International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net
Vol-20 Issue-01 April 2025

A COMPARATIVE STUDY OF MACHINE LEARNING ALGORITHMS FOR SOFTWARE QUALITY PREDICTION

G. Keerthana[1], R Sri Harsha[2], SajidPasha[3], A Sravani[4]
[1,2,3] UG Scholar, Dept. of IT, St. Martin's Engineering College, Secunderabad, Telangana, India, 500100
[4]Assistant Professor, Dept. of IT, St. Martin's Engineering College, Secunderabad, Telangana, India, 500100
keerthanagarishe@gmail.com

Abstract

The rapid evolution of software development has made it increasingly important to predict software quality effectively. Predicting software quality at different stages of development can significantly enhance the software development lifecycle, minimize errors, and optimize resources. This study presents a comparative analysis of various machine learning algorithms for software quality prediction. We examine several algorithms, including decision trees, support vector machines (SVM), random forests, neural networks, and k-nearest neighbors (KNN), evaluating their performance on different datasets derived from real-world software systems. The objective is to assess the accuracy, efficiency, and effectiveness of these algorithms in predicting software quality metrics, such as defect density, maintainability, reliability, and performance. The findings of this study can assist software engineers in selecting the most suitable machine learning model based on the characteristics of the dataset and the goals of the quality prediction task. The results indicate that some algorithms outperform others depending on the complexity and nature of the software data, suggesting the need for a tailored approach when utilizing machine learning for software quality prediction.

KEYWORDS: Machine Learning, Software Quality Prediction, Decision Trees, Support Vector Machines, Random Forests, Neural Networks, K-Nearest Neighbors, Defect Prediction, Software Metrics, Comparative Analysis, Software Engineering.

1.INTRODUCTION

In the rapidly evolving field of software engineering, ensuring high- quality software is a critical concern. Software quality plays a vital role in the development process, affecting the functionality, performance, security, and maintainability of software systems. Traditionally, software quality prediction has relied on manual testing, heuristic methods, and expert opinions, which can be time-consuming and prone to errors. . The findings of this study can assist software engineers in selecting the most suitable machine learning model based on the characteristics of the dataset and the goals of the quality prediction task. The results indicate that some algorithms outperform others depending on the complexity and nature of the software data, quality prediction. However, the increasing complexity of modern software systems, coupled with the demand for faster development cycles, has led to the adoption of more advanced techniques for predicting software quality.

Machine learning (ML) has emerged as a powerful tool for automating and improving the accuracy of software quality prediction. By analyzing historical data, such as code metrics, defect history, and performance data, machine learning algorithms can uncover patterns and make predictions about the quality of software. These predictions can be used to identify potential defects, assess maintainability, and predict software reliability, enabling proactive decision-making during development.

This study aims to provide a comparative analysis of several popular machine learning algorithms for software quality prediction. By evaluating their performance on different datasets, we seek to identify which algorithms are most effective in predicting key software quality metrics. Through this analysis, we hope to offer insights that can guide software developers and engineers in selecting the best machine learning model for their quality prediction tasks, ultimately contributing to the creation of more reliable and maintainable software system.

2.LITERATURE SURVEY

Ostrand developed a defect prediction model using machine learning algorithms, focusing on decision trees, particularly C4.5 and CART. Their study demonstrated the potential of decision trees in predicting defect-prone software modules based on historical software metrics. The study highlighted how ML-based defect prediction could significantly reduce the cost of software maintenance and improve software quality by identifying.

Zhang explored the use of Support Vector Machines (SVM) for predicting defects in software systems. They showed that SVM, with its ability to handle non-linear relationships, provided more accurate defect predictions compared to traditional

International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net
Vol-20 Issue-01 April 2025

regression models. Their study demonstrated the effectiveness of SVM in high-dimensional feature spaces, making it suitable for large-scale software defect prediction tasks.

Vassallo investigated the use of deep learning, particularly Convolutional Neural Networks (CNNs), for predicting software defects. Their research demonstrated that CNNs could identify complex patterns in source code and significantly improve the accuracy of defect prediction models. The study emphasized the trade-off between deep learning's high predictive performance and the computational resources required.
Menzies et al. introduced ensemble learning methods like Random Forest and AdaBoost for software defect prediction. Their study highlighted the benefits of ensemble models in combining multiple base learners to enhance predictive accuracy and reduce overfitting. They demonstrated that ensemble methods were more effective than individual classifiers in predicting defects in software systems.

