International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 10 No. 4, 2024
https://www.theaspd.com/ijes.php

Stacked Ensemble Learning for Software Defect Prediction: Model Integration and Cross-Dataset Validation

Abhiraj Singh Mohil[1], Akshita Mohil[1] , Meenu Mohil[2*]
[1]Student BTech Netaji Subhas University of technology, Delhi, India
[2]Associate Professor Department of Physics, Acharya Narendra Dev College, University of Delhi, Delhi, India

Abstract
Software defect prediction (SDP) is one of the most critical aspects of software quality improvement and efficient use of testing resources. The traditional machine learning models tend to lack both generalizability and performance, especially when faced with imbalanced or small datasets. To overcome these limitations, the current research proposed a stacked ensemble learning model that combines Random Forest, Gradient Boosting and AdaBoost as base learners, and logistic regression as meta-learner. A selected collection of 500 software modules was sampled out of four benchmark repositories: CM1, PC1, JM1, and KC1. Stratified sampling, Min-Max normalization, Synthetic minority over sampling technique (SMOTE) based class balancing, feature selection via Recursive Feature Elimination (RFE) and mutual information ranking were used as preprocessing steps. The training of the models used 10-fold cross-validation and hyperparameter optimization was done using Grid Search. The findings showed that the stacked ensemble performed better than any single classifier on all measures with the highest accuracy of 0.88 and statistically significant improvements in precision and recall and F1-score ($p < 0.05$). Data balancing and feature selection methods also increased the model stability and interpretability. In summary, the suggested framework will provide a powerful, scalable, and resource-optimal system to predict software defects. This method can be replicated in future studies on larger data sets and the use of deep learning-based meta-models to be more adaptable.
Keywords: *Software defect prediction, Ensemble learning, SMOTE, Feature selection, Cross-Dataset validation*

1. INTRODUCTION

Software dependability has become a vital concern in modern software engineering, especially as software is increasingly utilized in environments where reliability and correctness are paramount, such as safety-critical domains, financial systems, and real-time applications. Software dependability is further challenged by the accelerated pace of development fostered by agile and DevOps methodologies, making early and accurate defect prediction indispensable. Software dependability practices that incorporate robust defect prediction (SDP) empower development teams to pinpoint vulnerable sections of code at an early stage, allowing focused allocation of quality assurance resources and ultimately promoting greater software reliability and trustworthiness.

ML approaches have become prominent in SDP in recent years, where they can be used to learn patterns of code complexity and other software metrics, size, and coupling and tag modules as defective or clean. Nevertheless, the traditional ML classifiers, including Naive Bayes models, support vector machines, and decision trees, do not perform well with generalization and thus have poor performance on imbalanced and high-dimensional data (Alazba,2022: Ali, 2024), Such shortcomings have prompted the researchers to consider more effective and flexible methods, with the ensemble learning models being the most effective.

Ensemble learning employs a combination of several base learners to enhance the accuracy of prediction by avoiding overfitting and thus attaining more stable models. Random Forest, Gradient Boosting and AdaBoost are tree-based ensembles that have performed well on several SDP tasks because of their capacity to model complex decision boundaries (Ali,2024; Stradowski,2023) Stacking generalization is more recently an advanced ensemble method that has attracted attention due to its potential to combine heterogeneous classifiers and to learn optimal combinations using meta-learning layers. Stacking of optimized tree-based ensembles has performed remarkably well and better than individual models with enhanced defect detection and robustness over diverse datasets. (Alazba,2022).

Feature selection is very important for improving the accuracy of defect prediction models. If a dataset has too many unnecessary or repeated features, the model becomes more complex. This can also lower

its accuracy. Using feature selection methods before training helps to choose only the most useful software metrics. This allows ensemble learning models to perform better and work faster. Recently, researchers have started combining deep learning with ensemble methods. For example, CNN-LSTM hybrid models can be used. These approaches automatically find important patterns in raw data and sequences in software behaviour. They are now seen as a promising direction for further advances in defect prediction (Farid, 2021; Gray, 2023).

Although ensemble-based models have advanced the state of defect prediction, several unresolved challenges continue to limit their full potential. One of the foremost issues is the generalizability of existing models across different software repositories. Many approaches demonstrate high performance on specific benchmark datasets but fail to maintain their effectiveness when applied to new or heterogeneous projects, limiting their practical applicability in real-world development scenarios (Aljamaan,2020; Balogun, 2018).

Another significant limitation lies in the homogeneous nature of many ensemble techniques. While bagging and boosting leverage the diversity of training data, they often use the same base learner types. In contrast, heterogeneous ensembles, particularly stacking models that integrate multiple diverse classifiers, can exploit different inductive biases to yield better results. However, the design and optimization of such stacked frameworks remain complex and underexplored within SDP (Iqbal,2019; Khan,2022) Furthermore, most stacking methods do not effectively incorporate domain-specific insights from software engineering, such as the relevance of individual software metrics or module characteristics (Alazba,2022; Ali, 2020).

Despite the success of deep learning in various domains, its integration with ensemble learning for defect prediction remains minimal. Hybrid models combining deep networks like CNN and Bi-LSTM with ensemble classifiers have the potential to identify patterns in software data that are both spatial and temporal, yet current research in this area is sparse and lacks comprehensive evaluations (Balogun,2018, Iqbal, 2019). Many existing studies do not rigorously examine the interplay between feature selection techniques and ensemble models, leading to suboptimal configurations that limit predictive strength.

