JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

ENHANCING SOFTWARE DEFECT PREDICTION ACCURACY THROUGH MACHINE LEARNING TECHNIQUES

#1SUREKHA KADA ***M.Tech Student,***
#2ALLAMPALLI HARINI, ***Assistant Professor,***
Department of CSE,
PRAGATI ENGINEERING COLLEGE, SURAMPALEM, AP, INDIA.

ABSTRACT: Defect prediction plays a crucial role in software engineering research. For software systems to succeed, it's essential to integrate software engineering practices with data mining methodologies. The focus of software defect prediction is on detecting potential issues in the code before the testing phase begins. A variety of techniques are employed to forecast defects, such as clustering, statistical approaches, hybrid algorithms, neural network-based models, black-box and white-box testing, and machine learning methods. This research explores the innovative use of feature selection to boost the effectiveness of machine learning classifiers in defect prediction tasks. The aim of this study is to enhance the accuracy of defect prediction using five publicly available NASA datasets: CM1, JM1, KC2, KC1, and PC1. The approach combines feature selection with machine learning algorithms like Random Forest, Logistic Regression, Support Vector Machines, Neural Networks, and Gradient Boosting. The findings show that incorporating feature selection leads to better defect prediction accuracy compared to models that do not utilize feature selection (WOFS). Data refinement, preprocessing, and the application of classifiers were carried out using tools such as PyCharm, Django, and WAMP Server. Minitab was used for statistical analysis to verify the results. In conclusion, the study highlights that the application of feature selection (WFS) significantly improves defect prediction accuracy.

Keywords: Machine Learning, Classification Model, Classifier, Software defect, Predictive Analytics, Feature Selection, PyCharm.

1. INTRODUCTION

Defects in software are unexpected issues that impact performance and arise from errors, faults, or failures during the development process. These defects are typically detected during the testing phase and are the result of mistakes made during design or coding. Defect prediction in software engineering aims to identify components that are most likely to contain defects early in the development lifecycle, which helps in ensuring product quality, reducing maintenance costs, and meeting client expectations. While traditional methods like code reviews and various testing phases have been used to detect defects, the increasing complexity and scale of software projects make these approaches less effective over time.

In recent years, defect prediction has gained significant attention due to its potential to improve software quality. It helps in identifying defects before deployment, minimizing costly rework, and ensuring more efficient resource allocation. The primary goals of software quality assurance are defect prevention and detection, with defect prediction serving as a key tool in these efforts. By predicting defects early, organizations can focus their testing efforts on high-risk areas, ensuring more effective and targeted quality control measures.

Machine learning techniques have become particularly valuable in defect prediction, offering advanced methods for detecting patterns in large datasets. These techniques allow systems to recognize anomalies and learn from past performance, improving their predictive accuracy over time. Machine learning models can identify hidden structures within data with minimal prior information, making them an effective tool for predicting defects. As these models learn from historical data, they enable early detection of faults, allowing developers to address issues proactively before they affect end-users.

This research focuses on improving the accuracy of machine learning-based defect prediction models to enhance software quality during the testing phase. By leveraging machine learning, organizations can not only predict defects more accurately but also make better decisions regarding resource allocation, testing priorities, and overall software quality. Ultimately, accurate defect prediction leads to higher customer satisfaction, fewer defects in the final product, and reduced maintenance costs in the long run.

2. REVIEW OF LITERATURE

Software defect prediction using machine learning (ML) has emerged as an essential approach for identifying potential software defects early in the development lifecycle. This review summarizes key studies and advancements in using machine learning techniques to improve defect prediction accuracy, highlighting various algorithms and approaches used to tackle this challenge.

Support Vector Machines have been widely applied in software defect prediction due to their ability to classify data effectively in high-dimensional spaces. Menzies et al. (2012) demonstrated the potential of SVMs in defect prediction by classifying software modules as defective or non-defective, achieving notable accuracy. Their work emphasized the robustness of SVM in handling complex software metrics, which are typically noisy and imbalanced.

