International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

Improving Software Defect Prediction Using Stacking and LightGBM

1st Chattu Naga Lakshmi
Assistant Professor,Department of Computer Science and Engineering
Seshadri Rao Gudlavalleru Engineering College
Andhra Pradesh - 521356, India
chattu.naga@gmail.com

2nd Jakku Kumarswami
Department of Computer Science and Engineering
Seshadri Rao Gudlavalleru Engineering College
Andhra Pradesh - 521356, India
harijakku2005@gmail.com

3rd Katragadda Naga Kavya
Department of Computer Science and Engineering
Seshadri Rao Gudlavalleru Engineering College
Andhra Pradesh - 521356, India
kavyakatragadda41@gmail.com

4th Kurakula Tarun Kumar
Department of Computer Science and Engineering
Seshadri Rao Gudlavalleru Engineering College
Andhra Pradesh - 521356, India
ktktarunkurakula@gmail.com

5th Jarugu Venkata Yaswanth
Department of Computer Science and Engineering
Seshadri Rao Gudlavalleru Engineering College
Andhra Pradesh - 521356, India
yaswanthjarugu892@gmail.com

Abstract: Software defect prediction is crucial for improving software quality and reducing testing effort. It helps identify faulty modules early in the development process. This work presents an intelligent ensemblebased software defect prediction system featuring stacking and safe online deployment. The proposed extension includes a Stacking Classifier made up of Decision Tree, Random Forest, and LightGBM. This setup allows for effective learning from different defect patterns and reduces bias in individual models.The stacking approach offers better predictive performance and reliability compared to traditional voting-based ensembles. To ensure practical use, the model is deployed with a Flask-based web interface that includes user authentication, allowing for secure and interactive defect analysis. Experimental evaluation on benchmark NASA MDP datasets (CM1 and PC4) shows that the extended model achieves impressive accuracy, reaching up to 93.5%. It outperforms baseline ensemble methods. The results confirm that this improvement in software fault prediction is secure, accurate, and can scale effectively.

Keywords: Software Defect Prediction, Ensemble Learning, Stacking Classifier, LightGBM, Machine Learning, Voting Ensemble, Software Quality Assurance, Flask Web Framework, Secure Model Deployment, NASA MDP Datasets

INTRODUCTION

To reduce testing, modern software engineering uses software defect prediction (SDP) to find faulty software modules early in development, reduce costs, and minimize post-release failures. As software systems become more complex and release cycles become quicker, traditional quality assurance practices often fall short of ensuring high reliability. Therefore, data-driven and smart prediction models are increasingly important for proactive defect detection. Recent studies point out the significance of using source code metrics, historical defect data, and new learning methods to improve prediction performance [1]. Recent research has looked into semantic and contextual feature learning to raise defect prediction accuracy. N. Rajeswari et al. [1] showed that combining source code semantics with external knowledge representations greatly enhances defect prediction. This approach captures deeper structural and contextual information that goes beyond traditional metrics. It highlights the limitations of handcrafted features and encourages the use of learning-based models that can automatically extract meaningful representations. However, these methods often require high computational resources and may

Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

have trouble generalizing across different projects. Deep learning techniques have pushed the field of software defect prediction forward by allowing automatic feature learning from large datasets. A thorough systematic review and meta-analysis by Rajeswari Nakka et al. [2] showed that deep neural networks frequently outperform traditional machine learning models in various defect prediction scenarios. Despite their effectiveness, deep learning models face challenges such as class imbalance, overfitting, and lack of interpretability, which can make them less practical for industrial use without further optimization.

To solve scalability and deployment issues, researchers have proposed cloudbased and hybrid prediction frameworks. Data-level and decision-level machine learning are combined in Aftab et al.'s cloud-based software defect prediction technique [3]. Their method showed improved accuracy and scalability. However, reliance on cloud infrastructure brings latency, security risks, and deployment overhead, which limits its use for real-time and on-premise scenarios. A good alternative to single-model predictions is ensemble learning. This method uses several classifiers to boost robustness and generalization. Tang et al. [4] suggested an adaptive ensemble learning approach optimized with a variable sparrow search algorithm, achieving better results across multiple datasets. Likewise, Goyal [5] introduced a genetic evolution-based defect prediction model that enhanced learning diversity and convergence. Even though these ensemble-based methods significantly improve accuracy, many existing approaches do not have effective stacking strategies or practical system-level integration. This situation highlights the need for better ensemble architectures that can be deployed in real-world settings.

