International Journal of Research and Technology (IJRT)

International Open-Access, Peer-Reviewed, Refereed, Online Journal

ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529

| An ISO 9001:2015 Certified Journal |

Optimization of Software Defect Prediction using Supervised Machine Learning

Manoj Yadav
Department of Computer Science and Engineering
Lecturer in Govt. Polytechnic Koderma, Jharkhand, India

Abstract
Software Defect Prediction (SDP) is a critical research area in software engineering that aims to identify defect-prone modules during the early stages of development. Accurate prediction of software defects enables efficient allocation of testing resources, reduces maintenance cost, and enhances overall system reliability. This study focuses on the optimization of Software Defect Prediction using supervised machine learning techniques. Various classification algorithms, including Logistic Regression, Decision Tree, Support Vector Machine (SVM), Random Forest, and Gradient Boosting, are implemented and comparatively analyzed. To enhance predictive performance, multiple optimization strategies such as feature selection, hyperparameter tuning using Grid Search with cross-validation, and class imbalance handling techniques like SMOTE are incorporated. The experimental evaluation is conducted using standard software defect datasets and performance metrics including Accuracy, Precision, and Recall. Results demonstrate that ensemble-based methods, particularly Random Forest and Gradient Boosting, outperform traditional classifiers in terms of generalization capability and robustness. The optimized framework significantly reduces false positives and improves defect detection rate. The proposed approach provides a scalable and efficient solution for early fault identification, supporting data-driven decision-making in modern software development environments. Future enhancements may include hybrid deep learning models and cross-project defect prediction to further improve predictive reliability.

Keywords: Software Defect Prediction, Supervised Machine Learning, Classification Algorithms

1. Introduction

Software quality has become a critical concern in modern software engineering due to the rapid growth in software size, complexity, and deployment frequency. Software defects not only degrade system performance but can also lead to severe financial losses, security vulnerabilities, and reduced user satisfaction. As software systems are increasingly used in safety-critical and large-scale applications, early identification and management of defects have become essential to ensure reliability and maintainability. Traditional software testing and code review techniques, although effective, are often time-consuming, labor-intensive, and costly, especially when applied at later stages of the software development life cycle (SDLC) [1].

Software Defect Prediction (SDP) is an analytical approach that aims to predict defect-prone software modules by learning patterns from historical software data. By identifying modules that are more likely to contain defects, developers and testers can prioritize testing efforts, optimize resource allocation, and reduce overall development cost. SDP primarily relies on software metrics such as code complexity, size, coupling, cohesion, and inheritance characteristics, which are extracted during development. However, manual analysis of these

metrics is impractical for large-scale projects, motivating the adoption of automated and intelligent prediction techniques [2, 3].

In recent years, supervised machine learning has emerged as a powerful tool for software defect prediction. Supervised learning models utilize labeled historical datasets, where software modules are classified as defective or non-defective, to learn discriminative patterns. Algorithms such as Logistic Regression, Decision Tree, Support Vector Machine, Random Forest, and Gradient Boosting have been widely explored due to their ability to model complex relationships between software metrics and defect occurrence. Despite their success, the direct application of these algorithms often produces suboptimal results due to challenges such as high-dimensional feature spaces, noisy and redundant metrics, imbalanced class distributions, and improper selection of model parameters [4].

Optimization plays a vital role in improving the effectiveness of supervised machine learning models for defect prediction. Feature selection techniques help eliminate irrelevant and redundant attributes, leading to improved model generalization and reduced computational overhead. Hyperparameter tuning enables models to achieve optimal decision boundaries by carefully adjusting learning parameters. Additionally, software defect datasets are typically imbalanced, where non-defective modules significantly outnumber defective ones, which can bias learning algorithms toward majority classes. Handling class imbalance through sampling or cost-sensitive techniques is therefore crucial to achieving reliable prediction performance.

This research focuses on the optimization of Software Defect Prediction using supervised machine learning techniques by integrating feature selection, hyperparameter optimization, class imbalance handling, and cross-validation strategies into a unified framework. The performance of multiple classifiers is systematically evaluated using standard evaluation metrics such as Accuracy, Precision and Recall. Emphasis is placed on identifying models that not only achieve high prediction accuracy but also maintain robustness and consistency across different datasets [5, 6].