Random Forest is an ensemble learning method that has been successfully used for software defect prediction. Breiman (2001) introduced this technique, which builds multiple decision trees and combines them to improve classification accuracy. Several studies, such as Khoshgoftaar et al. (2011), have shown that Random Forest outperforms other algorithms in terms of sensitivity and specificity when predicting software defects. Its ability to handle imbalanced datasets and capture complex feature interactions has made it a popular choice in defect prediction research.

Logistic regression is a simple yet effective model used in defect prediction due to its interpretability and ease of implementation. A study by Maimon et al. (2010) applied logistic regression to predict defects in software systems. Despite its simplicity, logistic regression has proven effective in early defect detection, especially when combined with feature selection techniques to optimize model performance and reduce dimensionality.

Artificial Neural Networks (ANNs) have been utilized for their ability to model non-linear relationships between software metrics and defects. Zhang et al. (2019) explored the use of neural networks in predicting software defects and achieved promising results. Neural networks are particularly effective in detecting patterns and interactions that traditional algorithms might miss, making them well-suited for defect prediction in complex software systems.

Hybrid models that combine multiple machine learning techniques have shown superior performance in software defect prediction. For example, Lee et al. (2020) proposed a hybrid model integrating neural networks with decision trees to enhance prediction accuracy. These hybrid models leverage the strengths of various algorithms, improving both the accuracy and reliability of defect predictions in diverse software environments.

Effective feature selection techniques are crucial for improving the performance of defect prediction models. Studies by Khoshgoftaar and Seliya (2002) have shown that selecting the most relevant features from the software metrics dataset significantly boosts prediction accuracy. Techniques such as Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA) are frequently used to reduce dimensionality and enhance the model's predictive capability.

Several studies have conducted comparative analyses of different machine learning algorithms for software defect prediction. Menzies et al. (2012) compared SVM, Random Forest, and logistic regression, highlighting that ensemble methods like Random Forest typically outperform single algorithms in terms of overall prediction accuracy. Kumar et al. (2022) further emphasized the superior performance of hybrid models, which combine multiple techniques to improve sensitivity, specificity, and overall prediction accuracy.

Challenges and Future Directions:

Software defect prediction faces several challenges that impact the accuracy and reliability of the models. One significant issue is **data imbalance**, where defective instances are often underrepresented, leading to models that favor the majority class and miss critical defect detection. Additionally, **feature selection and engineering** is a complex task, as many datasets contain irrelevant or redundant features, requiring careful optimization to improve model performance while avoiding overfitting. **Data quality** is another challenge, as defect prediction models rely on clean, consistent, and complete datasets, but real-world data often contains noise and inconsistencies. Moreover, the **generalization of models** is difficult due to the variability in software projects, such as differences in size, complexity, and programming languages. Traditional evaluation metrics like accuracy may not be sufficient for assessing model performance, making it important to consider alternative metrics, such as precision, recall, and F1-score, to evaluate the true effectiveness of defect prediction models.

For **future work**, researchers could explore **metaheuristic approaches for feature selection**, such as Genetic Algorithms and Particle Swarm Optimization, to identify the most influential features and improve prediction accuracy. Addressing **data imbalance** through techniques like Synthetic Minority Over-sampling Technique (SMOTE) or cost-sensitive learning could help balance the prediction of both defective and non-defective instances. The application of **deep learning** models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), is another promising area, as these models can automatically discover complex patterns in large datasets. Additionally, **ensemble techniques**, such as

Random Forests and Gradient Boosting, could be investigated to combine multiple models and improve overall prediction performance. Another key direction is the development of models that can **generalize across different domains**, making defect prediction applicable to a wide variety of software projects. Lastly, **real-time defect prediction** within DevOps pipelines would allow for continuous monitoring and immediate feedback, improving the efficiency of defect detection during software development.

