In this report, we examine the state of the art in defect prediction using deep learning methods. (Akimova, E.N., Bersenev, A.Y., Deikov, A.A., Kobylkin Misilov, I.P., K.S. Konygin, and A.V. Mezentsev; 2021)

The broad interest in incorporating artificial intelligence (AI) capabilities into software and services has been sparked by recent advancements in machine learning within the IT industry. Because of this objective, businesses have improved their development methods. We present the results of an observational research we did on Microsoft's software development teams working on AI-based apps. Using our knowledge gained from creating both search and natural language processing AI apps and data science tools such as application diagnostics and bug reporting, we explore a nine-step workflow approach. (Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan; 2019)

Finding and defining activation functions that may boost neural network performance is a popular topic in the neural networks literature. Reviving interest in studying "trainable," "learnable," or "adaptable" activation functions throughout the learning process has been seen in the scientific community in recent years. They seem to improve the efficiency of the network. Many different kinds of models for a tunable activation function have been presented. In this article, we provide a comprehensive overview of these frameworks. We begin with a discussion of how the word "activation function" is used in the literature, then suggest a taxonomy of trainable activation functions, describe the shared and unique characteristics of modern and historical models, and conclude with a discussion of the primary benefits and drawbacks of this method. (Andrea Apicella, Francesco Donnarumma, Francesco Isgr, and Roberto Prevete; 2021).

III. PROBLEM FORMULATION

Based on previous research, we have identified the following issues:

- The identification of relevant software bugs is not perfectly retrieved.
- The retrieval of a software bug is not perfectly identified.
- The unidentified software bug may detect due to low accuracy.

IV. RESEARCH OBJECTIVES

The aims of the suggested effort are as follows:

- To improve precision for perfect retrieval of relevant software bugs.
- To improve recall for perfectly relevant software bugs in the retrieval process.
- To improve accuracy for exactness of software bug detection.

ISSN: 2582-8150

V. METHODOLOGY

The pseudocode of the proposed prediction model SVM-ICA (Support Vector Machine with Independent Component Analysis) is as follows:

inputs: Determine the various training and test data.
outputs: Determine the calculated accuracy
select the optimal value of cost and gamma for SVM.
while (stopping condition is not met) **do**
implement SVM train step for each data point.
implement SVM classify for testing data points.
end while
return accuracy

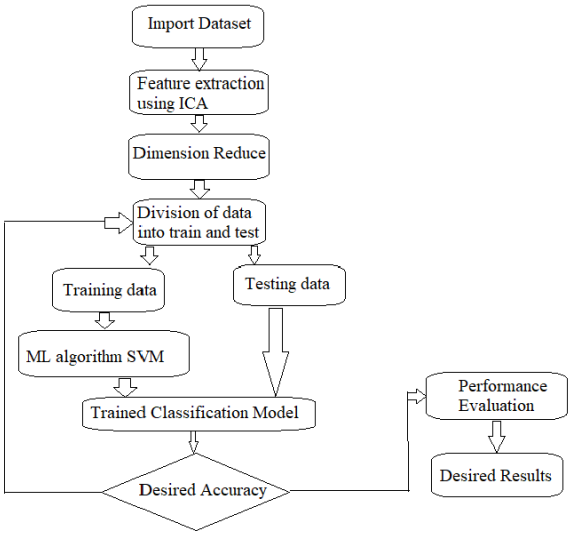

Figure 1: Proposed Model of SVM-ICA (Proposed Methodology)

VI. RESULTS AND ANALYSIS

The following observations are performed on anaconda navigator with python 3.11.1 with jupyter lab toolbox. The proposed procedure SVM-ICA perform on (PROMISE Dataset) JS1.csv and calculate precision, recall, F1-Score and accuracy parameters are calculated as follows:

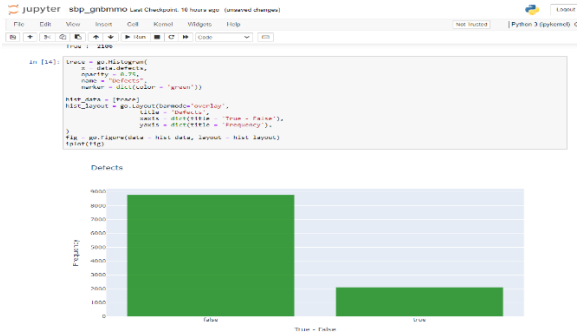

Figure 2: Evaluation of Defects for SVM-ICA (Proposed Prediction Model)

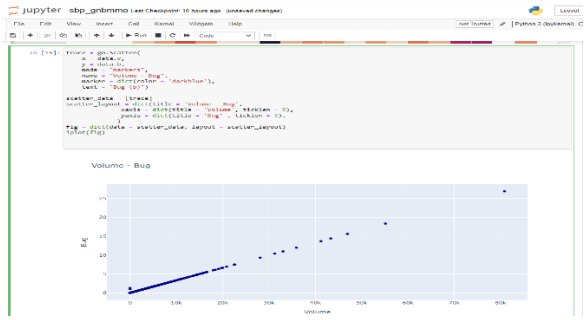

Figure 3: Evaluation of Volume of Bug for SVM-ICA (Proposed Prediction Model)