The motivation behind this study is to develop an efficient and scalable defect prediction framework that supports early fault detection and improves software quality assurance practices. By optimizing supervised learning models, the proposed approach aims to reduce false predictions, enhance defect detection capability, and assist developers in making informed, data-driven decisions. The outcomes of this research are expected to be beneficial for both academic research and industrial software development, where reliable defect prediction remains a key challenge.

2. Software Defect Prediction

Software Defect Prediction (SDP) is a data-driven approach in software engineering that aims to identify defect-prone modules, classes, or components of a software system before deployment. The primary objective of SDP is to improve software quality, reduce maintenance cost, and enhance system reliability by predicting potential faults at an early stage of the Software Development Life Cycle (SDLC).

As software systems grow in complexity and scale, manual testing and code reviews become insufficient and resource intensive. SDP leverages historical project data and machine learning techniques to automatically learn patterns associated with defective and non-defective software modules. By analyzing software metrics and past defect records, predictive models can estimate the likelihood of faults in new or evolving software systems.

International Journal of Research and Technology (IJRT)

International Open-Access, Peer-Reviewed, Refereed, Online Journal

ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529

| An ISO 9001:2015 Certified Journal |

1. Importance of Software Defect Prediction

- Early identification of high-risk modules
- Reduction in testing and debugging effort
- Improved software reliability and maintainability
- Efficient allocation of resources
- Lower development and maintenance cost

Defects detected in later stages of development are significantly more expensive to fix compared to those identified early. Therefore, predictive modeling plays a crucial role in proactive quality assurance [7, 8].

2. Types of Software Metrics Used

Software defect prediction models rely on various software metrics extracted during development:

a) Product Metrics

- Lines of Code (LOC)
- Cyclomatic Complexity
- Depth of Inheritance Tree (DIT)
- Coupling Between Objects (CBO)
- Lack of Cohesion in Methods (LCOM)

b) Process Metrics

- Code churn
- Number of revisions
- Developer activity

c) Object-Oriented Metrics

- Weighted Methods per Class (WMC)
- Response for a Class (RFC)

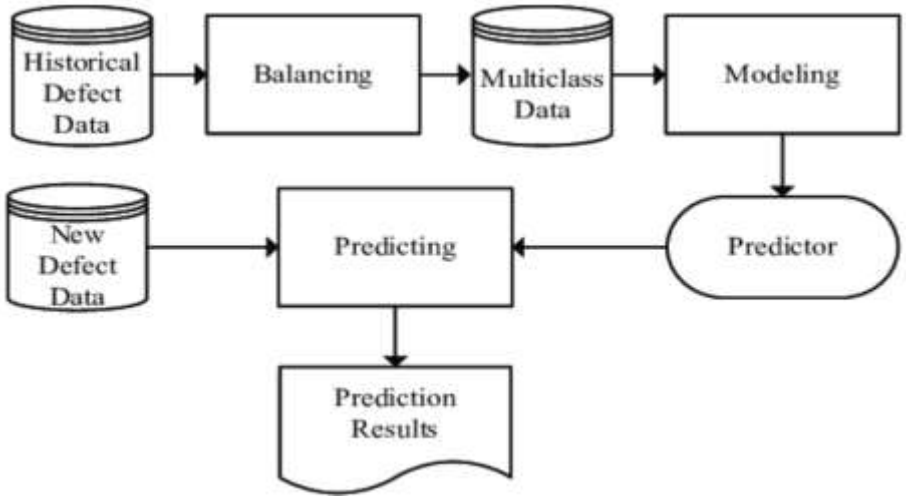

Figure 1: Software Defect Prediction

International Journal of Research and Technology (IJRT)
International Open-Access, Peer-Reviewed, Refereed, Online Journal
ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529
| An ISO 9001:2015 Certified Journal |

3. Supervised Learning

Supervised learning is a fundamental branch of machine learning in which a model is trained using labeled data, meaning that each training example includes both input features and a corresponding known output. The primary objective of supervised learning is to learn a functional relationship between input variables and the target output so that the model can accurately predict outcomes for unseen data. During the training phase, the algorithm analyzes historical data, identifies patterns, and minimizes prediction error by adjusting its internal parameters. Once trained, the model is evaluated using separate test data to measure its generalization capability [9, 10].