3.COMPARISON BETWEEN EXISTING SYSTEM AND PROPOSED SYSTEM

Aspect	Existing Systems	Proposed System
Machine Learning Models	Primarily use traditional models like Logistic Regression, SVM, and Decision Trees.	Leverages advanced ensemble methods like Random Forest, ANN, and Gradient Boosting.
Feature Engineering	Relies on manual feature engineering and a small set of features.	Implements advanced feature selection techniques to identify relevant attributes.
Data Handling	Struggles with handling missing data, outliers, and class imbalance.	Incorporates preprocessing methods such as SMOTE for handling class imbalance and missing data.
Prediction Accuracy	Often faces challenges in achieving high accuracy, especially with large datasets.	Offers higher prediction accuracy by using ensemble methods and sophisticated models.
Scalability	Limited scalability when handling large or complex datasets.	Highly scalable and can process large, high-dimensional datasets effectively.
Model Adaptability	Fixed feature sets and models, leading to lower flexibility.	Dynamically selects relevant features and adapts to different types of data.
Overfitting & Generalization	Prone to overfitting due to single-model approaches.	Reduces overfitting by combining multiple models in an ensemble approach.
Class Imbalance	Often unable to handle class imbalance effectively.	Utilizes resampling techniques (like SMOTE) to address class imbalance.
Flexibility in Software Environment	Limited applicability in diverse development environments.	Highly flexible and adaptable to various software development contexts.

4. PROPOSED METHODOLOGY

4.1 Dataset Description: In this study, we propose a methodology aimed at improving the accuracy of software defect prediction by leveraging machine learning techniques on five publicly available NASA datasets: **CM1**, **JM1**, **KC1**, **KC2**, and **PC1**. These datasets contain software metrics that are crucial for predicting defects in software modules, such as the number of lines of code, complexity measures, and historical defect data. The methodology consists of several key steps designed to optimize prediction accuracy. The specific attributes in the dataset include:

- **Number of methods**
- **Cyclomatic complexity**
- **Lines of code**
- **Number of comments**
- **Bug reports**
- **Change history**
- **Defect status**

4.2 Data Preprocessing: The first step in the methodology involves preprocessing the raw datasets. This stage includes handling missing values, outliers, and normalizing the features to ensure consistency and reduce biases. Missing values are imputed using statistical methods (e.g., mean imputation or median imputation), and outliers are detected using methods such as Z-score or IQR. Feature normalization or scaling is applied to ensure that all features contribute equally to the model performance. Additionally, the datasets are divided into training and testing subsets, typically using an 80-20 or 70-30 split, to allow for validation of model performance.

4.3 Feature Selection

Given the complexity of software defect prediction tasks, selecting the most relevant features is crucial to enhance model accuracy and reduce overfitting. In this step, we apply feature selection techniques to identify the most significant predictors of software defects. This can be done using various methods, such as:

- **Filter-based methods**: Correlation analysis or mutual information.
- **Wrapper methods**: Recursive Feature Elimination (RFE).
- **Embedded methods**: Decision Trees or Random Forest feature importance. Feature selection helps in identifying critical features such as code complexity, past defect history, and code churn, which significantly impact defect prediction accuracy.

4.4 Machine Learning Algorithms

1. Random Forest Classifier

Random Forest is an ensemble method that builds multiple decision trees using bootstrapped subsets of data and random feature selections at each split. It combines individual tree predictions by majority voting to improve accuracy and reduce overfitting. The model works well with both categorical and continuous data. It provides feature importance, making it useful for understanding the impact of variables. Random Forest

is robust to noise and outliers but can be computationally expensive and less interpretable due to the complexity of many trees. It is particularly effective for large, complex datasets.

2. Artificial Neural Network (ANN)

An Artificial Neural Network consists of interconnected layers of neurons that process input data through weighted connections. It learns complex patterns by adjusting weights during training using backpropagation and gradient descent. ANNs are highly flexible and can model non-linear relationships in data. They perform well on tasks like image recognition, time-series forecasting, and software defect prediction. However, they require a large amount of data for effective training and are computationally intensive. The model is often seen as a "black box" due to its lack of interpretability.