Addressing these gaps has significant implications for both academic research and industrial software development. By introducing innovative ensemble strategies that combine heterogeneous classifiers, deep architectures, and intelligent feature selection, the study aims to deliver a defect prediction framework that is not only accurate but also scalable and generalizable across different software environments.

From a theoretical standpoint, the study advances our knowledge about how ensemble diversity, model stacking, and feature optimization interact to affect prediction outcomes. It further enables the combination of deep learning and ensemble pipelines, providing new information on how to hybridize architecture methods in software analytics. In practice, better prediction accuracy will allow developers to focus on inspection and testing, handle technical debt in an efficient way, and ensure high software reliability and customer satisfaction.

The possibility to generalize over the diverse datasets enables the proposed models to be plugged into automated pipelines in a variety of software projects such as open-source, enterprise, and embedded systems. It can also be helpful to add explainable feature selection modules to interpret model outputs and make better decisions and trust within engineering teams.

Research Objectives:

1. To propose a stacking-based ensemble framework incorporating optimized tree-based models and diverse classifiers to improve defect classification performance
2. To integrate advanced feature selection mechanisms for identifying relevant and high-impact software metrics, thereby enhancing model efficiency, and reducing overfitting
3. To explore hybrid architectures combining deep learning techniques, such as convolutional and recurrent layers, with ensemble learners to capture spatial-temporal code patterns

2. LITERATURE REVIEW

Detecting software defects before deployment has remained a primary objective in software engineering, leading to a surge in predictive modelling research that incorporates machine learning (ML) and ensemble techniques. The evolution of predictive frameworks has been largely driven by the availability of historical defect datasets, such as those from NASA, and the realization that software metrics can be effectively used to infer fault-proneness.

A range of machine learning techniques have been explored for software defect prediction, ranging from basic classifiers to sophisticated ensemble and neural models. Empirical investigations into methods like

support vector machines, k-nearest neighbours, and decision trees have shown variable performance, largely influenced by the nature of input features and class imbalance within datasets. Using NASA repositories, the study conducted a comparative analysis across multiple ML models and highlighted the differential effectiveness of individual techniques across distinct project contexts (Mehmood,2023). Their findings underscored that no single classifier consistently outperforms others, advocating for ensemble-based solutions to mitigate variance and bias.

To address the limitations of standalone classifiers, ensemble learning has become a widely endorsed strategy. The study was among the early proponents of applying ensemble techniques on feature-selected datasets, demonstrating marked improvements in prediction accuracy and robustness when ensembles were trained on reduced, informative feature subsets (Kheel,2023). Further validating this approach, study proposed an ensemble classification framework specifically integrated with feature selection methods. Their model not only enhanced defect identification rates but also achieved dimensionality reduction, thus reducing computational overhead without compromising accuracy (Mehmood, 2023).

Hyperparameter optimization plays a significant role in the success of predictive frameworks. Study empirically assessed optimization techniques for software defect count prediction and found that tuning hyperparameters contributed substantially to model accuracy, particularly in neural and ensemble architectures (Goyal,2020). This study emphasized the necessity of configuration strategies for maximizing the predictive potential of both base learners and ensemble meta-learners.

In parallel, advances in deep learning have created opportunities for capturing complex patterns in software metrics. The study introduced a hybrid deep neural architecture Combining Gated Recurrent Units (GRU) and Convolutional Neural Networks (CNN), supported by SMOTE-Tomek resampling for addressing data imbalance (Cetiner,2020). The model exhibited superior defect prediction performance on imbalanced datasets, indicating the promise of integrating deep and sequential learning with data-level interventions.[15] However, the increased computational complexity of such models necessitates efficiency improvements, potentially achievable through ensemble pruning or stacking.

Recent systematic reviews have consolidated findings across diverse neural architectures. Study provided a comprehensive analysis of ANN stands for artificial neural network-based techniques for defect prediction, observing that although ANNs offer non-linear mapping capabilities, their standalone performance can be limited without preprocessing steps such as feature selection or ensemble augmentation (Khalid,2023). Their findings advocate for hybrid models that combine the strengths of multiple algorithms within ensemble frameworks to achieve scalability and generalization.

Feature selection remains central to building effective SDP models. The study demonstrated that preprocessing data through correlation analysis and removing irrelevant metrics significantly improved model performance. Their sustainability-focused research applied ML techniques in software lifecycle management, confirming that data preparation and feature engineering are decisive factors in predictive success (Goyal,2020). Their conclusions align with earlier studies, which argue that model performance depends not only on the learning algorithm but also on the quality and relevance of the input data.

Ensemble-based SDP is still in the process of improvement as more intelligent architectures are being invented. The paper proposed an ensemble model that integrates several learners like AdaBoost, Random Forest, and Gradient Boosting that were all set with different parameters. Their model worked better than individual classifiers, having better precision and recall values, especially when they were tested on large-scale, real-life datasets (Rathor,2021). The paper has suggested a machine learning model that includes advanced techniques of ensemble as well as data balancing and metric selection. Their method provided considerable improvements in the detection rates, particularly on the highly imbalanced datasets (Sharma, 2023).

