ISSN 2277-2685
IJESR/April-June. 2025/ Vol-15/Issue-2s/799-805
Nandari Sai Nikitha *et. al.,* / International Journal of Engineering & Science Research

Smart Ensemble Approach For Software Defect Prediction

A V S Radhika[1] , Nandari Sai Nikitha[2], Ballure Suprathika[3].
[1]Assistant Professor, Department of CSE, Bhoj Reddy Engineering College or Women.
[2,3]B. Tech Students, Department of CSE, Bhoj Reddy Engineering College for Women.

ABSTRACT *Software defect prediction is essential for improving software quality and reducing testing costs. The main goal is to detect and forward just faulty modules to the testing phase. This study presents an advanced ensemble-based model for software fault prediction that integrates many classifiers. The suggested approach utilizes a two-stage prediction technique to identify damaged modules. Initially, four supervised machine learning techniques are utilized: Random Forest, Support Vector Machine, Naïve Bayes, and Artificial Neural Network. These algorithms undergo repeated parameter optimization to get maximal accuracy. In the subsequent phase, the predicted accuracy of the different classifiers is amalgamated into a voting ensemble to get the final predictions. This ensemble method enhances the precision and dependability of fault forecasts. Seven historical defect datasets from the NASA MDP repository, namely CM1, JM1, MC2, MW1, PC1, PC3, and PC4, were used to develop and assess the suggested defect prediction system. The findings indicate that the suggested intelligent system for each dataset attained exceptional accuracy, surpassing twenty advanced defect prediction approaches, including base classifiers and ensemble algorithms.*

Keywords: *Machine learning, software defect prediction, heterogeneous classifiers, random forest, support vector machine, naïve Bayes.*

INTRODUCTION

Accelerated globalization has converted our globe into an interconnected village, with the software sector playing a pivotal role in advancing development. In today's digitally networked world, software applications have become essential to our global civilization, underpinning everyday activities, companies, and crucial infrastructure. This effect has strengthened, especially during the COVID-19 epidemic, which has expedited our dependence on online platforms for communication, trade, and distant labor.

In a Software Development Life Cycle (SDLC), the workflow from the development team to the Quality Assurance (QA) team often encompasses many phases. The development team first delivers the software code to the QA team for testing. The QA team meticulously assesses the program, detecting and documenting errors or problems. The iterative feedback loop between development and quality assurance continues until a high-quality, defect-free program is achieved.

The product is attained [6], [7]. Figure 1 illustrates the process from the Development team to the QA team inside the Software Development Life Cycle (SDLC). Nonetheless, achieving defect-free software presents significant obstacles. Three pivotal aspects that significantly impact software quality assurance are time, financial resources, and the availability of proficient personnel. The industry's increasing demand requires the development of efficient testing methodologies to maximize precious resources while upholding the highest standards of software quality [8]. This is where software defect prediction (SDP) assumes prominence. Software Defect Prediction

(SDP) involves using historical data and machine learning (ML) methodologies to anticipate and detect probable flaws in software systems prior to the testing process [9]. It examines the intricate array of software parameters, including code complexity, size, and historical defect data, to develop models that can assess the probability of problems [10]. The incorporation of software defect prediction alters the conventional development-to-QA process. The feedback loop is modified by proactively detecting possible flaws prior to the testing phase. The predictive insights enable developers to identify and resolve possible problems prior to the software's delivery to the QA team. This procedure optimizes efficiency and diminishes the conventional exchanges between development and QA. This transition fosters a more efficient and economical software development life cycle [11]. Figure 2 illustrates a visual depiction of the diminished feedback loop resulting from the implementation of the SDP model.

In the realm of software defect prediction, classification approaches are crucial. They include classifying data into categories or labels, making them very essential for detecting possible software problems. Classification approaches include decision trees, logistic regression, support vector machines, among others [12], [13]. These strategies facilitate the proactive evaluation and resolution of software quality issues. Historically, several research have used classification approaches to improve the precision of defect prediction models. Nonetheless, previous research in this domain exhibits limitations, including issues associated with classification methodologies, overfitting, and underfitting [14].