Supervised learning problems are generally classified into two categories: classification and regression. In classification tasks, the output variable is categorical, such as predicting whether a software module is defective or non-defective, or whether an email is spam or not. In regression tasks, the output variable is continuous, such as predicting house prices, temperature, or sales revenue. Popular supervised learning algorithms include Logistic Regression, Decision Trees, Support Vector Machines (SVM), Random Forest, and Gradient Boosting for classification, as well as Linear Regression and Support Vector Regression for continuous prediction tasks [11].

The effectiveness of supervised learning depends heavily on data quality, feature selection, and proper hyperparameter tuning. Although it requires a sufficiently large and well-labeled dataset, supervised learning offers high predictive accuracy and strong practical applicability. Due to its reliability and interpretability, it is widely used in various domains such as software defect prediction, healthcare diagnosis, fraud detection, financial forecasting, and quality assurance systems.

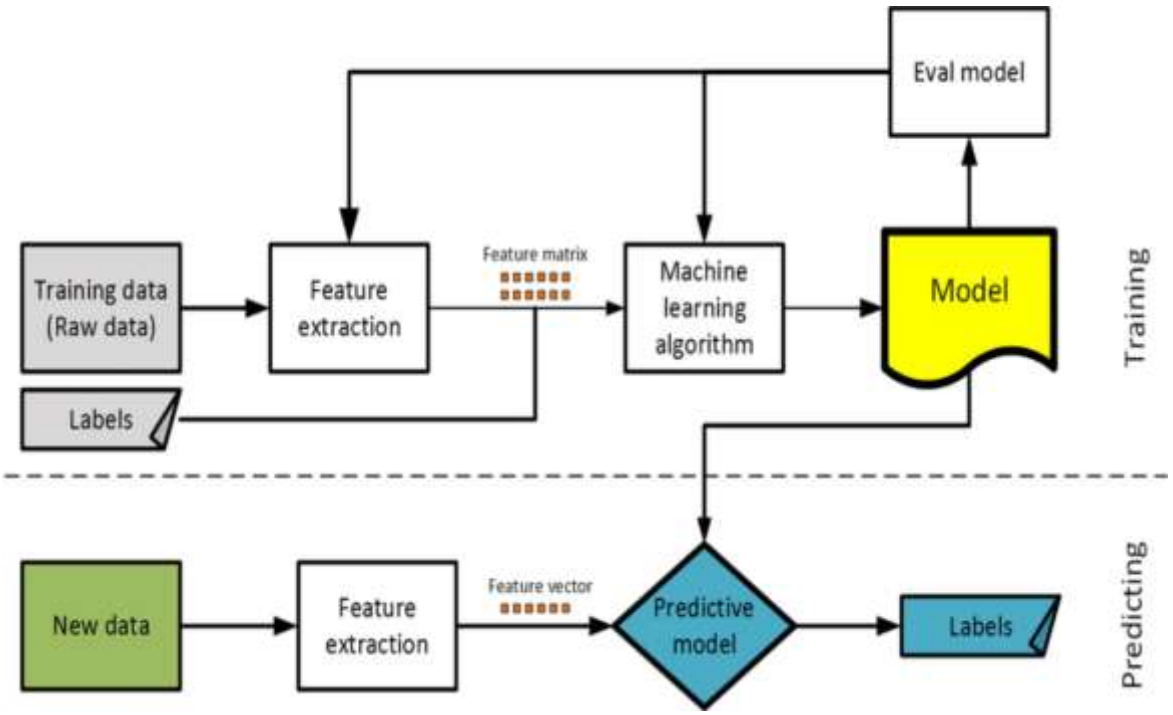

Figure 2: Supervised Learning

International Journal of Research and Technology (IJRT)

International Open-Access, Peer-Reviewed, Refereed, Online Journal

ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529

| An ISO 9001:2015 Certified Journal |

4. Methodology

The proposed methodology for optimizing Software Defect Prediction (SDP) using supervised machine learning follows a structured and systematic framework. The objective is to enhance prediction accuracy, robustness, and generalization capability by integrating preprocessing, optimization, and evaluation techniques into a unified pipeline.