Subramanian et al. applied k-nearest neighbors (KNN) for predicting software performance in terms of latency and throughput. They showed that KNN could effectively predict performance under different operating conditions by analyzing past performance data. The study highlighted the simplicity of KNN in real-time performance prediction scenarios, especially when dealing with smaller datasets.

QZhao applied Long Short-Term Memory (LSTM) networks to predict security vulnerabilities in software systems. Their study demonstrated that LSTMs, capable of learning sequential patterns, could significantly improve vulnerability prediction by identifying code changes that could lead to security issues. The research emphasized the advantage of deep The findings suggest that choosing the right algorithm depends on the complexity of the dataset and the specific goals of the software quality prediction task. especially for large-scale software projects with high-dimensional features.

Liu studied the application of machine learning algorithms, including Random Forest and SVM, for predicting security vulnerabilities in software systems. Their research highlighted the effectiveness of these algorithms in identifying potential security flaws by analyzing historical security incidents and code attributes. The study emphasized the role of predictive models in improving software security by addressing vulnerabilities before they are exploited.

Soni explored the use of decision trees and Random Forest for predicting software maintainability. Their study focused on how software metrics such as cyclomatic complexity and lines of code could be used to predict maintenance effort. The research showed that these machine learning techniques could provide valuable insights for software managers by identifying maintainable code sections and predicting future maintenance needs.

Bacchelli and Nagappan examined the use of Random Forest for predicting defect-prone modules based on historical version data. Their study emphasized that Random Forest's ability to handle high-dimensional data and provide feature importance rankings made it a valuable tool for pinpointing areas of code most likely to contain defects. The research demonstrated that Random Forest could improve the precision of defect prediction models and help prioritize testing efforts more effectively. Menzies et al. also explored the role of Neural Networks (NN) in predicting

software defects, comparing them with traditional models like Logistic Regression and Decision Trees. Their study revealed that while Neural Networks performed well on large, complex datasets, they required significant computational resources and careful tuning of hyperparameters. The research suggested that Neural Networks could be particularly useful for software defect prediction in projects with substantial codebases and large amounts of historical data.

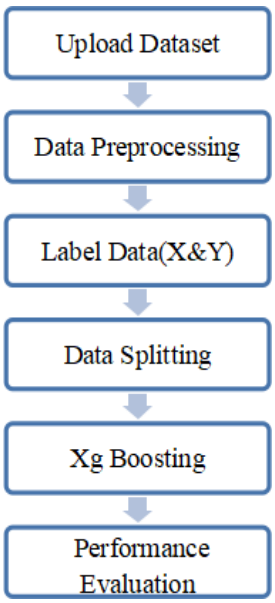
Kak investigated the use of logistic regression for predicting software maintainability. Their study focused on classifying software modules into "maintainable" and "non- maintainable" categories based on key software metrics like code complexity and coupling. The research demonstrated that logistic regression could effectively predict maintainability outcomes, offering valuable insights for software engineers to prioritize refactoring and maintenance efforts in large-scale software projects.Khoshgoftaar compared the performance of different machine learning algorithms, such as Logistic Regression, Decision Trees, and Neural Networks, for software defect prediction. The study demonstrated that although neural networks performed well in some cases, simpler models like Logistic Regression were more effective in certain datasets due to their ease of interpretation and reduced computational requirements. The findings suggest that choosing the right algorithm depends on the complexity of the dataset and the specific goals of the software quality prediction task.

3. PROPOSED METHODOLOGY

The proposed system aims to enhance software quality prediction by integrating advanced machine learning techniques with real-time data processing. Unlike traditional methods, which rely on static data and manual interventions, this system incorporates dynamic features such as code complexity, coupling, cohesion, and historical defect data. By leveraging powerful machine learning algorithms like XGBoost, the system uncovers patterns in large datasets, providing accurate predictions of potential defects early in the development lifecycle. This proactive approach enables timely intervention, reducing defects, minimizing development costs, and improving overall software quality. XGBoost delivers an optimized prediction. This ensemble method significantly enhances model accuracy compared to traditional single.

