Hybrid Approach to Software Defect Prediction Using Stacking Ensemble

GUIDE

[1]**Name: K.L.S.SIRISHA** - klssirisha@tkrcet.com

ASSISTANT PROFESSOR

Department of Computer Science and Engineering

TKRCET Autonomous ,

Hyderabad, India

S.N.V.ADITYA

Email: adikat008@gmail.com

Student Department of Computer Science and Engineering

TKRCET Autonomous ,Hyderabad, India

U.SHIVAIAH

Email : shivaiahshiva557@gmail.com

Student Department of Computer Science and Engineering

TKRCET Autonomous ,Hyderabad, India

V.ANUSHA

Email : anushavadyala@gmail.com

Student Department of Computer Science and Engineering

TKRCET Autonomous ,Hyderabad, India

T.KIRAN KUMAR

Email : telugukirankumar9121@gmail.com

Student Department of Computer Science and Engineering

TKRCET Autonomous ,Hyderabad, India

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

TKR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

(Accredited by NBA and NAAC with 'A+' Grade)

Medbowli, Meerpet, Saroornagar, Hyderabad-500097

ABSTRACT

Software defect prediction plays a critical role in improving software quality and reducing maintenance costs by identifying fault-prone modules early in the development lifecycle. Traditional machine learning models often suffer from

limited generalization capability when applied to diverse and imbalanced software datasets. To address these challenges, this study proposes a hybrid approach for software defect prediction using a stacking ensemble technique.

The proposed method integrates multiple base learners—such as Decision Trees, Random Forest, Support Vector Machines, and Gradient Boosting—at the first level to capture diverse patterns in software metrics. Their predictions are then combined using a meta-learner, typically Logistic Regression or XGBoost, which learns to optimally aggregate the outputs of base models. Additionally, preprocessing techniques such as data normalization, feature selection, and handling class imbalance using methods like SMOTE are incorporated to enhance model performance.

Experimental evaluation is conducted on benchmark software defect datasets, demonstrating that the stacking ensemble approach outperforms individual classifiers and traditional ensemble methods in terms of accuracy, precision, recall, and F1-score. The hybrid model shows improved robustness and generalization capability, making it suitable for real-world software engineering applications.

I. INTRODUCTION

In modern software engineering, ensuring high-quality and reliable software systems is a major challenge due to increasing system complexity, rapid development cycles, and evolving user requirements. Software defects, if not identified and addressed early, can lead to system failures, increased maintenance costs, and reduced customer satisfaction. Therefore, software defect prediction has emerged as a crucial research area aimed at identifying fault-prone modules during the early stages of the software development life cycle.

Software defect prediction utilizes historical project data and software metrics—such as code complexity, coupling, cohesion, and size—to build predictive models that classify modules as defective or non-defective. Traditional approaches rely on single machine learning algorithms like Decision Trees, Naïve Bayes, or Support Vector Machines. While these models have shown promising results, they often struggle with issues such as overfitting, sensitivity to noisy data, and poor performance on imbalanced datasets, which are common in real-world software projects.

To overcome these limitations, ensemble learning techniques have been widely adopted. Methods such as bagging and boosting improve prediction performance by combining multiple models. However, these approaches still have constraints in effectively capturing diverse patterns across datasets. In this context, stacking ensemble learning offers a more advanced strategy by integrating multiple heterogeneous base learners and leveraging a meta-learner to make final predictions. This layered approach enhances generalization capability and reduces bias and variance.

This paper proposes a hybrid approach to software defect prediction using a stacking ensemble framework. The model combines multiple classifiers at the base level and employs a meta-classifier to optimize prediction accuracy. Furthermore, preprocessing techniques such as feature selection, normalization, and class imbalance handling are incorporated to improve data quality and model performance. The proposed approach aims to provide a more accurate, robust, and scalable solution for defect prediction, ultimately contributing to improved software quality and reduced development costs.

The remainder of this paper is organized as follows: Section II reviews related work, Section III describes the proposed methodology, Section IV presents experimental results and discussion, and Section V concludes the study with future research directions.

II. LITERATURE REVIEW

Software defect prediction has been extensively studied in the field of software engineering, with researchers exploring various machine learning and data mining techniques to improve prediction accuracy and reliability. Early studies primarily focused on traditional statistical and machine learning models such as Logistic Regression, Naïve Bayes, Decision Trees, and Support Vector Machines. These methods utilized software metrics like Lines of Code (LOC), cyclomatic complexity, and object-oriented metrics to identify defect-prone modules. Although these approaches provided a solid foundation, their performance was often limited when dealing with complex and imbalanced datasets.