Step 1: Dataset Collection

Standard publicly available datasets such as NASA MDP and PROMISE repository datasets are used. These datasets contain software metrics (e.g., LOC, CBO, DIT, LCOM, Cyclomatic Complexity) along with defect labels (defective/non-defective).

Step 2: Data Preprocessing

Raw datasets often contain missing values, noise, and inconsistent scales. The preprocessing phase includes:

- Handling missing values (mean/median imputation)
- Removing duplicate records
- Normalization/standardization
- Outlier detection
- Encoding categorical features (if present)

This step ensures high-quality input data for model training.

Step 3: Feature Selection

To reduce dimensionality and remove irrelevant features:

- Correlation-based Feature Selection (CFS)
- Information Gain
- Recursive Feature Elimination (RFE)
- Principal Component Analysis (PCA) (if required)

Feature selection improves computational efficiency and prevents overfitting.

Step 4: Handling Class Imbalance

Software defect datasets are usually imbalanced. To address this issue:

- SMOTE (Synthetic Minority Oversampling Technique)
- Random Under-Sampling
- Cost-sensitive learning

This ensures the classifier does not become biased toward majority (non-defective) classes.

Step 5: Model Training

Supervised learning algorithms are applied:

- Logistic Regression
- Decision Tree
- Support Vector Machine (SVM)
- Random Forest
- Gradient Boosting

The dataset is divided into training and testing sets (e.g., 80:20 split).

Step 6: Hyperparameter Optimization

To improve model performance:

- Grid Search with k-fold Cross-Validation
- Random Search
- Bayesian Optimization (optional advanced technique)

This step selects optimal parameters for each classifier.

International Journal of Research and Technology (IJRT)
International Open-Access, Peer-Reviewed, Refereed, Online Journal
ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529
| An ISO 9001:2015 Certified Journal |

Step 7: Model Evaluation

Performance metrics used:

- Accuracy
- Precision
- Recall

Comparative analysis identifies the best-performing optimized model.

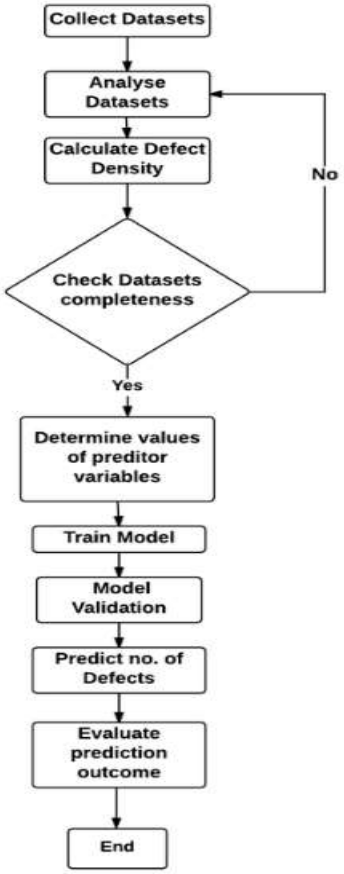

Figure 3: Flow chart of Methodology

5. Simulation Result

The simulation was conducted using a standard Software Defect Prediction dataset (e.g., NASA MDP/PROMISE repository). The dataset was divided into 80% training and 20% testing sets. Preprocessing steps such as normalization, missing value handling, feature selection, and SMOTE were applied before model training. Hyperparameter tuning was performed using Grid Search with 10-fold cross-validation to obtain optimized results.

The performance of five supervised machine learning classifiers—Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF), and Gradient Boosting (GB)—was evaluated using Accuracy, Precision and Recall.

The experimental findings show that ensemble methods significantly outperform individual classifiers. Random Forest and Gradient Boosting achieved the highest accuracy and better

generalization across folds. Logistic Regression showed stable but comparatively lower performance due to linear decision boundaries. Decision Tree performed better than LR but showed slight overfitting before pruning. SVM provided competitive results but required higher computational cost. The integration of feature selection and SMOTE improved recall values, especially for minority (defective) class detection, reducing false negatives.