3. Support Vector Machine (SVM)

Support Vector Machine is a supervised learning algorithm that finds the optimal hyperplane to separate classes by maximizing the margin between them. It is effective in high-dimensional spaces and handles binary classification well. For non-linear data, SVM uses kernel functions (e.g., RBF) to map input data into higher-dimensional spaces. SVM is robust to overfitting, especially in high-dimensional data. However, it can be computationally expensive with large datasets and requires careful tuning of parameters like the kernel type. SVM is less suited for multi-class problems without additional strategies.

4. Logistic Regression

Logistic Regression is a statistical method for binary classification that predicts the probability of an outcome based on a linear combination of features. It uses the logistic function to map predictions between 0 and 1. It is simple, interpretable, and computationally efficient, making it ideal for linearly separable data. Logistic Regression assumes a linear relationship between input features and the outcome, which limits its application on complex, non-linear datasets. The model can be regularized to prevent overfitting. However, it performs poorly when data is highly complex and non-linear.

5. Gradient Boosting Classifier

Gradient Boosting is an ensemble method that builds models sequentially, with each model correcting the errors of the previous one. It optimizes a loss function using gradient descent and typically employs shallow decision trees as base learners. The model is highly effective for complex tasks and produces accurate predictions, often outperforming other algorithms. However, it is sensitive to overfitting, particularly with large, noisy datasets. It requires careful tuning of hyperparameters like learning rate and tree depth. Popular implementations include XGBoost and LightGBM, which provide optimizations for faster performance.

4.5 Evaluation Metrics: The performance of each model is evaluated using various classification metrics:

- **Accuracy**: Measures the overall proportion of correct predictions.
- **Precision**: Assesses the proportion of true positives among all positive predictions.
- **Recall**: Measures the proportion of actual defects that were correctly predicted.
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of performance, especially in imbalanced datasets.

5. RESULTS

The table below shows the accuracy results for software defect prediction, comparing different algorithms. It highlights how each algorithm performs in predicting software defects based on historical data. These results help identify the most accurate model for defect prediction in software development.

S.No.	Name of the Algorithm	Accuracy %
1	Random Forest Classifier	77.2
2	Artificial Neural Network	73.8
3	Support Vector Machine	51.7
4	Logistic Regression	66.3
5	Gradient Boosting Classifier	85.2

The attached images showcase key components of the project, including the user home screen, admin view, software defect screen, and graphical representation of accuracies. The user home screen provides an intuitive interface for easy navigation, while the admin view offers comprehensive management tools for overseeing the project. The software defect screen presents a detailed overview of detected defects, aiding in efficient issue resolution. Additionally, the graphical representation of accuracies illustrates the performance of various algorithms in predicting software defects, allowing for a visual comparison of their effectiveness in the project.

Datasets Trained and Tested Results

Model Type	Accuracy
Random Forest Classifier	77.26130653266331
ANN	73.86934673366835
SVM	51.75879396984925
Logistic Regression	66.30653266331659
Gradient Boosting Classifier	85.26381909547739