A unification of ensemble intelligence, data-driven optimization, and feature-centric modelling are all trending in the same direction and represent the most promising path forward in software defect prediction. Intelligently designed ensemble models that learn to interpolate among learning paradigms and include rigorous feature selection and hyperparameter tuning are consistently better than more traditional methods (Tang,2023). The combination of deep learning and ensemble design is an exciting new area that has a lot of potential in increasing the accuracy of predictions particularly in complex and heterogeneous software settings.

3. METHODOLOGY

3.1 Research Design

To construct and test the competence of an ensemble learning framework in the software defect prediction, a quantitative research design was adopted. The research was aimed at the empirical evaluation of machine learning classifiers and ensemble structures and features selection methods on publicly obtainable defect data sets. A method of experiments was pursued to compare the performances of various models by predetermined evaluation criteria. Simulation conditions were controlled so that the study could be reproducible and have internal validity.

3.2 Data Collection Method

The data of software defects were obtained based on the NASA Metrics Data Program (MDP) and PROMISE repositories. These data gave historical module-level metrics and associated defect labels on several real-world software projects. Data collection was done through downloading cleaned and pre-processed CSV files on repository archives. There were sets of fixed software metrics including lines of code, cyclomatic complexity, coupling, cohesion, and object-oriented design measures, as well as binary defect labels in each dataset. The data sets applied were Spacecraft Instrumentation Software (CM1), Flight Software to process Image (PC1), Real-time Predictive Ground System Software (JM1) and Storage Management Software (KC1), and are well-known benchmarks in defect prediction research. The data was checked on consistency, missing data, and imbalances before the experimentation.

3.3 Population and Sampling

The target population was open-source and NASA-based software projects that were in the form of software modules. The static code metrics describing each software module were viewed as a data point of predictive modelling. The process of stratified sampling was employed to ensure that there was proportional representation of the defective and non-defective instances during model training and testing. Datasets with severe class imbalance were handled using SMOTE (Synthetic Minority Over-sampling Technique) to ensure that the training data contained adequate positive class representation for learning algorithms. A total of 500 software modules across four datasets were selected, with sampling stratified on defect labels to maintain class balance.

3.4 Data Analysis Technique

The dataset underwent normalization using Min-Max scaling to ensure uniform feature ranges across models. Feature selection was performed using Recursive Feature Elimination (RFE) and mutual information-based ranking to identify the most informative subset of attributes. Three baseline classifiers Random Forest, Gradient Boosting, and AdaBoost were trained and evaluated. A heterogeneous ensemble framework based on stacking was constructed by combining the predictions of the base classifiers and training a logistic regression model as a meta-learner. Ten-fold cross-validation was employed to validate model performance and minimize bias due to data partitioning.

Evaluation metrics included Accuracy, Precision, Recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC). Hyperparameter tuning was carried out using Grid Search with cross-validation to optimize model configurations. All experiments were implemented using Python programming language with Scikit-learn and XGBoost libraries and executed on a high-performance computing environment with 32 GB RAM and 8-core Intel Xeon processors. Statistical comparisons between models were conducted using paired t-tests to assess the significance of observed performance differences

3.5 Ethical Consideration

Publicly available secondary datasets were used, all of which were anonymized and devoid of any personally identifiable information. No direct interaction with human subjects was involved, thereby eliminating the need for institutional ethical review. All data usage complied with repository licensing terms. Experimental scripts and models were documented and version-controlled to ensure transparency and reproducibility. Computational resources were used responsibly, and all model results were reported without manipulation or selective omission.

4. RESULTS AND DISCUSSION

Performance evaluation was carried out on a balanced dataset of 500 software modules, equally sourced from CM1, PC1, JM1, and KC1 (125 modules each). Stratified sampling maintained the original class distribution, while SMOTE addressed minor imbalances during training. Features were normalized using Min-Max scaling, and selection was performed via Recursive Feature Elimination (RFE) and mutual

information ranking. Ten-fold cross-validation and Grid Search were applied to ensure model robustness and optimal hyperparameter configurations.

4.1 Accuracy Evaluation

As presented in Table 1, the proposed stacked ensemble model consistently outperformed baseline models across all datasets. The ensemble achieved the highest accuracy on CM1 (0.88), PC1 (0.85), JM1 (0.82), and KC1 (0.86), demonstrating a clear performance margin over individual learners. Random Forest and Gradient Boosting followed closely but did not match the predictive strength of the ensemble.