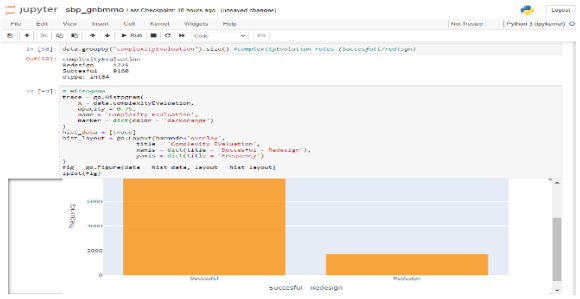

Figure 4: Evaluation of Bug Frequency in Software for SVM-ICA (Proposed Prediction Model)

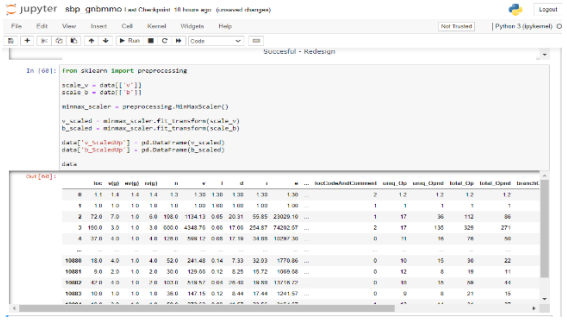

Figure 5: Evaluation of Min-Max Normalization of SVM-ICA (Proposed Prediction Model)

ISSN: 2582-8150

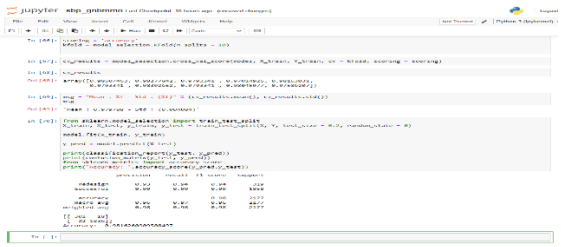

Figure 6: Calculation of confusion matrix, precision, recall, F1-Score and accuracy among different models and SVM-ICA (Proposed Prediction Model)

Table 1: Estimation of Confusion Matrix among different models and SVM-ICA (Proposed Prediction Model)

Models	Prediction	Module has bugs	
		No	Yes
Random Forest	Classifier predicts no bugs	1652	80
	Classifier predicts some bugs	341	103
Naïve Bayes	Classifier predicts no bugs	1663	69
	Classifier predicts some bugs	364	80
Logistic Regression	Classifier predicts no bugs	1701	31
	Classifier predicts some bugs	415	29
Decision Tree	Classifier predicts no bugs	1453	279
	Classifier predicts some bugs	268	176
ANN	Classifier predicts no bugs	1738	36
	Classifier predicts some bugs	339	63
SVM-ICA (Proposed)	**Classifier predicts no bugs**	**1836**	**18**
	Classifier predicts some bugs	**22**	**301**

Table 2: Estimation of Precision, Recall, F1-Score and Accuracy among different models an SVM-ICA (Proposed Prediction Model)

Models	Precision	Recall	F1-Score	Accuracy
Random Forest	0.95	0.83	0.9	80.65 %
Naïve Bayes	0.96	0.82	0.88	80.10 %
Logistic Regression	0.98	0.8	0.88	79.5 %
Decision Tree	0.83	0.84	0.83	74.86 %
ANN	0.97	0.84	0.9	82.77 %
SVM-ICA (Proposed)	**0.99**	**0.99**	**0.99**	**98.16 %**

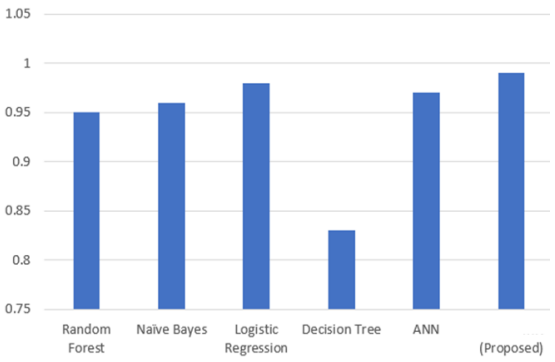

Figure 7: Graphical Analysis of Precision among different models and SVM-ICA (Proposed Prediction Model)

The above graph show that the proposed model gives better precision for bug prediction as compare than other models. The precision of SVM-ICA is improve

ISSN: 2582-8150

by 0.01 as compare than Logistic Regression prediction model.

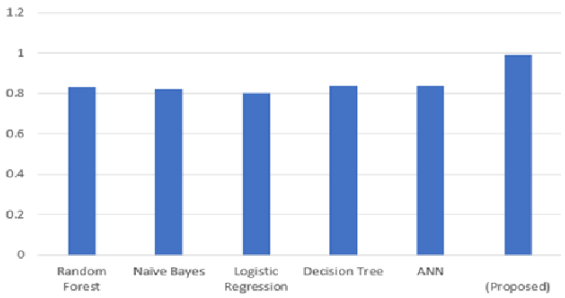

Figure 8: Graphical Analysis of Recall among different models and SVM-ICA (Proposed Prediction Model)

The above graph show that the proposed model gives better recall for bug prediction as compare than other models. The recall of SVM-ICA is improved by 0.15 as compare than Decision Tree and ANN prediction model.