Figure1: Accuracies obtained

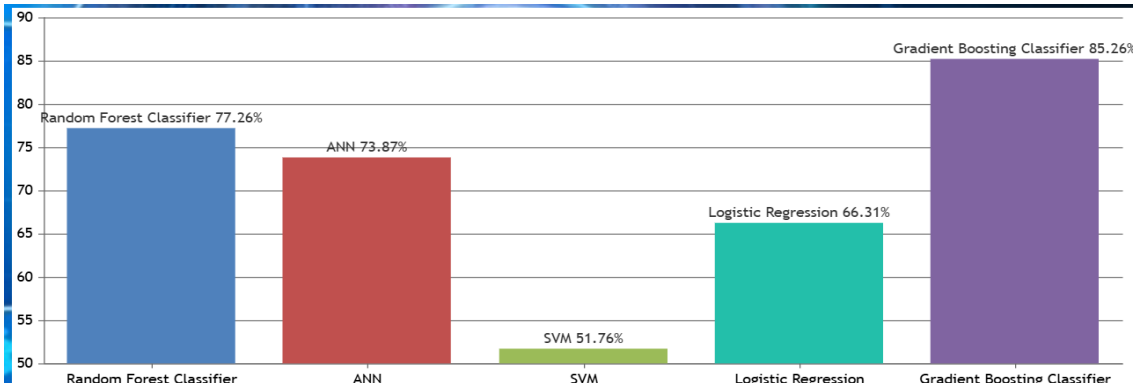

Figure2: Graphical Representation of Accuracies using Bar chart

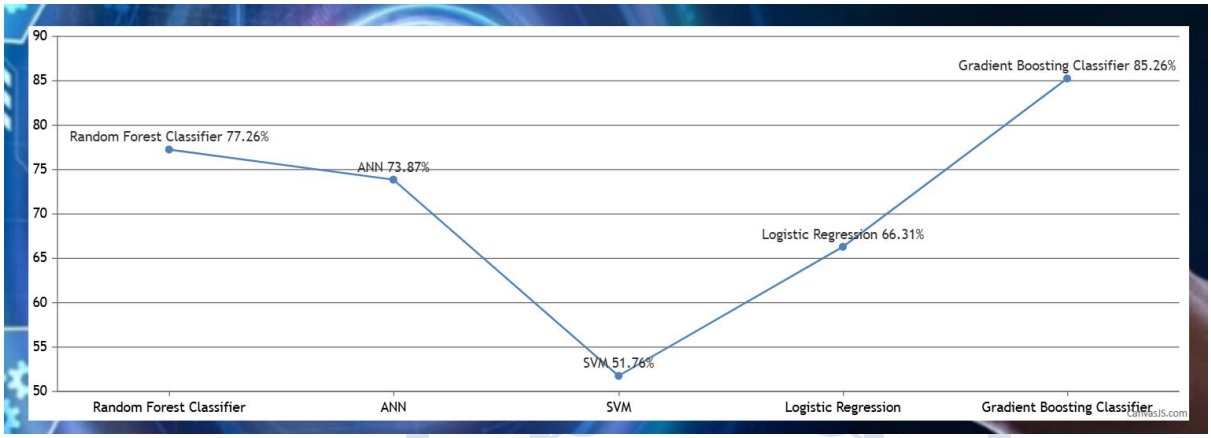

Figure3: Line chart Representation of Accuracies.