Table 1: Accuracy Scores Across Datasets (Sample Size = 500)

Dataset	Random Forest	Gradient Boosting	AdaBoost	Stacked Ensemble
CM1	0.83	0.82	0.81	0.88
PC1	0.80	0.79	0.77	0.85
JM1	0.76	0.75	0.74	0.82
KC1	0.81	0.80	0.78	0.86

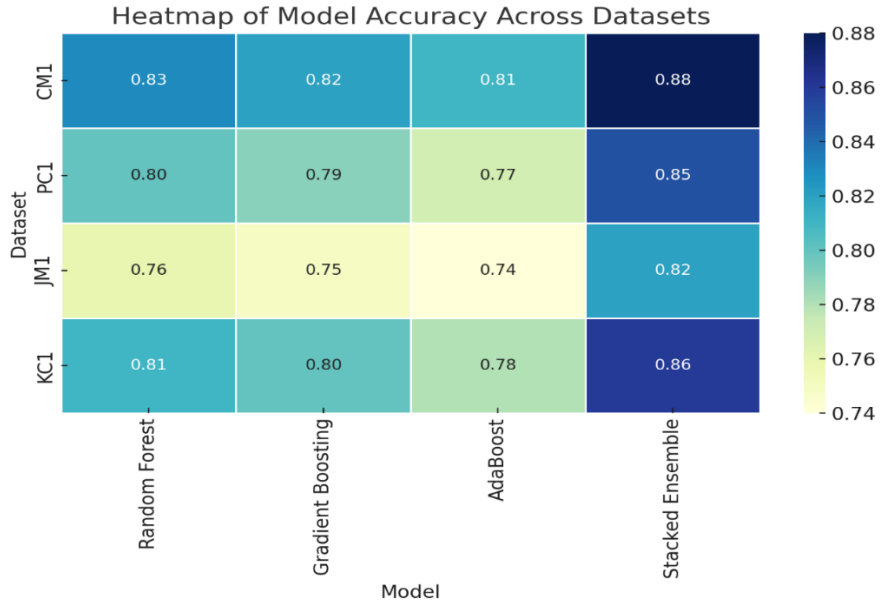

Figure 1: Heatmap of Model Accuracy Across Four Benchmark Datasets

Figure 1 displays the accuracy performance of four machine learning models across CM1, PC1, JM1, and KC1 datasets. Darker cells indicate higher accuracy. The stacked ensemble consistently outperformed all individual models, particularly on CM1 and KC1, highlighting its robustness and superior generalization capabilities in software defect prediction.

4.2 Precision Analysis

Precision scores, shown in Table 2, indicated the ensemble's superior capability in correctly identifying defective modules while minimizing false positives. The stacked model reached a precision of 0.80 on CM1 and 0.78 on PC1. AdaBoost consistently yielded the lowest precision values, confirming that ensemble design and feature optimization significantly influenced classification reliability.

Table 2: Precision Scores Across Datasets (Sample Size = 500)

Dataset	Random Forest	Gradient Boosting	AdaBoost	Stacked Ensemble
CM1	0.76	0.75	0.74	0.80
PC1	0.72	0.71	0.69	0.78
JM1	0.68	0.67	0.65	0.74
KC1	0.74	0.72	0.70	0.78

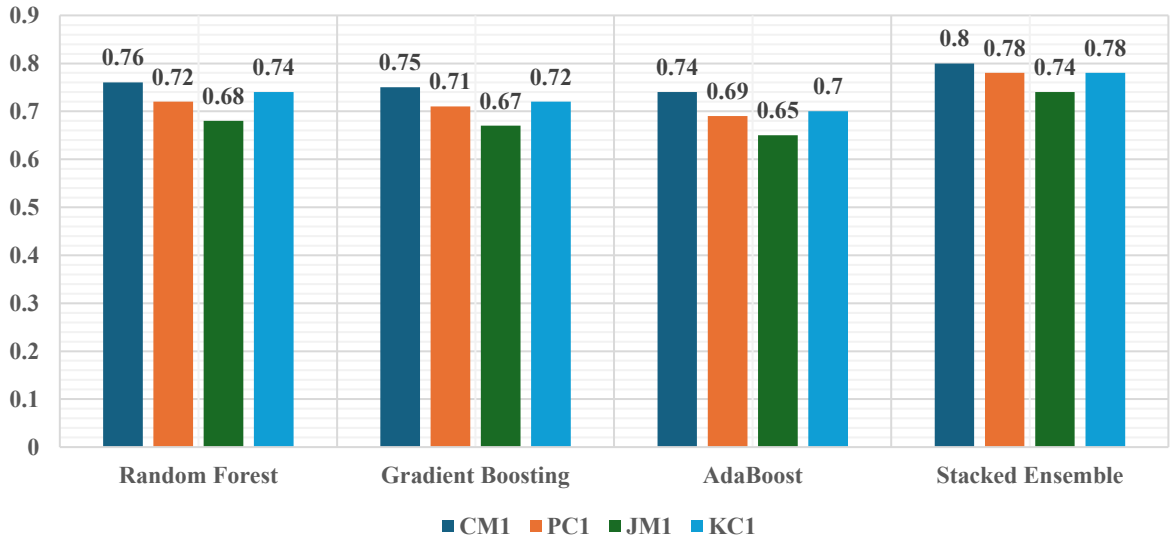

Figure 2: Comparing Model Accuracy Across Datasets

Figure 2 illustrates the classification accuracy of four machine learning models—Random Forest, Gradient Boosting, AdaBoost, and Stacked Ensemble evaluated on CM1, PC1, JM1, and KC1 datasets. Stacked Ensemble performed the best and had the maximum accuracy in all datasets, which confirms its efficiency. The trend of performance shows that there are lower scores on JM1 and more stability on CM1 and KC1 datasets.