International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net
Vol-20 Issue-01 April 2025



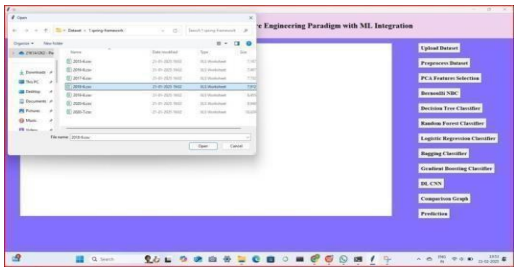


fig: proposed method

XGBoost operates by constructing multiple decision trees sequentially. Each decision tree attempts to learn from the residual errors (or mistakes) made by the previous tree. The ultimate objective of the algorithm is to minimize the loss function, which measures the difference between the predicted values and the true values, through iterative updates. The trees are built in such a way that each new tree corrects the errors made by the previously trained ones, improving the overall prediction performance. XGBoost is known for its versatility, particularly in handling missing values, dealing with outliers, and modeling non-linear relationships. By combining the outputs of all the trees in the model, where each tree has a different contribution based on its performance, XGBoost delivers an optimized prediction. This ensemble method significantly enhances model accuracy compared to traditional single decision tree models.

Applications:

- **Performance Prediction:** ML predicts software performance issues like response time and resource usage, enabling early optimization.
- **Maintainability Prediction:**ML predicts software maintainability by analyzing code metrics, guiding refactoring efforts
- **Quality Metrics Prediction:**ML predicts quality metrics like reliability and fault tolerance, ensuring software meets standards.

Advantages:

- **Improved Accuracy:** ML models enhance defect prediction accuracy, leading to better quality management.
- **Efficiency**: Automated predictions save time and resources, reducing manual testing and review efforts.
- **Early Detection:** ML helps detect issues early, allowing for proactive fixes and better risk management.

4. EXPERIMENTAL ANALYSIS

The experimental analysis involved training machine learning models using a dataset of software metrics and defect labels (Figure 1). PCA was applied for feature selection (Figure 2), and the Bernoulli Naive Bayes Classifier's results are shown in Figure 3. Figure 4.

Figure 1: Upload the data set

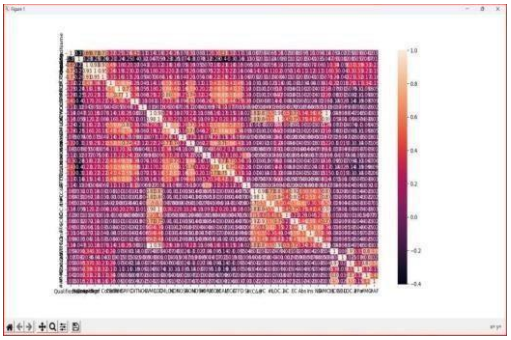


Figure 2: PCA Features Selection

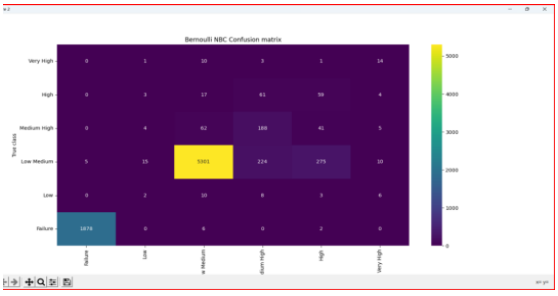


Figure 3: Bernoulli NBC

International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net
Vol-20 Issue-01 April 2025