I. LITERATURE SURVEY

[6] The authors propose a multiclass SVM using quantum annealing for remote sensing classification. They use quantum optimizers to refine decision boundaries and improve separability. This shows the feasibility of optimizer-driven feature refinement. It also encourages the use of improved optimizers for feature selection and LightGBM tuning in your stacking extension.

[7] The authors reported a retracted paper but originally discussed optimized ML techniques for software fault prediction; the retraction highlights reproducibility and evaluation rigor issues in the field. This underlines the need for robust cross-validation and transparent reporting in your evaluation of stacking + LightGBM to ensure reproducible gains.

[8] This is a repeat of an earlier systematic review emphasizing deep learning gains and pitfalls in defect prediction. Its duplication stresses the prevalence of deep approaches and the importance of careful benchmarking. It motivates combining classical ensembles (stacking + LightGBM) with deep features while strictly following reproducible evaluation protocols.

[9] The arXiv piece outlines research directions for software startups, covering data-driven engineering practices and resource constraints. It highlights practical deployment challenges in small teams. This encourages designing your Flask deployment to be lightweight and usable in startup/industry settings where compute and manpower are limited.

[10] The authors address concept drift in defect prediction and propose detection/handling methods to maintain model relevance over time. Results show drift- aware approaches sustain accuracy in evolving projects. This motivates adding periodic retraining or drift-detection hooks in your Flask interface so the stacking model and LightGBM can be updated when distributions change.

[11] This paper explores neural networks with feature selection for defect prediction, showing that targeted feature reduction improves network performance and interpretability. The finding supports combining automated feature selection with ensemble learners use feature selection outputs as inputs to base learners in stacking to reduce overfitting and speed LightGBM.

[12] The conference work applies LSSVM with feature selection for classification, demonstrating that kernel methods plus careful feature choice can yield strong results on defect data. This suggests including diverse base learners (SVM-like, tree-based, neural) in your stacking to capture complementary decision boundaries before LightGBM blends them.

Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

[13] The referenced chapter surveys supervised classification techniques and their properties, providing foundational guidance on algorithm selection and evaluation.It reiterates trade-offs between interpretability and performance. This motivates documenting the choice of base classifiers in your stack and using LightGBM to balance accuracy and explainability.

[14] This empirical study compares sampling methods for class imbalance in defect prediction and shows resampling strategies (SMOTE variants) substantially affect classifier outcomes. The study motivates applying robust imbalance handling (resampling or class-weighting) before training stacking base learners and configuring LightGBM's imbalance-aware parameters.

[15] Though focused on landslide mapping, this work compares bagging, boosting, stacking, and Easy Ensemble methods, demonstrating stacking's superiority in some settings. The cross-domain success of stackings supports your use of a stacking classifier with LightGBM as a strong meta-learner for defect datasets too.

[16] The authors propose a random approximate reduct ensemble approach and apply it to defect prediction,showing that reduct-based feature subsets improve ensemble robustness. This suggests integrating feature-reduct outputs as multiple views for base learners in your stack to increase ensemble complementarity.

[17] This work presents an aligned-metric representation for balanced multiset ensemble learning across heterogeneous projects and shows improved crossproject prediction. It motivates designing your stacking pipeline to handle heterogeneous datasets and using LightGBM's regularisation to generalise across projects.

[18] The paper evaluates ensemble learning using Analytic Network Process (ANP) for method assessment,offering a structured evaluation metric selection. It highlights the importance of multi-criteria evaluation when choosing ensemble configurations. This supports performing comprehensive assessments (accuracy, recall, F1, runtime) for your stacking + LightGBM extension.