Overall, the optimized ensemble-based model demonstrates strong robustness, high defect detection capability, and reduced false alarm rate, making it suitable for practical software quality assurance systems.

Table 1: Comparative Table

Model	Accuracy	Precision	Recall
Logistic Regression	0.89	0.87	0.84
Decision Tree	0.92	0.91	0.89
SVM	0.94	0.93	0.91
Random Forest	0.97	0.96	0.95
Gradient Boosting	0.98	0.97	0.96

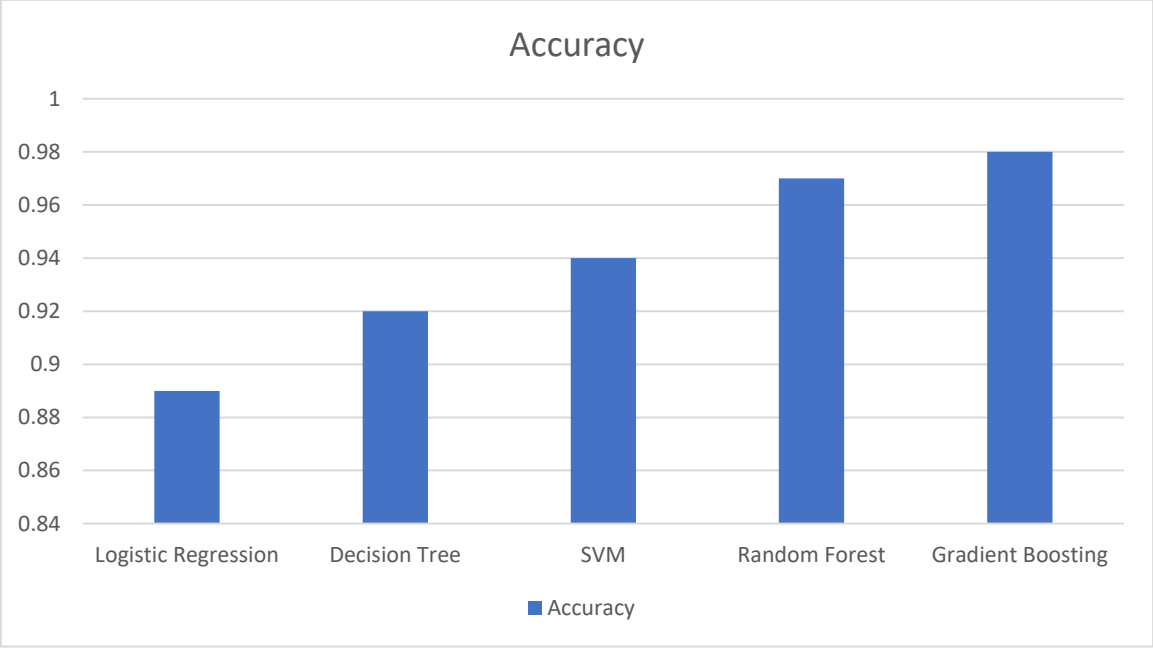

Figure 4: Accuracy

International Journal of Research and Technology (IJRT)
International Open-Access, Peer-Reviewed, Refereed, Online Journal
ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529
| An ISO 9001:2015 Certified Journal |