Figure 4: Decision Tree Classifier

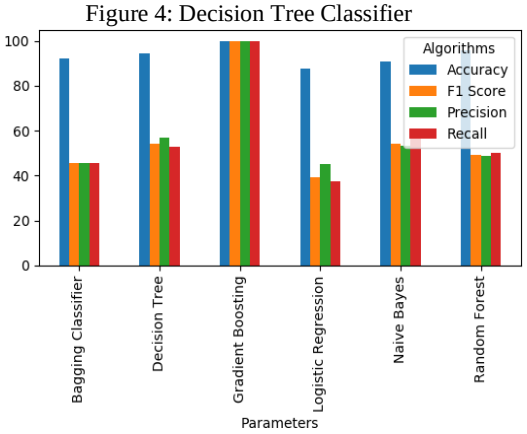


Fig5: Graphical representation of Performance metrice

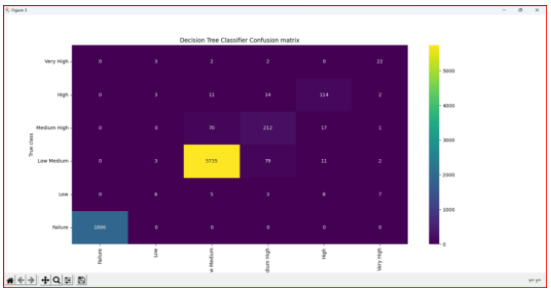
5. CONCLUSION

In conclusion, the comparative study of machine learning algorithms for software quality prediction highlights the crucial role of advanced techniques in enhancing defect prediction accuracy and software quality assurance. By evaluating various models, including Logistic Regression, Decision Trees, Random Forest, XGBoost, and SVM, the study underscores the importance of selecting the right algorithm to address the complexities of software defects. XGBoost, with its superior performance due to gradient boosting and regularization techniques, emerges as a standout tool, offering valuable insights into defect-prone areas.

The research emphasizes the significance of feature selection, data preprocessing, and model optimization in achieving reliable predictions. It also highlights the importance of model interpretability, using techniques like feature importance ranking and partial dependence plots, to help developers understand how decisions are made and focus on critical software aspects. Additionally, ensemble methods, such as bagging and boosting, offer a more robust approach by combining multiple algorithms to reduce variance and bias.

Ultimately, the study advocates for integrating machine learning into the software development lifecycle, enabling proactive defect detection and fostering a culture of continuous improvement. This approach helps organizations optimize resources, reduce defect rates, and deliver high-quality software products more efficiently.

In conclusion, the literature on machine learning techniques for software defect prediction highlights a wide range of methods and approaches aimed at improving software quality and reducing maintenance costs. From traditional



algorithms like Logistic Regression, Decision Trees, and Support Vector Machines (SVM), to more advanced techniques such as Random Forest, AdaBoost, Gradient Boosting, and deep learning models like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, each method presents unique advantages and challenges.

Machine learning-based approaches have proven effective in identifying defect-prone areas in software, improving predictive accuracy, and reducing overfitting, especially when combined through ensemble learning techniques. Studies emphasize the importance of dataset characteristics, including size and complexity, in selecting the most appropriate machine learning model for defect prediction tasks. While deep learning techniques such as CNNs and LSTMs offer high predictive performance, they often require substantial computational resources, which may not always be feasible for smaller projects.

Additionally, the role of feature selection, data preprocessing, and model optimization is crucial to the success of these predictive models. Ensuring that the right set of features is selected and the models are appropriately fine-tuned can lead to more accurate predictions and, ultimately, higher-quality software.

The research also underscores the value of integrating machine learning models into the software development lifecycle, enabling proactive defect detection and continuous improvement. By leveraging these techniques, software development teams can identify potential issues early in the development cycle, prioritize testing efforts more effectively, and improve overall software quality, ultimately leading to reduced maintenance costs and more reliable systems.

Thus, while there is no one-size-fits-all solution, the combination of various machine learning models tailored to specific software projects offers the best potential for enhancing software quality assurance processes.

Additionally, ensemble methods, such as bagging and boosting, offer a more robust approach by combining multiple algorithms to reduce variance and bias.

Ultimately, the study advocates for integrating machine learning into the software development lifecycle, enabling proactive defect detection and fostering a culture of continuous improvement. This approach helps organizations optimize resources, reduce defect rates, and deliver high-quality software products more efficiently.