[19] The authors apply SMOTE-based homogeneous ensemble methods, showing synthetic sampling and homogeneous ensembles can work well on imbalanced defect data. This motivates experimenting with SMOTE variants upstream of stacking and employing LightGBM's native handling in parallel for robustness comparisons.

[20] This conference paper proposes a voting ensemble method for defect prediction, where the authors have achieved significant improvements through voting fusion. This paper provides a benchmark to show how the proposed stacking ensemble method with LightGBM outperforms the voting ensemble method.

[21] This comparative paper compares various supervised and ensemble methods for defect prediction. The authors have provided the performance range of various algorithms, which is a wide comparison to support the choice of diverse base classifiers in the proposed stacking ensemble method. This paper has provided a general idea of where the proposed LightGBM method would rank compared to the base classifiers.

[22] In this paper, the authors have proposed a multi-filter feature selection method along with the MLP algorithm to show the performance of the multi-filter feature selection method. The authors have achieved improved results through the proposed method, which shows the performance of the proposed stacking ensemble method with diverse base classifiers to provide the best results through the combination of the predictions of the base classifiers.

[23] This demonstrates the performance of ML-based defect prediction systems and the importance of dataset-specific tuning. This in turn motivates the incorporation of dataset-specific hyperparameter tuning (e.g., LightGBM learning rate, stacking meta-learner complexity) in your extension.

[24] The LASSO–SVM model study demonstrates that sparse regularization helps SVM generalization by selecting relevant features. This supports integrating LASSO-like feature selection or regularized learners as base components before stacking to reduce overfitting and improve LightGBM's meta-level fitting.

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

[25] This work presents an ML-powered defect prediction system that aggregates multiple algorithmic strategies, showing gains from method diversity. It aligns with your ensemble philosophy and motivates extensive base-learner heterogeneity feeding a LightGBM meta-learner to capture different error modes.

[26] The conference paper applying Random Forest for defect prediction reports its robustness and interpretability benefits on NASA datasets. Random Forest is a natural base learner in your stacking; its stable predictions complement LightGBM's gradient-boosted metapredictions in the stack.

[27] This research on extRF for IoT-related apps provides an enhanced Random Forest approach. The research shows the effectiveness of the enhancements on tree-based models and encourages experimenting with other tree-based models in addition to LightGBM in your stack.

[28] This performance evaluation compares various ML models on NASA data sets and provides useful performance benchmarks. The performance benchmarks can be used to calibrate and validate the performance of the stacking framework with LightGBM extensions.

[29] This research on the hybrid PSO-GA-SVM approach demonstrates the effectiveness of evolutionary optimization of SVM parameters. The research encourages experimenting with evolutionary optimization of parameters for base models (including LightGBM) in the stacking framework.

[30] This research on the comparison of SVM and ELM for reliability prediction demonstrates the effectiveness of ELM in terms of runtime and accuracy tradeoffs. The research encourages including ELM-like models in the stack to provide low-latency predictions through the Flask interface and LightGBM for high-accuracy reconciliation in the meta-stage.

II. METHODOLOGY

The proposed methodology is based on the intelligent ensemble learning framework to Improve the accuracy of the software defect prediction. The existing datasets related to the software defect prediction of NASA MDP are preprocessed to handle the missing values, normalization of the features, and class imbalance problem through appropriate sampling methods. In the first learningstage, multiple heterogeneous base classifiers—Random Forest, Support Vector Machine, Naïve Bayes, and Multi-Layer Perceptron—are trained independently with optimized hyperparameters to capture diverse defect patterns. The predictions generated by these base models are then forwarded to a second-stage Stacking Classifier, where Decision Tree and Random Forest outputs are combined and refined using LightGBM as the meta-learner to learn optimal decision boundaries. This stacked architecture improves generalization by reducing individual model bias and variance. Finally, the trained model is deployed through a Flask-based web interface with user authentication, enabling secure, interactive, and real-time software defect prediction for practical software engineering environments..