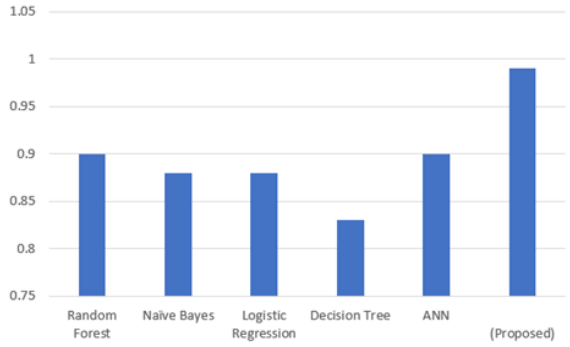

Figure 9: Graphical Analysis of F1-Score among different models and SVM-ICA (Proposed Prediction Model)

The above graph show that the proposed model gives better F1-Score for bug prediction as compare than other models. The F1-Score of SVM-ICA is improve by 0.09 as compare than Random Forest and ANN prediction model.

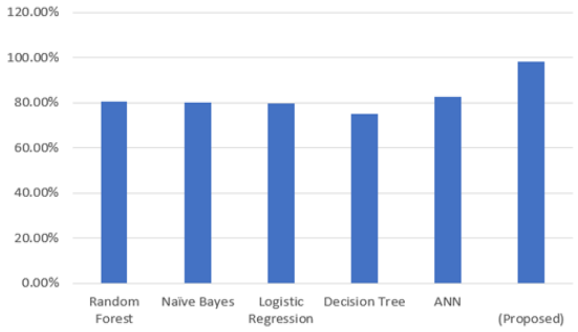

Figure 10: Graphical Analysis of Accuracy among different models and SVM-ICA (Proposed Prediction Model)

The above graph show that the proposed model gives better Accuracy for bug prediction as compare than other models. The Accuracy of SVM-ICA is improved by 15.39 % as compare than ANN prediction model.

VII. CONCLUSIONS

These are supposed to have conclusions, and these does: In comparison to ANN, the suggested model provides more accurate predictions. There is a 15.39% increase in precision. When compared to Logistic Regression, the suggested model provides more accurate predictions. There is a 1% increase in accuracy. Third, the proposed model outperforms the Decision Tree and ANN in terms of prediction recall. A 15% increase in recall is achieved. The suggested model outperforms both Random Forest and ANN in terms of prediction F1-score. There is a 9% rise in F1-Score. Consequently, SVM-ICA (Support Vector Machine with Independent Component Analysis) is a more accurate approach for predicting software bugs. 6.2 Future Work Suggestions Our suggested technique increases bug prediction accuracy and is useful for future development. Future improvements should include testing the accuracy with other datasets and using more AI techniques to validate the accuracy estimate. Due to the massive quantity of data needed for performance estimation of train data, the proposed model has a processing time restriction. To better predict the system's performance in the future, the same algorithms will be used to data in real time.

ISSN: 2582-8150

REFERENCES

[1] Görkem Giray, Kwabena Ebo Bennin, Ömer Köksal, Önder Babur, Bedir Tekinerdogan, "On the use of deep learning in software defect prediction", The Journal of Systems & Software, 2023.

[2] G. P. Bhandari and R. Gupta, "Machine learning based software fault prediction utilizing source code metrics," in 250 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), 2018, pp. 40–45.

[3] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," Appl. Soft Comput., vol. 27, pp. 504–518, Feb. 2015.

[4] L. Son et al., "Empirical Study of Software Defect Prediction: A Systematic Mapping," Symmetry (Basel)., vol. 11, no. 2, p. 212, Feb. 2019.

[5] C. W. Yohannese and T. Li, "A Combined-Learning Based Framework for Improved Software Fault Prediction," Int. J. Comput. Intell. Syst., vol. 10, no. 1, p. 647, Dec. 2017.

[6] A. Hudaib et al., "ADTEM-Architecture Design Testability Evaluation Model to Assess Software Architecture Based on Testability Metrics," J. Softw. Eng. Appl., vol. 08, no. 04, pp. 201–210, Apr. 2015.

[7] S. Elmidaoui, L. Cheikhi, and A. Idri, "Towards a Taxonomy of Software Maintainability Predictors," Springer, Cham, 2019, pp. 823–832.

[8] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics," in 2009 3rd International Symposium on Empirical Software Engineering and Measurement, 2009, pp. 367–377.

[9] "PROMISE DATASETS PAGE." [Online]. Available: http://promise.site.uottawa.ca/SERepository/datasets page.html. [Accessed: 01-Jul 2019].

[10] R. Malhotra and S. Kamal, "An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data," Neurocomputing, vol. 343, pp. 120–140, May 2019.