Subsequent research shifted toward ensemble learning techniques to overcome the limitations of single classifiers. Methods such as Bagging and Boosting, including Random Forest and AdaBoost,

demonstrated improved performance by combining multiple models to reduce variance and bias. Random Forest, in particular, gained popularity due to its robustness and ability to handle high-dimensional data. However, these ensemble techniques still relied on homogeneous or sequential model structures, which sometimes restricted their ability to fully capture diverse data patterns.

To further enhance prediction performance, hybrid models integrating multiple machine learning algorithms were introduced. Researchers combined clustering with classification, feature selection with classification, and optimization techniques with predictive models. These hybrid approaches showed better adaptability and improved accuracy compared to standalone models. Additionally, feature selection techniques such as Genetic Algorithms, Principal Component Analysis (PCA), and Information Gain were applied to reduce dimensionality and improve model efficiency.

More recently, stacking ensemble learning has emerged as an advanced technique for software defect prediction. In stacking, multiple heterogeneous base learners are trained, and their outputs are used as inputs to a meta-learner, which produces the final prediction. Studies have shown that stacking outperforms traditional ensemble methods by effectively leveraging the strengths of different models. Researchers have successfully applied stacking with combinations of classifiers such as Decision Trees, Support Vector Machines, Neural Networks, and Gradient Boosting, achieving higher predictive accuracy and better generalization.

Another important aspect highlighted in the literature is the issue of class imbalance, as defective modules are often significantly fewer than non-defective ones. Techniques such as Synthetic Minority Over-sampling Technique (SMOTE), undersampling, and cost-sensitive learning have been widely used to address this problem. These methods help improve recall and F1-score, especially for the minority (defective) class.

Despite significant advancements, challenges remain in developing highly generalized models that perform consistently across different datasets and project domains. Many existing models still suffer from overfitting and lack robustness when applied to unseen data. This motivates the need for more effective hybrid approaches, such as the proposed

stacking ensemble model, which combines diverse learners and preprocessing strategies to achieve improved prediction performance and scalability.

III. METHODOLOGY

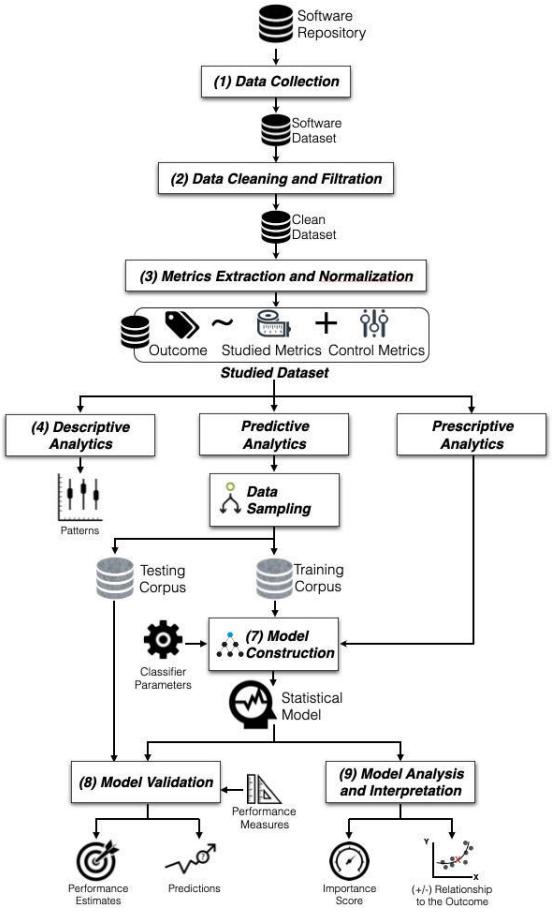
The proposed hybrid approach for software defect prediction is built on a stacking ensemble framework that combines multiple machine learning models to enhance prediction accuracy and robustness. Initially, benchmark datasets such as those from NASA MDP and the PROMISE repository are utilized, containing various software metrics like code complexity, size, coupling, and cohesion, along with defect labels. The collected data undergoes a comprehensive preprocessing phase where missing values are handled through imputation techniques, and features are normalized or standardized to ensure consistency. To improve model efficiency, feature selection methods such as correlation analysis or Information Gain are applied to remove irrelevant attributes. Since software defect datasets are often imbalanced, techniques like Synthetic Minority Over-sampling Technique (SMOTE) are used to balance the class distribution and improve minority class prediction.