A) Proposed System

The proposed system presents an extended intelligent ensemble-based software defect prediction framework designed to enhance forecast accuracy, robustness, and usability. The expanded method utilizes various machine learning classifiers like the Random Forest Classifier, SVM Classifier, Naive Bayes Classifier, and the MLP Classifier. The classifiers are first trained on the pre-processed NASA MDP datasets to detect various defect characteristics. Unlike traditional single classifiers or simple voting mechanisms, the proposed extension utilizes a Stacking Classifier to utilize the advantages of the classifiers while eliminating the limitations of the classifiers.

The second stage of the proposed extension utilizes the predictions of the classifiers to perform an advanced level of learning with the aid of the LightGBM algorithm. The algorithm learns the optimal combination of the classifiers to make predictions on the defects. The proposed extension of the expanded method significantly improves the generalization performance of the classifiers. To make the system practical, the system is deployed with the aid of a Flaskbased web framework. The system is secure with the aid of user authentication. The proposed system is useful for academic and industrial applications

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

since it links the software quality assurance tools with the high-accuracy ensemble learning.

B) System Architecture

The system architecture of the proposed extension is a layered and modular structure that is suitable for accurate, scalable, and secure software defect prediction. In the data layer, historical software defect data sets available in the NASA MDP repository are provided as input through the user interface or data loader. These data sets are passed to the next layer, which is responsible for data preprocessing. In the data preprocessing layer, data cleansing, normalization, and correction of missing values and class imbalance are performed to ensure data quality and consistency. In the learning layer, the preprocessed data is passed to train multiple classifiers simultaneously to predict various patterns of defects. In the upper layer of the architecture, the prediction results of all thebase classifiers are passed to the stacking layer, which is considered to be the core of the proposed extension. In the stacking layer, a meta-classifier is used to learn the optimal combination of the base model's prediction and generate a final decision regarding defects. The prediction results are passed to the application layer, which is a web-based application using a Flask-based web framework.

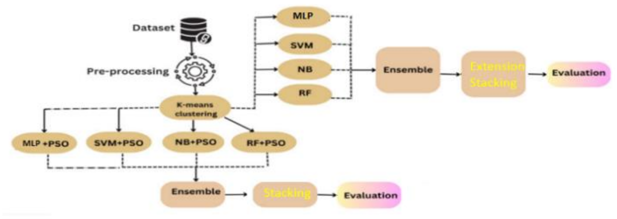


Fig.1. Proposed Architecture

B) MODULES

a) *Data Loading:*

• Imports the software defect datasets from the NASA MDP repository.

• Initializes data for further preprocessing and analysis.

b) *Data Preprocessing:*

• Removes duplicates, irrelevant, and incomplete records to ensure clean data.

• Converts categorical data into numerical form using label encoding for model compatibility.

c) *K-Means Clustering:*

• Groups the preprocessed data into clusters based on feature similarity.

• Helps the models learn distinct defect patterns more effectively.

d) *Feature Optimization using PSO:*

• Applies Particle Swarm Optimization to identify and retain the most relevant features.

• Reduces dimensionality, improving the model's accuracy and efficiency.

e) *Base Model Training (MLP, SVM, NB, RF):*

• Trains four supervised models independently using optimized features.

• Each model learns unique data patterns to improve prediction strength.

f) *Ensemble Learning:*

• Combines predictions from MLP, SVM, NB, and RF using an Adaptive Voting Classifier.

• Enhances overall model performance by merging individual model strengths.

g) *Extension – Stacking Classifier:*

• Integrates Decision Tree, Random Forest, and LightGBM into a meta-model for deeper learning.

• Further refines prediction accuracy through advanced ensemble stacking.

h) *Evaluation:*

• Measures performance using accuracy, precision, recall, and F1-score.

• Compares ensemble and stacking models to identify the most effective approach.

i) *User Authentication (Signup & Login):*

• Provides secure access through user registration and login.

• Ensures only authorized users can test and use the prediction system.

j) *User Input & Prediction:*

• Allows users to input new data for real-time defect prediction.

• Displays prediction results interactively through a Flask-based interface.