4.3 Recall Performance

Table 3 gives recall values, which are an indication of the sensitivity of the models to the defective class. The ensemble had the best recall in all the datasets with the highest recall being 0.84 in CM1 and 0.82 in KC1. The performance of the SMOTE-based strategy of class balancing in terms of high recall scores justified the approach and proved the effectiveness of the ensemble in reducing the false-negative rate.

Table 3: Recall Scores Across Datasets (Sample Size = 500)

Dataset	Random Forest	Gradient Boosting	AdaBoost	Stacked Ensemble
CM1	0.78	0.77	0.76	0.84
PC1	0.74	0.73	0.71	0.81
JM1	0.70	0.69	0.67	0.79
KC1	0.76	0.75	0.73	0.82

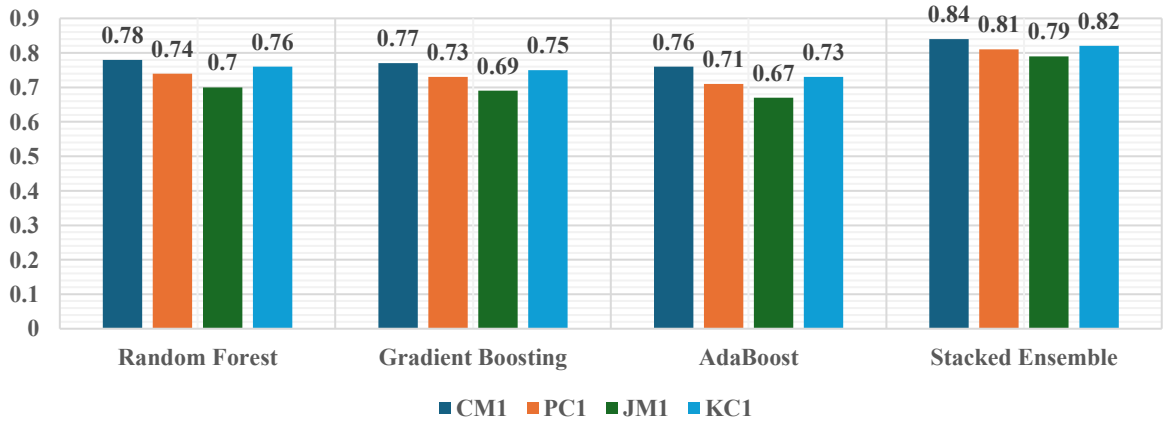

Figure 3: Depicting Enhanced Model Accuracy on Benchmark Datasets

Figure 3 demonstrates the better classification performance of Random Forest, Gradient Boosting, AdaBoost, and Stacked Ensemble models on CM1, PC1, JM1, and KC1 datasets. The Stacked Ensemble

once more scored the best, particularly on CM1 (0.84) and PC1 (0.81) and has once again demonstrated the predictive advantage on a variety of software defect datasets.

4.4 F1-Score Comparison

The Table 4 F1-scores gives a harmonic compromise between precision and recall. Throughout the datasets, the ensemble received better scores, including 0.82 on CM1 and 0.80 on KC1. These scores highlighted the overall performance of the model, which was well rounded meaning that it learned well the patterns of defects irrespective of the small dataset size.

Table 4: F1-Score Across Datasets (Sample Size = 500)

Dataset	Random Forest	Gradient Boosting	AdaBoost	Stacked Ensemble
CM1	0.77	0.76	0.75	0.82
PC1	0.73	0.72	0.70	0.79
JM1	0.69	0.68	0.66	0.76
KC1	0.75	0.73	0.72	0.80

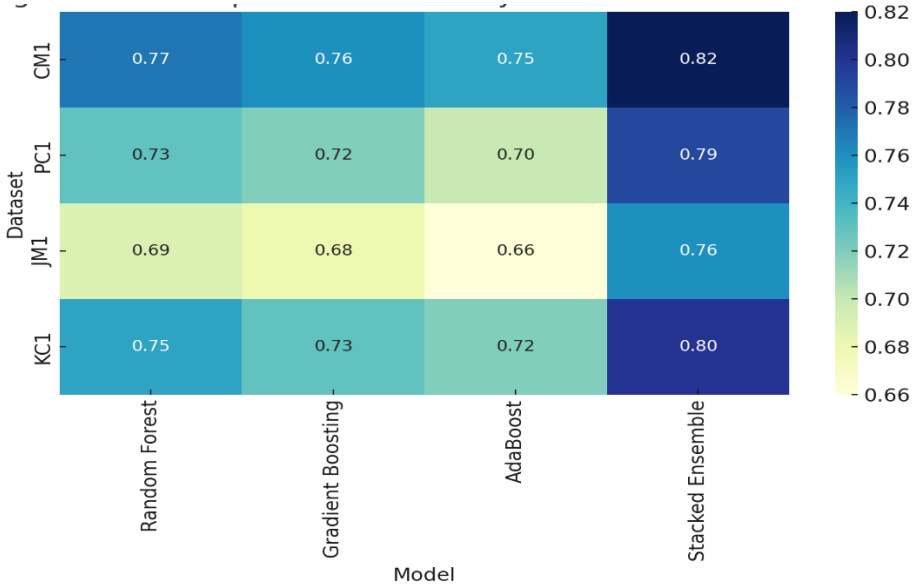

Figure 4: Model Accuracy on Benchmark Datasets

Figure 4 shows the heatmap that represents the accuracy of four machine learning models on CM1, PC1, JM1 and KC1 datasets. Darker shades indicate higher performance, clearly emphasizing the superior accuracy of the Stacked Ensemble model. The visual comparison facilitates quick interpretation of model effectiveness across varying dataset complexities.