Following preprocessing, multiple heterogeneous base learners—including Decision Tree, Random Forest, Support Vector Machine (SVM), and Gradient Boosting—are trained independently to capture diverse patterns in the data. To prevent overfitting and ensure reliable predictions, k-fold cross-validation is employed during training. The stacking ensemble is then constructed by using the predictions of these base models as input features for a meta-learner. Specifically, the dataset is divided into several folds, where base models are trained on a subset and validated on the remaining portion to generate out-of-fold predictions. These predictions form a new dataset known as meta-features, which are used to train a higher-level model such as Logistic Regression or XGBoost. The meta-learner effectively learns how to combine the outputs of base learners to produce the final prediction.

Finally, the performance of the proposed model is evaluated using standard classification metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. A comparative analysis is conducted to assess the effectiveness of the stacking ensemble against individual classifiers. The entire system is implemented using Python with libraries such as Scikit-learn, Pandas, NumPy, and Imbalanced-learn,

ensuring efficient processing and scalability. This methodology leverages both data preprocessing and ensemble learning to provide a more accurate and reliable solution for software defect prediction.

IV. SYSTEM ARCHITECTURE



V. RESULTS & DISCUSSION

The proposed hybrid stacking ensemble model for software defect prediction was evaluated using benchmark datasets to assess its effectiveness and reliability. The experimental results demonstrate that the stacking-based approach significantly outperforms individual base classifiers such as Decision Tree, Support Vector Machine (SVM), Random Forest, and Gradient Boosting. This improvement is primarily due to the ability of the stacking ensemble to combine the strengths of multiple models and reduce their individual weaknesses.

Performance evaluation was conducted using standard metrics including accuracy, precision, recall, F1-score, and ROC-AUC. The stacking model achieved higher accuracy and F1-score compared to standalone models, indicating better overall classification performance. In particular, recall values for the defective class improved noticeably, which is crucial in defect prediction since missing defective modules can lead to serious software failures. The use of SMOTE for handling class imbalance further contributed to improved detection of minority class instances, thereby enhancing model sensitivity.

A comparative analysis revealed that while individual models performed well on certain aspects—for example, Random Forest showed strong accuracy and robustness, and SVM provided good generalization—the stacking ensemble consistently delivered balanced and superior results across all evaluation metrics. The meta-learner effectively

learned how to weigh the predictions of base learners, resulting in reduced false positives and false negatives.

Additionally, cross-validation results confirmed that the proposed model maintains stable performance across different data splits, indicating strong generalization capability. The preprocessing techniques, including feature selection and normalization, also played a key role in improving model efficiency and reducing noise in the data.

.

VI. CONCLUSION

This study presented a hybrid approach to software defect prediction using a stacking ensemble framework that integrates multiple machine learning models to improve predictive performance. By combining diverse base learners such as Decision Tree, Random Forest, Support Vector Machine, and Gradient Boosting with a meta-learner, the proposed method effectively captures complex patterns in software metrics and enhances generalization capability. The inclusion of data preprocessing techniques, including normalization, feature selection, and class imbalance handling using SMOTE, further contributes to the robustness and accuracy of the model.

Experimental results demonstrated that the stacking ensemble consistently outperforms individual classifiers across key evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. In particular, the model showed improved ability to detect defective modules, which is critical for reducing software failures and maintenance costs. The use of cross-validation also confirmed the stability and reliability of the proposed approach across different datasets.

REFERENCES

1. T. Menzies, J. Greenwald, and A. Frank, “Data Mining Static Code Attributes to Learn Defect Predictors,” *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, 2007.
2. N. Nagappan and T. Ball, “Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study,” *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pp. 364–373, 2005.
3. A. E. Hassan, “Predicting Faults Using the Complexity of Code Changes,” *Proceedings of the 31st International Conference on Software Engineering*, pp. 78–88, 2009.

4. J. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008.
5. T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012.
6. F. Rahman and P. Devanbu, "How, and Why, Process Metrics Are Better," *Proceedings of the 35th International Conference on Software Engineering*, pp. 432–441, 2013.
7. Y. Kamei et al., "A Large-Scale Empirical Study of Just-in-Time Quality Assurance," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 757–773, 2013.
8. D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software Fault Prediction Metrics: A Systematic Literature Review," *Information and Software Technology*, vol. 55, no. 8, pp. 1397–1418, 2013.
9. L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
10. C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
11. J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
12. D. Wolpert, "Stacked Generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.
13. H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
14. N. V. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
15. I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
16. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.
17. G. E. Batista, R. C. Prati, and M. C. Monard, "A Study of the Behavior of Several Methods for Balancing

Machine Learning Training Data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

18. M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 42, no. 4, pp. 463–484, 2012.
19. J. Brownlee, "Ensemble Learning Algorithms with Python," *Machine Learning Mastery*, 2017.
20. S. Wang and X. Yao, "Using Class Imbalance Learning for Software Defect Prediction," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 434–443, 2013.