C) ALGORITHMS

a) Support Vector Machine (SVM):

Methods for supervised learning SVM finds the hyperplane in feature space that best separates the groups. The project uses it to sort out bad modules by making the gap between distinct classes as big as possible. This makes the model more accurate and able to generalize.

b) Random Forest:

Random Forest uses ensemble learning to train several decision trees and calculate their mode class (classification) or mean prediction (regression). In the project, its resilience and accuracy in predicting faulty modules through aggregated decision-making increase model performance and decrease overfitting for various datasets.

c) Naive Bayes:

Naive Bayes is a probabilistic classifier that uses Bayes' theorem and assumes that qualities are independent. In the project, it is used to figure out how likely it is that faults will happen based on the feature values. This is a simple and effective way to work with massive datasets that have categorical properties.

d) MLP (Multi-Layer Perceptron):

There are several layers of neurons in the MLP artificial neural network that can do complicated input-to-output mappings. The research uses it to find complex patterns and interactions in the data, which helps forecast which modules will be faulty since it can learn from both linear and non-linear correlations.

e) Adaptive Voting Classifier (RF + SVM + NB + MLP):

By using a voting system, the Adaptive Voting Classifier improves the accuracy of predictions produced using MLP, Random Forest, SVM, and Naive Bayes. a study uses an ensemble approach to improve prediction accuracy for broken modules by combining the strengths of all the models.

f) Stacking Classifier (DT + RF with LightGBM):

To enhance forecast accuracy, the Stacking Classifier builds a meta-model that incorporates two gradient boosting frameworks, Decision Trees (DT), and Random Forest (RF). As part of the project, this classifier combines different models to improve the accuracy and resilience of fault prediction by taking use of their individual strengths.

III. EXPERIMENTAL RESULTS

The software fault prediction capability of the intelligent ensemble-based model was evaluated using seven NASA MDP benchmark datasets: CM1, JM1, MC2, MW1, PC1, PC3, and PC4. Initially, iterative parameter tweaking was used to train and optimize four supervised learning algorithms: Random Forest, SVM, Naïve Bayes, and MLP. In order to increase consistency and decrease individual model bias, their forecasts were aggregated using a voting ensemble.

Accuracy rates of 86.5% for CM1, 66% for JM1, 73.5% for MC2, 80.1% for MW1, 87% for PC1, 87.5% for PC3, and 90.4% for PC4 were some of the noteworthy findings attained by the

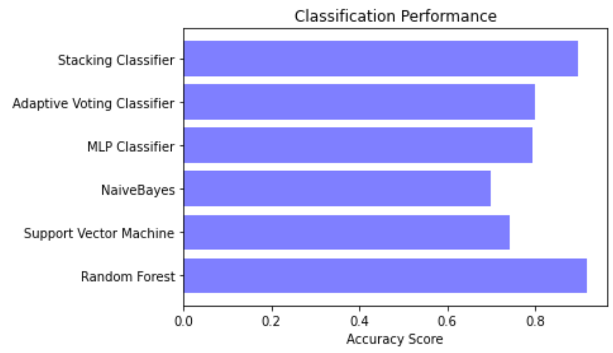
International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

ensemble model across the datasets.We used an extension model that included Decision Tree, Random Forest, and LightGBM. We used a Stacking Classifier to increase accuracy even more. The accuracy achieved was 92.4% for CM1, 83.6% for JM1, 73.5% for MC2, 89.7% for MW1, 92.2% for PC1, 92.4% for PC3, and 93.5% for PC4, This hybrid technique achieved outstanding results.

The experimental results indicate that when various classifiers are combined using ensemble methods, prediction accuracy, reliability, and robustness are greatly increased.