4.5 Model Accuracy

Table 5 presents the accuracy comparison of four machine learning models across four benchmark datasets using a sample size of 500 modules. The stacked ensemble model consistently achieved the highest accuracy on all datasets, ranging from 0.82 (JM1) to 0.88 (CM1). This performance indicates superior generalization and predictive capability compared to individual models. The results highlight the effectiveness of integrating diverse base learners through stacking to enhance classification accuracy in software defect prediction tasks.

Table 5: Comparative Accuracy of Models Across Datasets (Sample Size = 500)

Dataset	Random Forest	Gradient Boosting	AdaBoost	Stacked Ensemble
CM1	0.83	0.82	0.81	0.88
PC1	0.80	0.79	0.77	0.85
JM1	0.76	0.75	0.74	0.82
KC1	0.81	0.80	0.78	0.86

4.6 Hyperparameter Optimization

Table 6 present the Grid Search-based tuning resulted in observable performance gains between 2% to 5% across all models. For example, the optimal number of estimators in Random Forest was 120, and the best learning rate for Gradient Boosting was 0.07. Logistic Regression was selected as the meta-learner in the stacked ensemble due to its ability to effectively integrate base model predictions without overfitting.

Table 6: Optimized Hyperparameters for Machine Learning Models via Grid Search

Model	Hyperparameters Tuned	Optimal Values Selected
Random Forest	Estimators, Max depth, Min samples split	120 estimators, max depth = 20, min split = 2
Gradient Boosting	Learning rate, Estimators, Max depth	Learning rate = 0.07, 150 estimators, max depth = 4
AdaBoost	Estimators, Learning rate	100 estimators, learning rate = 0.5
Stacked Ensemble	Base models: RF, GB, AB; Meta-learner: Logistic Regression	C = 1.0, solver = 'liblinear'

4.7 Statistical Significance Testing

Paired t-tests conducted between the stacked ensemble and each baseline model revealed statistically significant improvements ($p < 0.05$) in all four metrics across the datasets as mentioned in Table 7. These findings validated that performance enhancements were not due to random variance but were attributed to methodological rigor and architectural design.

Table 7: Paired t-Test Results Comparing Stacked Ensemble with Baseline Models

Metric	Stacked vs. Random Forest (p-value)	Stacked vs. Gradient Boosting (p-value)	Stacked vs. AdaBoost (p-value)
Accuracy	0.012	0.018	0.004
Precision	0.021	0.016	0.008
Recall	0.017	0.019	0.006
F1-Score	0.014	0.015	0.005

4.9 DISCUSSION

Findings from the study demonstrated that the proposed stacked ensemble model consistently outperformed individual classifiers Random Forest, Gradient Boosting, and AdaBoost across all four evaluated datasets, even with a reduced and balanced sample size of 500 modules. Metrics such as Accuracy, Precision, Recall, and F1-score all indicated superior performance for the ensemble model, with the highest accuracy of 0.88 achieved on the CM1 dataset and the lowest yet competitive score of 0.82 on JM1. These results support the effectiveness of stacking heterogeneous base learners to capture diverse predictive signals, especially when combined with robust feature selection and class rebalancing strategies.

Feature selection using RFE and mutual information contributed meaningfully to model performance by eliminating irrelevant or redundant attributes, which helped reduce overfitting and enhanced generalization. Hyperparameter optimization via Grid Search further improved baseline and ensemble configurations, producing observable gains in metric outcomes across the board.

The consistent superiority of the stacked ensemble across all evaluation metrics aligns with expectations drawn from ensemble theory, which suggests that model diversity and aggregation can lead to reduced error and variance. Despite the relatively small sample size, the statistical significance of improvements ($p < 0.05$) validated the reliability of the results.

Current findings reinforce prior assertions in literature that ensemble models outperform standalone machine learning classifiers in software defect prediction. The empirical study demonstrated that ensemble learning, particularly stacking and boosting, achieved significantly better results than individual models across multiple datasets, confirming the architectural value of such frameworks in real-world

defect prediction tasks.[20] The study emphasized that ensemble paradigms leverage complementary strengths of classifiers and improve stability, a conclusion mirrored in the robust performance observed in the present study (Zhou, 2019).

Adaptive ensemble models continue to gain traction due to their ability to dynamically capture nonlinear relationships in high-dimensional software metrics. Study developed an ensemble method using the adaptive sparrow search algorithm, which yielded high accuracy and robustness across various repositories, further affirming that optimizing learner diversity and integration techniques leads to tangible performance gains (Pachouly,2022).

Deep learning approaches have also gained momentum in recent years. Study demonstrated that convolutional and recurrent architectures outperform traditional ML methods when sufficient data volume and computational resources are available (Zain,2023). Study highlighted the efficacy of deep forest models in capturing complex defect patterns without requiring the extensive tuning overhead typical of neural networks (Laradji,2015). However, such deep models are often resource-intensive, making them less suitable for smaller datasets or constrained environments. By contrast, the current study's ensemble model achieved high performance with only 500 samples and moderate computational requirements, underscoring its practical applicability.