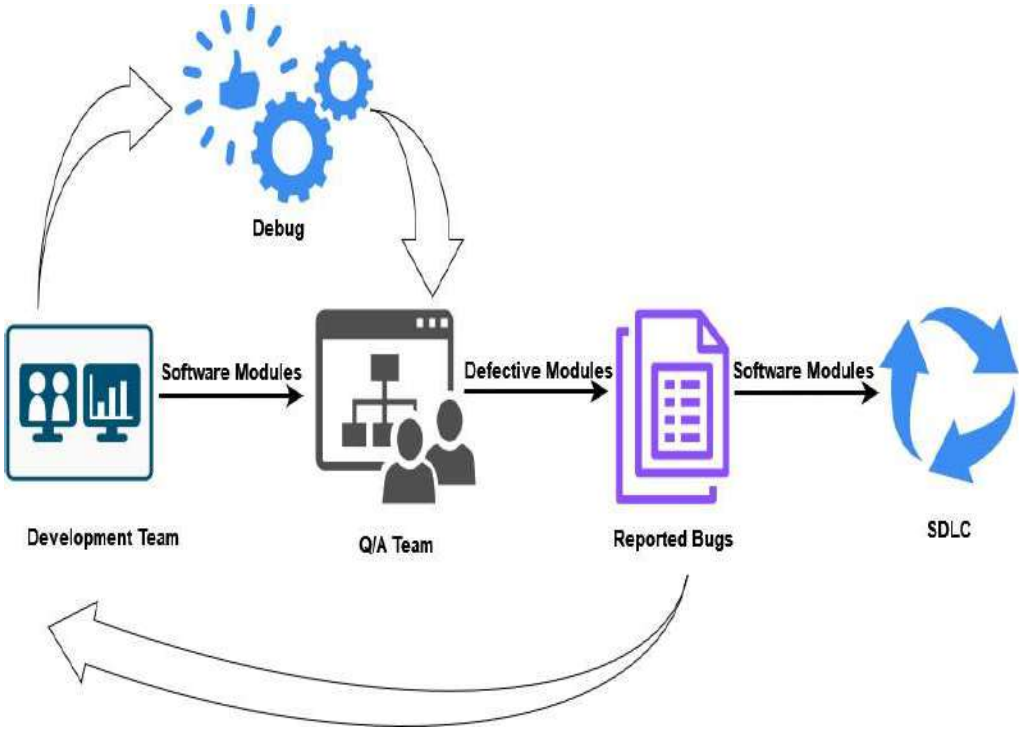

FIGURE 1. Development-to-QA workflow in SDLC.[1]

Alongside classification, Ensemble Modeling (EM) has developed as a promising approach that amalgamates predictions from various machine learning models to enhance overall performance [15]. Ensemble approaches, such bagging, boosting, stacking, and random forest, enhance the field by addressing intrinsic problems [15]. Ensemble approaches mitigate the hazards of overfitting, underfitting, and biases inherent in individual classifiers by consolidating the predictions of numerous base classifiers. They have shown their value in improving the accuracy and resilience of defect prediction models by mitigating the inherent biases of individual classifiers [16], [17]. Researchers have faced a common obstacle: the intrinsic vulnerability of ensemble approaches to biases that might affect their effectiveness [18], [19]. Figure 3 provides a summary of software fault prediction with machine learning approaches.