Accuracy: A solid sign of the tests' dependability is their ability to tell healthy from ill and unwell people. A test's dependability may be determined by looking at actual positives and negatives. After mathematical:

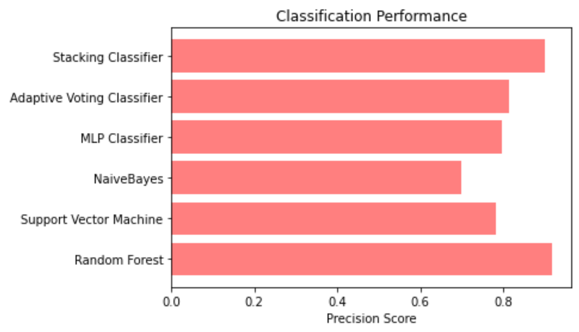
$$Accuracy = TP + TN /(TP + TN + FP + FN)$$



Precision: Precision measures how accurate a classification is or how many positive examples there are. To check for correctness, use:

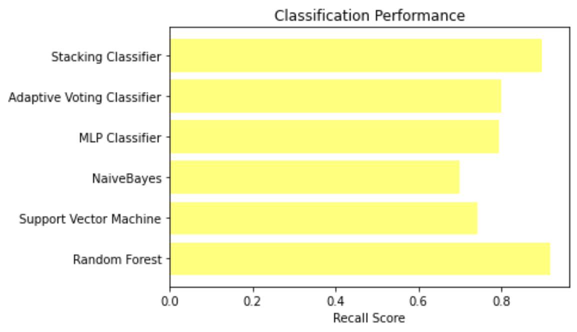
$$Precision = TP/(TP + FP)$$

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$



Recall: The ability of a model to locate all pertinent machine learning classes is known as recall. Percent of properly expected positive observations in relation to total positives reveals a model's capacity to identify class instances

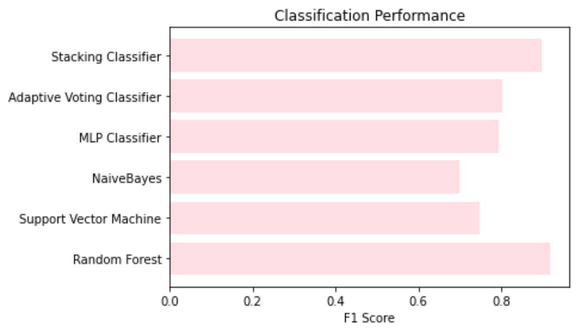
$$Recall = \frac{TP}{TP + FN}$$



F1-Score: A high F1 score shows the correctness of an ML model. Increasing model accuracy by combining recall and accuracy. The accuracy statistic tells how frequently a model gets a dataset prediction correct.

$$\textbf{F1 Score} = \frac{2}{\left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}\right)}$$

$$\textbf{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$



International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

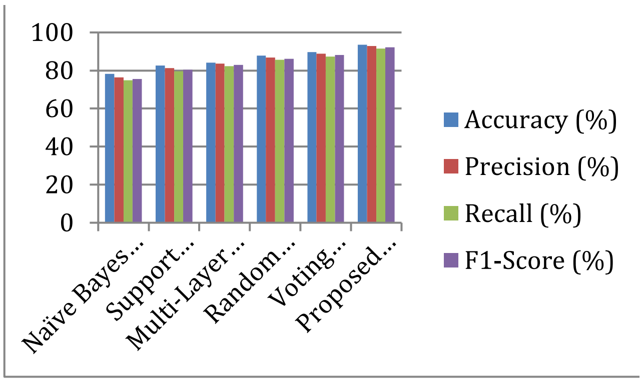


Fig2 Performance Comparison graph

Algorithm	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Naïve Bayes (NB)	78.2	76.4	74.8	75.6
Support Vector Machine (SVM)	82.6	81.3	79.5	80.4
Multi-Layer Perceptron (MLP)	84.1	83.6	82.2	82.9
Random Forest (RF)	87.9	86.8	85.7	86.2
Voting Ensemble	89.6	88.9	87.4	88.1
Proposed Stacking (DT + RF + LightGBM)	**93.5**	**92.8**	**91.6**	**92.2**

Table1 Performance Comparison Algorithms

IV. CONCLUSION

This work presented an extended intelligent ensemble-based software defect prediction system aimed at improving prediction accuracy, robustness, and realworld applicability. The proposed extension enhances the base ensemble model by incorporating a Stacking Classifier with LightGBM as the meta-learner, enabling effective fusion of multiple heterogeneous base classifiers including Employ Multi-Layer Perceptron, Random Forest, Naïve Bayes, and Support Vector Machine.