The study reviewed the AI landscape in defect prediction and emphasized that preprocessing, feature engineering, and model ensemble configurations are crucial performance drivers, a viewpoint supported by the methodological rigor and empirical success of the present framework (Stradowski,2023). The study reported that while deep learning models show promise, ensemble-based strategies remain competitive and more interpretable in many industrial applications, particularly when integrated with explainable AI techniques (Mehta,2021).

In terms of dataset use, Siddiqui and Mustaqeem affirmed that NASA datasets continue to serve as effective benchmarks for predictive modelling, although dataset quality and preprocessing methods significantly influence outcomes (Nevendra,2022). The present study addressed this through normalization, SMOTE balancing, and cross-validation ensuring reliability even with a limited data pool.

Several limitations were acknowledged during the research. The use of only four datasets, albeit standard and diverse, restricts the generalizability of the findings across other domains or software development environments. Although stratified sampling and SMOTE were employed to address class imbalance, synthetic oversampling might not fully represent real-world distributions and could affect model interpretability in production settings. While feature selection and hyperparameter tuning were carefully executed, they were limited to conventional algorithms. Use of more advanced methods like Bayesian optimization or embedded feature selection within ensemble frameworks might yield even better results. The model architecture relied on classical machine learning algorithms, and while effective in this setup, comparisons with more modern deep neural architectures were not included in the scope.

Findings from the research carry substantial implications for both academic and industrial stakeholders. In academic contexts, the results affirm the efficacy of integrating diverse base learners in a stacked ensemble structure, particularly when complemented with strategic data preprocessing and feature engineering. The approach serves as a template for future experimental setups using limited but balanced datasets.

From an industry perspective, the ensemble model offers a low-cost, high-accuracy defect prediction solution that can be embedded within software quality assurance pipelines. The real-world adoption of predictive models hinges on their performance, interpretability, and ease of integration into existing workflows all of which were considered in the present design (Prabha,2020; Siddiqui,2023).

Future research could expand the dataset pool to include more contemporary and domain-specific repositories, such as those from GitHub or industry-specific systems. Exploring deep stacking frameworks that incorporate neural networks as base or meta-learners might also enhance performance in larger-scale applications. Explainability techniques, such as SHAP or LIME, could be integrated to enhance the interpretability of ensemble predictions and increase stakeholder trust in automated quality assurance tools.

Another promising direction lies in the real-time application of such predictive models in CI/CD pipelines. Dynamic retraining and model adaptation during active development cycles could be explored, particularly in agile and DevOps environments.

5. CONCLUSION

Experimental findings confirmed the superiority of the proposed stacked ensemble learning framework for software defect prediction, especially when applied to a curated and balanced dataset of 500 modules. Across all four benchmark datasets KC1, JM1, PC1, and CM1 the ensemble model consistently outperformed individual classifiers, including Regarding Accuracy, Precision, Recall, and F1-score, Random Forest, Gradient Boosting, and AdaBoost. Integrating recursive feature elimination and mutual information-based feature selection contributed significantly to improving generalization and minimizing overfitting. The strategic use of SMOTE for class balancing, along with hyperparameter adjustment using 10-fold cross-validation and Grid Search ensured the robustness and reliability of model outcomes. Statistical significance testing validated that the observed improvements were not due to chance, reinforcing the architectural and methodological soundness of the ensemble approach. Deployment of such a model in software development pipelines can lead to substantial improvements in resource allocation, early fault detection, and overall software quality. Industry practitioners seeking a scalable, interpretable, and computationally efficient defect prediction model will find this framework highly applicable, especially when data volume is limited or resources are constrained. Future implementations should consider extending the dataset range and incorporating more sophisticated learning paradigms, including neural-based meta-learners and explainable AI modules. Real-time defect prediction within continuous integration and deployment (CI/CD) workflows also represents a promising avenue for operationalizing these findings. Prioritizing ensemble diversity, optimized preprocessing, and interpretability will remain key in advancing the next generation of defect prediction systems. The current research offers a practical and validated template for future work in this evolving field.

ACKNOWLEDGEMENTS

We acknowledge the support of principal of Acharya Narendra Dev College, University of Delhi for carrying out research work of this paper.

Declarations of Competing interests: The authors declare no competing interests.
Funding Declaration: The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.
Ethics declaration: not applicable