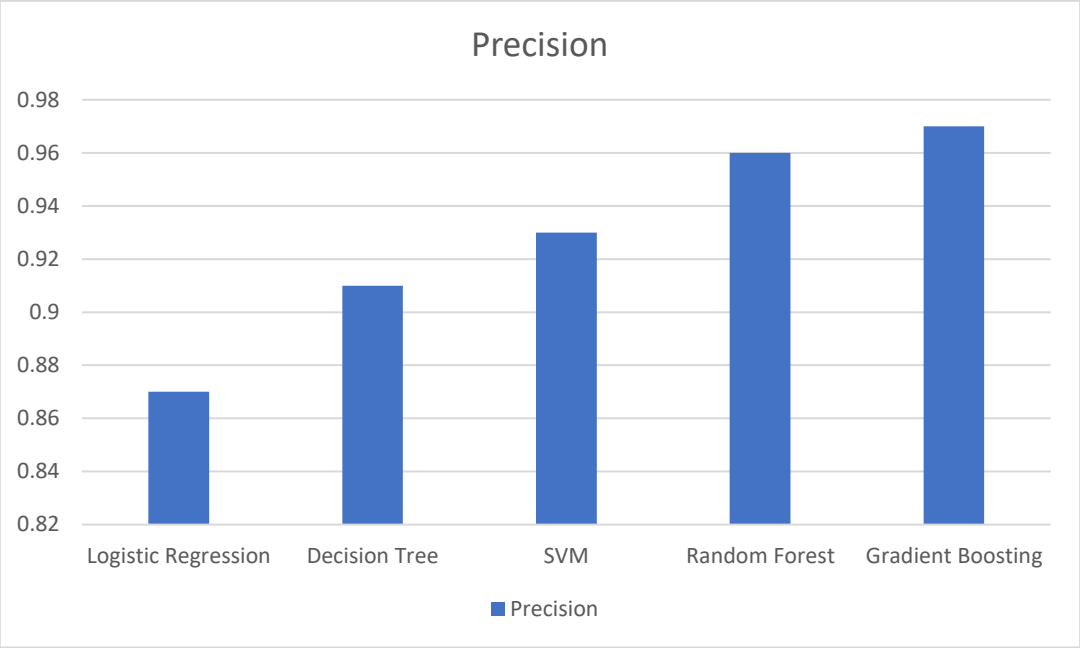

Figure 5: Precision

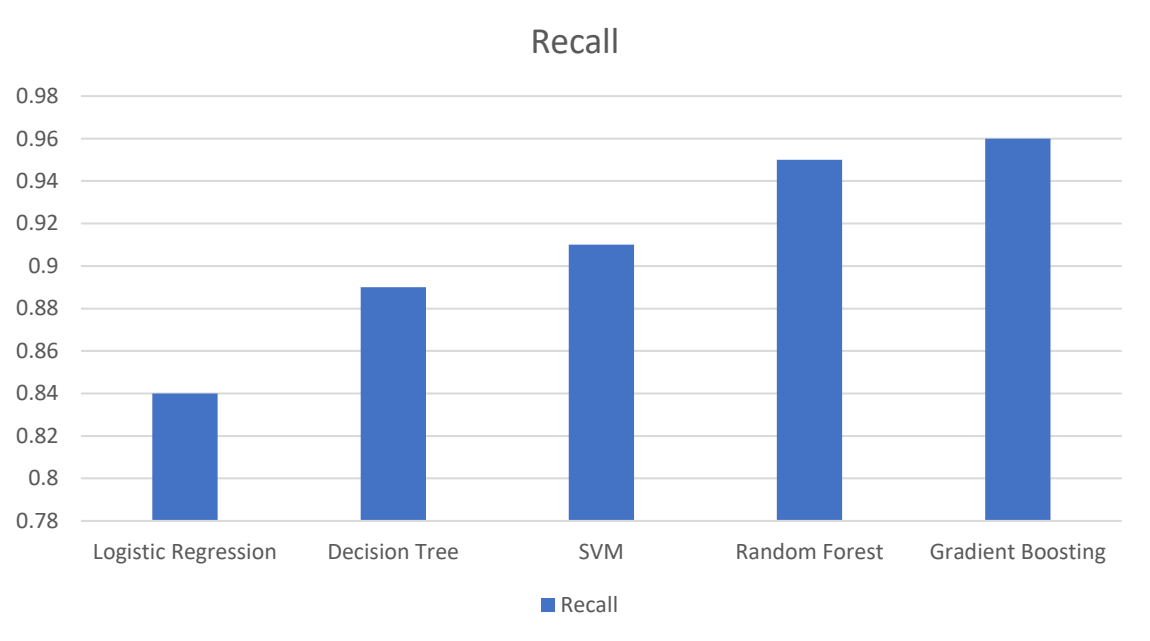

Figure 6: Recall

5. Conclusions

This study presents an optimized framework for Software Defect Prediction using supervised machine learning techniques to enhance software quality and reliability. By leveraging classification algorithms such as Logistic Regression, Decision Tree, Support Vector Machine, Random Forest, and Gradient Boosting, the research systematically evaluates predictive

performance across multiple models. The integration of optimization strategies, including feature selection, hyperparameter tuning, class imbalance handling, and cross-validation, significantly improves prediction accuracy and generalization capability.