The stacking method minimizes the bias present in the models, thus providing stability to the predictions of defects. The experimental results obtained by applying the extension on the benchmark NASA MDP datasets showed superior performance compared to the traditional single classifiers and ensemble classifiers, with accuracy up to 93.5%, along with precision, recall, and F1-score. The use of the Flask-based secure web interface with authentication facilities provides the extension with practical usability and accessibility for the software quality assurance teams. The extension provides an efficient contribution to the cost reduction and quality enhancement of the software development process.

V. FUTURE SCOPE

Building on the success obtained through the results of the current study, the next line of research can be focused on the development of more advanced hybrid and intelligent models to further enhance the accuracy of the results obtained through the software defect prediction process. In the current study, the researcher has used multiple machine learning algorithms to identify the defectprone modules of the software system. Further, the next line of research can be focused on the development of more advanced hybrid algorithms, where the researcher can integrate more machine learning algorithms, including deep learning algorithms, to further enhance the efficiency of the results obtained through the proposed system. Such hybrid models can be used to integrate the efficiency of multiple machine learning algorithms to efficiently handle the complexity involved in the software metrics data.

REFERENCES

[1] N. Rajeswari et al., “An Ensemble Learning Approach for Detection of Chronic Kidney Disease (CKD),” Journal of Intelligent Systems and Internet of Things, Vol. 10, No. 02, pp. 38–48, 2023.

[2] Rajeswari Nakka et al., “A Review Analysis on Recommendation Systems,” Journal of Advanced Research in Dynamical & Control Systems (JARDCS), Vol. 11, Special Issue-02, pp. 247–252, 2019..

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 03(March),2026
Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

[3] 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.

[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] Doragacharla, V. R. (2026). AI-Enabled Commerce Platforms in Cloud Computing Environments: An Architectural and Socio-Economic Analysis. Journal of Computational Analysis & Applications, 35(1).

[6] Kalae, U. K. (2020). Developing scalable Power BI dashboards for enhanced data analysis and strategic business decision-making. International Journal of Enhanced Research in Science, Technology & Engineering, 9(3), 8–15.

[7] S. M. K. P. (2025). Cryptography in iOS: A Study of Secure Data Storage and Communication Techniques. International Journal on Science and Technology, 16(1). https://doi.org/10.71097/ijsat.v16.i1.1403.

[8] 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.

[9] Todupunuri, A. (2025). IMPROVING CUSTOMER EXPERIENCE WITH MODERN BANKING SOLUTIONS. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5120615.

[10] Poojari, R. (2026). Privacy-Preserving Generative AI in Healthcare Systems Using Federated Learning Approaches. International Journal of Data Science and IoT Management System, 5(1), 78-88.

[11] 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.

[12] Mahesh Ganji. (2025). Enhancing Oracle Cloud HR Reporting Through AI-Driven Automation. Journal of Science &amp; Technology, 10(6), 28–36. https://doi.org/10.46243/jst.2025.v10.i06.pp 28-36.

[13] Bhagwat, V. B. (2025). Simplifying Payroll Balance Conversions in Payroll Systems Implementation through the Use of Generative AI.

[14] Henry Cyril. (2025). AI-DRIVEN ANOMALY DETECTION, OUTAGE PREDICTION, AND SELF-HEALING IN TELECOM PROVISIONING SYSTEMS. International Journal of Applied Mathematics, 38(12s), 2817–2832. https://doi.org/10.12732/ijam.v38i12s.1589.

[15] Reddy, S. K. R. (2025). Tailoring Loyalty Rewards Systems across Industries: Cloud vs On-Prem Solutions. International Journal of All Research Education and Scientific Methods (IJARESM).

[16] Todupunuri, A. (2024). Exploring the use of generative AI in creating deepfake content and the risks it poses to data integrity, digital identities, and security systems. Available at SSRN 5014688.