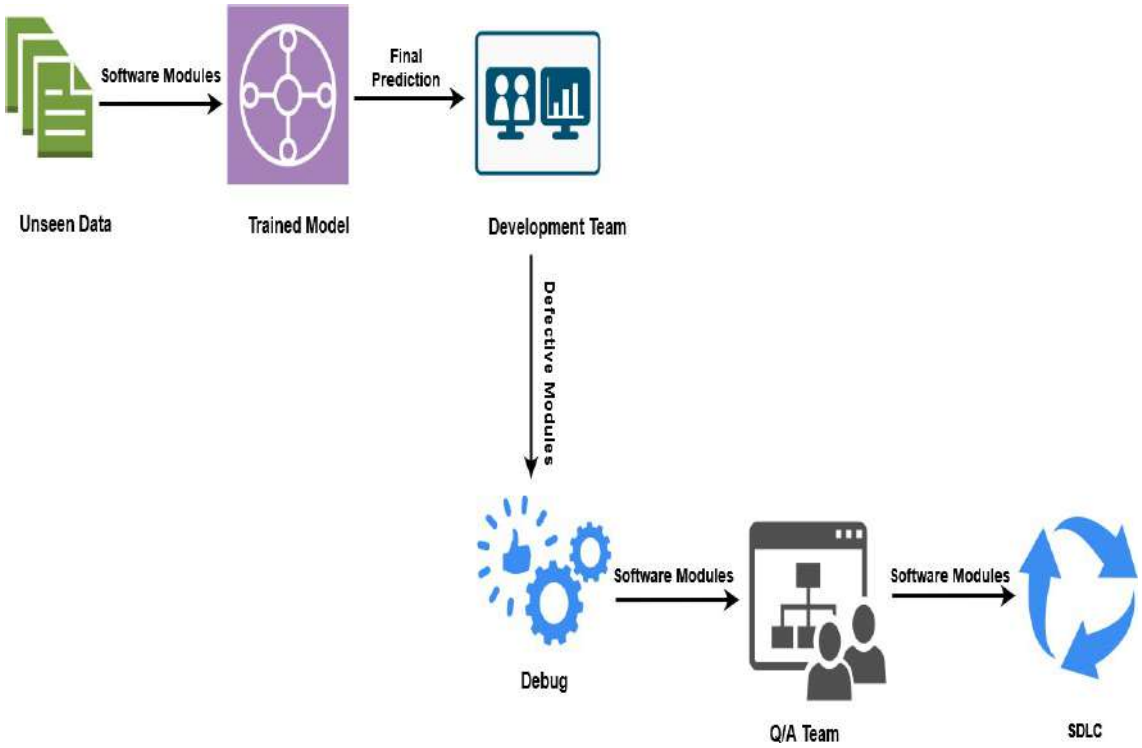

FIGURE 2. Development-to-QA workflow using SDP.[1]

LITERATURE REVIEW

Authors in [3] created an intelligent cloud-based SDP system that integrates data fusion with decision-level machine learning fusion methodologies. The system amalgamated predicted performance from three classifiers: naïve Bayes (NB), artificial neural network (ANN), and decision tree (DT) using a fuzzy logic-based fusion approach. The suggested system, assessed using NASA information, surpassed other methodologies and sought to attain high-quality software with reduce expenses. A comprehensive comparative analysis of multiple classifiers was performed regarding software defect prediction [23]; the authors examined ten machine learning algorithms, namely

Decision Tree, Naive Bayes, K-Nearest Neighbor, Support Vector Machine, Random Forest, Extra Trees, Adaboost, Gradient Boosting, Bagging, and Multi-Layer Perceptron. The investigation used benchmark NASA datasets from the PROMISE repository, including CM1, KC1, KC2, JM1, and PC1. The experimental findings indicated that the used algorithms attained superior average accuracy rates on the PC1 dataset. The Random Forest classifiers with the PCA method demonstrated enhanced average performance across the datasets. Figure 4 depicts the categorization procedure.

In [24], the authors tackled the issue of handling a substantial quantity of software defect reports in software development and maintenance. A software defect prediction (SDP) model using LASSO–SVM was developed to enhance prediction accuracy. The model integrated feature selection using the minimal absolute value compression and selection approach with the support vector machine algorithm. This methodology significantly improved predictive accuracy, with simulation outcomes demonstrating an accuracy of 93.25% and 66.67%, a recall rate of 78.04%, and an f-measure of 72.72%. The suggested model surpassed conventional approaches for accuracy and speed. In [25], researchers introduced a cloud-based system for real-time software fault prediction, evaluating four back-propagation training methods. Bayesian regularization (BR) proved to be the most efficacious. The approach used a fuzzy layer to determine the optimal training function depending on performance metrics. NASA datasets accessible to the public were used for assessment, applying several metrics. The findings indicated that BR surpassed other training algorithms and commonly used machine-learning methodologies.