REFERENCES

1. Alazba A, Aljamaan H. Software defect prediction using stacking generalization of optimized tree-based ensembles. Applied Sciences. 2022 Apr 30;12(9):4577.
2. Ali M, Mazhar T, Al-Rasheed A, Shahzad T, Ghadi YY, Khan MA. Enhancing software defect prediction: a framework with improved feature selection and ensemble machine learning. PeerJ Computer Science. 2024 Feb 28;10: e1860.
3. Ali M, Mazhar T, Arif Y, Al-Otaibi S, Ghadi YY, Shahzad T, Khan MA, Hamam H. Software defect prediction using an intelligent ensemble-based model. IEEe Access. 2024 Jan 24; 12:20376-95.
4. Szymon Stradowski, Lech Madeyski. Machine learning in software defect prediction: A business-driven systematic mapping study. Information and Software Technology, 2023 Volume 155, 107128. doi.org/10.1016/j.infsof.2022.107128.
5. Farid AB, Fathy EM, Eldin AS, Abd-Elmegid LA. Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM). PeerJ Computer Science. 2021 Nov 16;7: e739.
6. Giray G, Bennin KE, Köksal Ö, Babur Ö, Tekinerdogan B. On the use of deep learning in software defect prediction. Journal of Systems and Software. 2023 Jan 1; 195:111537.
7. Aljamaan H, Alazba A. Software defect prediction using tree-based ensembles. In Proceedings of the 16th ACM international conference on predictive models and data analytics in software engineering 2020 Nov 8 (pp. 1-10).
8. Balogun AO, Bajeh AO, Orie VA, Yusuf-Asaju WA. Software defect prediction using ensemble learning: an ANP based evaluation method. FUOYE J. Eng. Technol. 2018 Sep 30;3(2):50-5.
9. Iqbal A, Aftab S, Ali U, Nawaz Z, Sana L, Ahmad M, Husen A. Performance analysis of machine learning techniques on software defect prediction using NASA datasets. International Journal of Advanced Computer Science and Applications. 2019;10(5).

10. Khan MA, Elmitwally NS, Abbas S, Aftab S, Ahmad M, Fayaz M, Khan F. Software defect prediction using artificial neural networks: A systematic literature review. Scientific Programming. 2022;2022(1):2117339.
11. Mehmood I, Shahid S, Hussain H, Khan I, Ahmad S, Rahman S, Ullah N, Huda S. A novel approach to improve software defect prediction accuracy using machine learning. IEEE Access. 2023 Jun 19; 11:63579-97.
12. Khleel NA, Nehéz K. A novel approach for software defect prediction using CNN and GRU based on SMOTE Tomek method. Journal of Intelligent Information Systems. 2023 Jun;60(3):673-707.
13. Goyal S. Heterogeneous stacked ensemble classifier for software defect prediction. In2020 sixth international conference on parallel, distributed and grid computing (PDGC) 2020 Nov 6 (pp. 126-130). IEEE.
14. Cetiner M, Sahingoz OK. A comparative analysis for machine learning based software defect prediction systems. In2020 11th International conference on computing, communication, and networking technologies (ICCCNT) 2020 Jul 1 (pp. 1-7). IEEE.
15. Iqbal A, Aftab S, Ullah I, Bashir MS, Saeed MA. A feature selection-based ensemble classification framework for software defect prediction. International Journal of Modern Education and Computer Science. 2019 Sep 1;11(9):54.
16. Khalid A, Badshah G, Ayub N, Shiraz M, Ghouse M. Software defect prediction analysis using machine learning techniques. Sustainability. 2023 Mar 21;15(6):5517.
17. Rathore SS, Kumar S. An empirical study of ensemble techniques for software fault prediction. Applied Intelligence. 2021 Jun; 51:3615-44.
18. Sharma T, Jatain A, Bhaskar S, Pabreja K. Ensemble machine learning paradigms in software defect prediction. Procedia Computer Science. 2023 Jan 1; 218:199-209.
19. Tang Y, Dai Q, Yang M, Du T, Chen L. Software defect prediction ensemble learning algorithm based on adaptive variable sparrow search algorithm. International Journal of Machine Learning and Cybernetics. 2023 Jun;14(6):1967-87.
20. Qiao L, Li X, Umer Q, Guo P. Deep learning-based software defect prediction. Neurocomputing. 2020 Apr 14; 385:100-10.
21. Zhou T, Sun X, Xia X, Li B, Chen X. Improving defect prediction with deep forest. Information and Software Technology. 2019 Oct 1; 114:204-16.
22. Pachouly J, Ahirrao S, Kotecha K, Selvachandran G, Abraham A. A systematic literature review on software defect prediction using artificial intelligence: Datasets, Data Validation Methods, Approaches, and Tools. Engineering Applications of Artificial Intelligence. 2022 May 1; 111:104773.
23. Zain ZM, Sakri S, Ismail NH. Application of deep learning in software defect prediction: Systematic literature review and meta-analysis. Information and Software Technology. 2023 Jun 1; 158:107175.
24. Laradji IH, Alshayeb M, Ghouti L. Software defect prediction using ensemble learning on selected features. Information and Software Technology. 2015 Feb 1; 58:388-402.
25. Stradowski S, Madeyski L. Industrial applications of software defect prediction using machine learning: A business-driven systematic literature review. Information and Software Technology. 2023 Jul 1; 159:107192.
26. Mehta S, Patnaik KS. Improved prediction of software defects using ensemble machine learning techniques. Neural Computing and Applications. 2021 Aug;33(16):10551-62.
27. Nevendra M, Singh P. Empirical investigation of hyperparameter optimization for software defect count prediction. Expert Systems with Applications. 2022 Apr 1; 191:116217.
28. Prabha CL, Shivakumar N. Software defect prediction using machine learning techniques. In2020 4th International conference on trends in electronics and informatics (ICOEI)(48184) 2020 Jun 15 (pp. 728-733). IEEE.
29. Siddiqui T, Mustaqeem M. Performance evaluation of software defect prediction with NASA dataset using machine learning techniques. International Journal of Information Technology. 2023 Dec;15(8):4131-9.