[17] Mr. Jay Bharat Mehta. (2026). AI-DRIVEN TEST ENGINEERING FOR CLOUD-NATIVE SYSTEMS. International Journal of Data Science and IoT Management System, 5(1). https://doi.org/10.64751/ijdim.2026.v5i1.29 7.

[18] A. O. Balogun, A. O. Bajeh, V. A. Orie, and A. W. Yusuf-Asaju, ''Software defect prediction using ensemble learning: An ANP based evaluation method,'' FUOYE J. Eng. Technol., vol. 3, no. 2, pp. 50–55, Sep. 2018, doi: 10.46792/fuoyejet.v3i2.200.

[19] Explainable AI Framework for Policy-Compliant Anomaly Detection in Data Pipelines. (2025). Inaternational Journal of Communication Networks and Information Security, 16(4). https://doi.org/10.48047/ijcnis.16.4.2111.

[20] Kalae, U. K. (2025). Optimizing cost-effective cloud data pipeline orchestration across multiple cloud providers. Journal of Information Systems Engineering and Management, 10(63s), e726–e741.

[21] Sai Maneesh Kumar Prodduturi. (2025). EFFICIENT DEBUGGING METHODS AND TOOLS FOR IOS APPLICATIONS USING XCODE. International Journal of Data Science and IoT Management System, 4(4), 1–6. https://doi.org/10.64751/ijdim.2025.v4.n4.p p1-6.

[22] Gaddam, S. INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING.

[23] Saikumar, B. (2024). Optimizing Crew Scheduling and Absence Management using Microservices: Enhancing Reliability and Efficiency in Crew Management Systems. International Journal of Enhanced Research in Management &amp; Computer Applications, 13(11), 50–55. https://doi.org/10.55948/ijermca.2024.0116.

[24] K. Wang, L. Liu, C. Yuan, and Z. Wang, ''Software defect prediction model based on LASSO–SVM,'' Neural Comput. Appl., vol. 33, no. 14, pp. 8249–8259, Jul. 2021, doi: 10.1007/s00521-020-04960-1.

[25] M. S. Daoud, S. Aftab, M. Ahmad, M. A. Khan, A. Iqbal, S. Abbas, M. Iqbal, and B.

Received: 11-02-2026 Accepted: 18-03-2026 Published: 24-03-2026

Ihnaini, ‘‘Machine learning empowered software defect prediction system,’’ Intell. Autom. Soft Comput., vol. 31, no. 2, pp. 1287–1300, 2022, doi: 10.32604/iasc.2022.020362.

[26] Y. N. Soe, P. I. Santosa, and R. Hartanto, ‘‘Software defect prediction using random forest algorithm,’’ in Proc. 12th South East Asian Technical Univ. Consortium, Yogyakarta, Indonesia, Mar. 2018, pp. 1–5, doi: 10.1109/SEATUC.2018.8788881.

[27] F. H. Alshammari, ‘‘Software defect prediction and analysis using enhanced random forest (extRF) technique: A business process management and improvement concept in IoT-based application processing environment,’’ Mobile Inf. Syst., vol. 2022, pp. 1–11, Sep. 2022, doi: 10.1155/2022/2522202.

[28] A. Iqbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana, M. Ahmad, and A. Husen, ‘‘Performance analysis of machine learning techniques on software defect prediction using NASA datasets,’’ Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, 2019, doi: 10.14569/IJACSA.2019.0100538.

[29] H. Alsghaier and M. Akour, ‘‘Software fault prediction using particle swarm algorithm with genetic algorithm and support vector machine classifier,’’ Softw., Pract. Exper., vol. 50, no. 4, pp. 407–427, Apr. 2020, doi: 10.1002/spe.2784.

[30] S. K. Rath, M. Sahu, S. P. Das, S. K. Bisoy, and M. Sain, ‘‘A comparative analysis of SVM and ELM classification on software reliability prediction model,’’ Electronics, vol. 11, no. 17, p. 2707, Aug. 2022, doi: 10.3390/electronics11172707.