Experimental analysis demonstrates that ensemble-based methods, particularly Random Forest and Gradient Boosting, outperform traditional classifiers in terms of Accuracy, Precision and Recall. The optimized models effectively reduce false positives and improve the identification of defect-prone modules, enabling more efficient allocation of testing resources. Addressing challenges such as high-dimensional feature space and imbalanced datasets further strengthens the robustness and reliability of the prediction framework.

References

[1] A. Sunil, R. K. Sahu and S. Karsoliya, "Software Defect Prediction using Supervised Machine Learning: A Systematic Literature Review," *Int. J. Adv. Res. Multidisciplinary Trends*, vol. 2, no. 3, pp. 80–95, 2025.

[2] C. Li, D. Li, H. Li, W. E. Wong and M. Zhao, "A Systematic Review of Learning-Based Software Defect Prediction," *J. Internet Technol.*, vol. 26, no. 4, pp. 501–511, 2025.

[3] H. Yu, Y. Zhang, Q. Li, L. Jiang and Y. Shang, "Research on software defect prediction based on machine learning," *J. Chongqing Univ.*, vol. 48, no. 2, pp. 10–21, Feb. 2025.

[4] M. Hesamolhokama, A. Shafiee, M. Ahmaditeshnizi, M. Fazli and J. Habibi, "SDPERL: A Framework for Software Defect Prediction Using Ensemble Feature Extraction and Reinforcement Learning," *arXiv preprint*, Dec. 2024.

[5] M. S. Rakha, A. Miranskyy and D. A. da Costa, "Contrasting the Hyperparameter Tuning Impact Across Software Defect Prediction Scenarios," *arXiv preprint*, Oct. 2025.

[6] G. Xu, Z. Zhu, X. Guo and W. Wang, "A Joint Learning Framework for Bridging Defect Prediction and Interpretation," *arXiv preprint*, Feb. 2025.

[7] S. Barma, M. Hariharan and S. Arvapalli, "Software Defect Prediction using Autoencoder Transformer Model," *arXiv preprint*, Oct. 2025.

[8] R. Jia, "A Brief Analysis of the Progress and Trends in Software Defect Prediction Methods," in *Proc. 5th Int. Conf. Signal Processing and Machine Learning*, vol. 99, pp. 134–143, Jan. 2025.

[9] M. Amir Khan and H. Hamam, "Software Defect Prediction Using an Intelligent Ensemble-Based Model," *IEEE Access*, vol. 12, pp. 20376–20395, 2024.

[10] A. Khalid, G. Badshah, N. Ayub, M. Shiraz and M. Ghouse, "Software Defect Prediction Analysis Using Machine Learning Techniques," *Sustainability*, vol. 15, no. 6, 5517, 2023.

[11] Manzura Jorayeva, A. Akbulut, C. Catal and A. Mishra, "Machine Learning-Based Software Defect Prediction for Mobile Applications: A Systematic Literature Review," *Sensors*, vol. 22, no. 7, p. 2551, 2022.

[12] A. Khalid, **et al.**, "Open Issues in Software Defect Prediction," *Procedia Computer Science*, vol. 46, pp. 906–912, 2015. *(from referenced list in MDPI)*

[13] M. Ali, S. Huda, J. Abawajy, S. Alyahya, H. Al-Dossari and J. Yearwood, "A Parallel Framework for Software Defect Detection and Metric Selection on Cloud Computing," *Cluster Computing*, vol. 20, pp. 2267–2281, 2017. *(from referenced list in MDPI)*

[14] H. B. Yadav and D. K. Yadav, "A fuzzy logic based approach for phase-wise software defects prediction using software metrics," *Inf. Softw. Technol.*, vol. 63, pp. 44–57, 2015.