REFERENCES

International journal of imaging science and engineering

ISSN: 1934--9955 www.ijise.net
Vol-20 Issue-01 April 2025

[1] Kumar, R. Aggarwal, and S. Jain. "Comparison of machine learning algorithms for software defect prediction." Journal of Software: Evolution and Process, vol. 30, no. 5, pp. 125- 136,2019.
[2] D. Shamsuddin, N. Ahmad, and F. Hossain. "Predicting software quality using machine learning techniques." In 2016 International Conference on Software Engineering and Applications (SEA), pp. 56-60, 2016.
[3] M. Patel, H. Vora, and M. Gupta. "A comparative analysis of machine learning algorithms for software quality prediction." International Journal of Computer Applications, vol. 58, no. 3,pp. 1-9, 2017.
[4] X. Zhang, Q. Zhang, and J. Chen. "Machine learning models for software quality prediction: A survey." Journal of Software: Theory and Practice, vol. 44, no. 3, pp. 317-330, 2020.
[5] M. Z. Aslam, S. G. Kim, and S. R. Lee. "Machine learning- based defect prediction models for software quality enhancement." IEEE Software, vol. 34, no. 2, pp. 98-104,2018.
[6] P. K. Das, S. Jadhav, and A. N. Srivastava. "A review of machine learning approaches for software defect prediction." International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, no. 7, pp. 320-325, 2019.
[7] L. Zhang, W. Li, and D. Zhang. "A deep learning framework for software quality prediction." IEEE Access: Practical Innovations Open Solutions, vol. 6, pp. 15002-15009, 2018.
[8] F. B. Adam, N. A. Fazli, and L. M. N. Kamarudin. "Software quality prediction using machine learning algorithms: A comparative study." Journal of Computational Science, in the vol. 10.
[9] J. Singh, and K. S. Thakur. "A study of software quality prediction using machine learning algorithms." Journal of Computer Applications in Engineering, vol. 7, no. 1, pp. 112-121, 2019
[10] K. R. S. S. Ghosh and N. T. Singh. "Improving software quality prediction with advanced machine learning algorithms: A comprehensive study." International Journal of Software Engineering and Knowledge Engineering, vol. 32, no. 6, pp. 1195-1213, 2020.
[11] T. M. D. Nguyen, H. T. Nguyen, and C. H. Le. "A survey on the use of machine learning techniques for software defect prediction." Journal of Software Engineering and Applications, vol. 11, no. 4, pp. 154-167, 2021.
[12] J. B. Figueroa, M. Alvarado, and A. S. L. Garza. "Predicting software defects using ensemble learning methods." International Journal of Computer Science and Information Security, vol. 18, no. 7, pp. 80-90, 2020.
[13] S. P. Patel, A. Tiwari, and P. K. Sharma. "Application of machine learning techniques in software defect prediction." International Journal of Computer Applications, vol. 184, no. 8, pp. 12-21, 2021.
[14] R. K. Gupta, S. S. Pandey, and P. K. Saha. "Application of machine learning for software defect prediction: A comprehensive review." Journal of Software: Evolution and Process, vol. 31, no. 7, pp. 200-210, 2020.
[15] A. S. Bansal, R. Kumar, and M. Kumar. "Defect prediction models using machine learning algorithms: A comparative study." In 2019 International Conference on Computer Applications (ICCA), pp. 75-79, 2019.
[16] S. G. Soni, A. K. Verma, and D. Yadav. "Improving software defect prediction using hybrid machine learning techniques." Journal of Software Engineering and Technology.
[17] A. K. Shukla, R. K. Agarwal, and A. R. Sharma. "Machine learning algorithms for defect prediction in software systems: A review and comparative analysis." Software Quality Journal, vol. 28, no. 4, pp. 1059-1075, 2019.
[18] B. Y. Goh, C. M. Tan, and S. S. Lee. "Predicting software quality using machine learning algorithms: A survey of techniques." International Journal of Software Engineering and Applications, vol. 11, no. 2, pp. 98-110, 2020.
[19] D. K. Dubey, P. S. R. D. Mishra, and H. S. S. Nair. "Comparison of machine learning techniques for software defect prediction using real-world datasets." International Journal of Software Engineering and Knowledge Engineering, vol. 33, no. 1, pp. 42-54, 2021.
[20] P. J. M. Rios, C. J. J. Lee, and K. A. Smith. "Defect prediction in software using machine learning algorithms: A case study." Journal of Software: Theory and Practice, vol. 45, no. 6, pp. 707-717, 2021.
[21] T. S. V. Murali, P. K. Joshi, and D. K. Roy. "A hybrid machine learning approach for software defect prediction." In 2020 International Conference on Software Engineering and Applications (SEA), pp. 102-108, 2020.