The researchers in [26] used machine learning techniques to evaluate the efficacy of varying tree quantities in the RF method for software defect prediction, using the RAPIDMINER machine learning tool. They evaluated the efficacy of varying quantities of trees inside the RF algorithm. The findings demonstrate that augmenting the quantity of trees marginally enhances accuracy, achieving a peak accuracy of 99.59% and a nadir accuracy of 85.96%. The study emphasized the efficacy of the RF algorithm in predicting software defects, especially when using about one hundred trees.

METHODOLOGY

The suggested study presents an intelligent ensemble-based methodology for software fault prediction. This model employs several supervised machine learning classifiers to improve accuracy. The novel methodology seeks to effectively tackle issues in forecasting software flaws. The outputs of individual base classifiers are aggregated using a voting ensemble model, effectively using the advantages inherent in the proposed VESDP approach. This ensemble method improves the model's predictive capability by integrating many classifier outputs, resulting in a more reliable and precise software fault prediction. Figure 5 illustrates an overview of the suggested paradigm. The suggested VESDP model consists of two layers: training and testing. The training layer has three stages:

1) data preprocessing 2) base classification 3) ensemble classification.

PERFORMANCE EVALUATIONL

In the aforementioned formulations, $\alpha\ \lambda$ denotes the faulty modules in the software that were accurately identified as defective by the model; conversely, $\alpha\ \theta$ signifies the non-defective modules in the software

that were properly classified as non-defective by the model. The βλ and βθ values show a discrepancy between actual and projected values. βλ implies that the module was non-faulty however classified as defective, while βθ signifies that the module was defective yet classified as non-defective by the model.

RESULTS AND DISCUSSION

This study created a voting ensemble-based software defect prediction model (VESDP). Seven publicly available NASA datasets (CM1, JM1, MC2, MW1, PC1, PC3, and PC4) were retrieved from the MDP repository to conduct the tests. During the preparation phase, the datasets underwent three sub-activities: partitioning, cleansing, and normalization. The partitioning process splits the datasets into two subsets, namely training and testing, at a ratio of 70:30 according to the class-based splitting criterion [53]. Initially, four diverse supervised classification methods, including RF, SVM, NB, and ANN, were used to train the model. The classifiers were repeatedly tuned until each achieved the maximum accuracy for the used datasets. The prediction accuracy of separate classifiers is amalgamated using the voting ensemble approach, hence enhancing the performance of the proposed model. Eight commonly used assessment metrics were employed to assess performance [22], [54]. All performance metrics have been obtained via a confusion matrix generated by tools given by Python [23], [55]. The outcomes derived from both the training and testing datasets for each classifier are detailed below.

FIGURE 3. Comparison graph.[1]

CONCLUSION

Software defect prediction seeks to discover erroneous modules prior to the testing process, allowing for concentrated testing on those modules most likely to have flaws. An effective defect prediction model may save software development expenses by reducing the resources allocated to quality assurance operations during testing. This paper suggested an intelligent ensemble approach for software fault prediction. The model was executed with benchmark datasets obtained from the NASA defect repository. The proposed model amalgamated the predicted accuracy of four diverse supervised classifiers by the voting ensemble classification approach. Eight performance metrics were used for statistical analysis. A comparative study was undertaken to demonstrate the efficacy of the method used in the suggested model against state-of-the-art techniques. The developed VESDP model surpassed contemporary research and demonstrated its efficacy in the software fault prediction process.

LIMITATION OF PROPOSED MODEL

The training data substantially influences the efficacy of any machine-learning model, including ensemble models. The training dataset's inconsistency, absence of data, or noise may adversely impact the model's predictive capability. The requirements, development methodologies, and code used in software development are always evolving. An ensemble model trained on historical data may find it challenging to adjust to unforeseen changes in project dynamics or novel development paradigms. The algorithm analyzes historical data to provide projections based on identified patterns. Should the present task significantly diverge from the projects inside the training dataset, the model's efficacy may be compromised.

FUTURE WORK

Future research should investigate sophisticated feature selection methodologies using evolutionary algorithms or bat search algorithms to improve the robustness of the suggested model in software fault prediction. Moreover, integrating layered ensemble methodologies such as bagging, boosting, and stacking might enhance the dependability of predictions. These upgrades would facilitate the ongoing advancement of defect prediction models, improving their usefulness in practical software development contexts.

REFERENCES

[1] Misbah Ali, Tehseen Mazhar, Yasir Arif, Shaha Al-Otaibi, (Member, Ieee), Yazeed Yasin Ghadi, Tariq Shahzad[5], Muhammad Amir Khan[6], And Habib Hamam, (Senior Member, IEEE) Software Defect Prediction Using an Intelligent Ensemble-Based Model, VOLUME 12, 2024, DOI *10.1109/ACCESS.2024.3358201, IEEE ACCESS.*

[2] Z. M. Zain, S. Sakri, and N. H. A. Ismail, ''Application of deep learning in software defect prediction: Systematic literature review and meta- analysis,'' *Inf. Softw. Technol.*, vol. 158, Jun. 2023, Art. no. 107175, doi: 10.1016/j.infsof.2023.107175.

[3] M. Unterkalmsteiner et al., ''Software startups—A research agenda,'' 2023, *arXiv:2308.12816*.

[4] S. Aftab, S. Abbas, T. M. Ghazal, M. Ahmad, H. A. Hamadi, C. Y. Yeun, and M. A. Khan, ''A cloud-based software defect prediction system using data and decision-level machine learning fusion,''

Mathematics, vol. 11, no. 3, p. 632, Jan. 2023, doi: 10.3390/math11030632.

[1] S. Goyal, ''Heterogeneous stacked ensemble classifier for software defect prediction,'' in *Proc. 6th Int. Conf. Parallel, Distrib. Grid Comput. (PDGC)*, Waknaghat, India, Nov. 2020, pp. 126–130, doi: 10.1109/PDGC50313.2020.9315754.

[2] S. Mehta and K. S. Patnaik, ''Stacking based ensemble learning for improved software defect prediction,'' in *Proc. 5th Int. Conf. Microelec- tron., Comput. Commun. Syst.*, vol. 748, 2021, pp. 167–178.

[3] M. Shafiq, F. H. Alghamedy, N. Jamal, T. Kamal, Y. I. Daradkeh, and M. Shabaz, ''Retracted: Scientific programming using optimized machine learning techniques for software fault prediction to improve software quality,'' *IET Softw.*, vol. 17, no. 4, pp. 694–704, Jan. 2023, doi: 10.1049/sfw2.12091.

[4] Y. Tang, Q. Dai, M. Yang, T. Du, and L. Chen, ''Software defect prediction ensemble learning algorithm based on adaptive variable sparrow search algorithm,'' *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 6, pp. 1967–1987, Jan. 2023, doi: 10.1007/s13042-022-01740-2.

[5] S. Goyal, ''3PcGE: 3-parent child-based genetic evolution for software defect prediction,'' *Innov. Syst. Softw. Eng.*, vol. 19, no. 2, pp. 197–216, Jun. 2023, doi: 10.1007/s11334-021-00427-1.

[6] J. Liu, J. Ai, M. Lu, J. Wang, and H. Shi, ''Semantic feature learning for software defect prediction from source code and external knowledge,'' *J. Syst. Softw.*, vol. 204, Oct. 2023, Art. no. 111753, doi: 10.1016/j.jss.2023.111753.

[7] A. K. Gangwar and S. Kumar, ''Concept drift in software defect prediction: A method for detecting and handling the drift,'' *ACM Trans. Internet Technol.*, vol. 23, no. 2, pp. 1–28, May 2023, doi: 10.1145/3589342.

[8] M. S. Alkhasawneh, ''Software defect prediction through neural network and feature selections,'' *Appl. Comput. Intell. Soft Comput.*, vol. 2022, pp. 1–16, Sep. 2022, doi: 10.1155/2022/2581832.

[9] T. F. Husin and M. R. Pribadi, ''Implementation of LSSVM in classification of software defect prediction data with feature selection,'' in *Proc. 9th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI)*, Jakarta, Indonesia, Oct. 2022, pp. 126–131, doi: 10.23919/EECSI56542.2022.9946611.