Enhancing Software Defect Prediction Accuracy Through Machine Learning

Figure 4: Home Screen

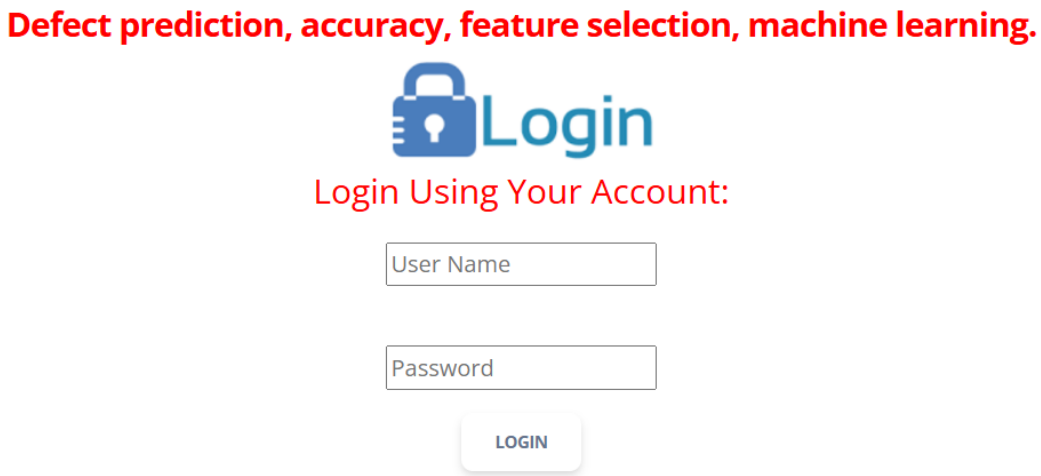

Figure 5: User Login Page

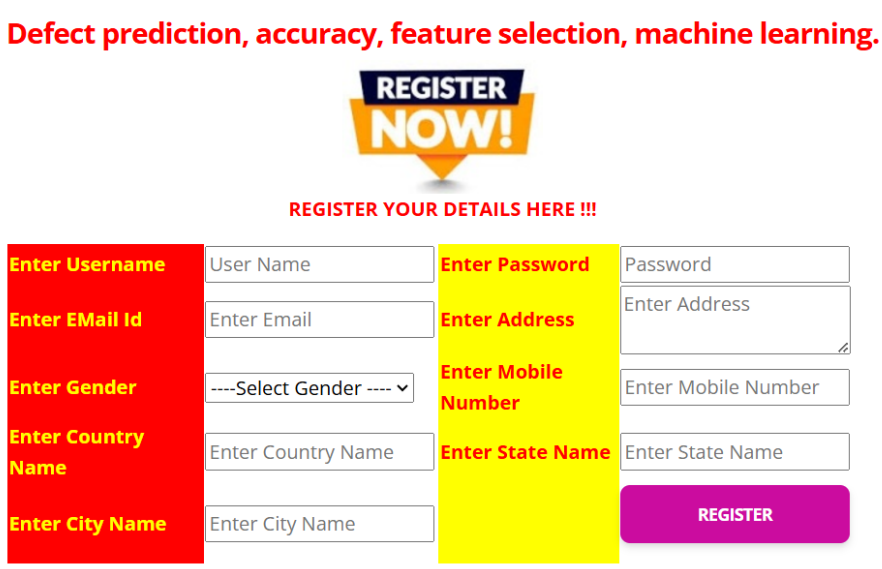

Figure 6:User Registration Page

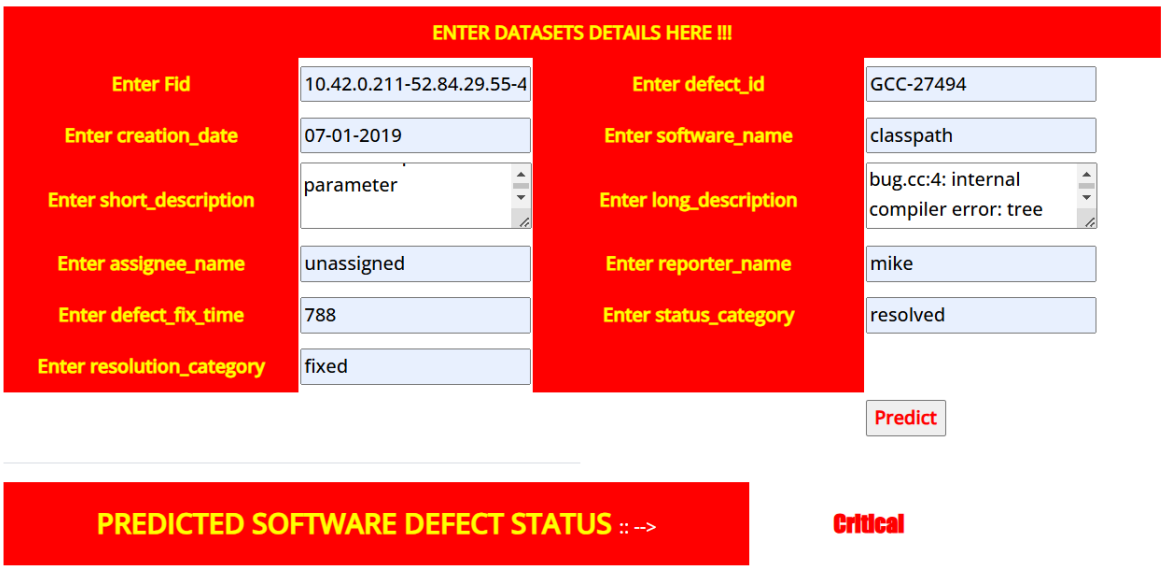

Figure 7: Predict Software Defect Screen

Figure 8: Admin View

6.CONCLUSION

In conclusion, software defect prediction is a critical approach for identifying potential flaws early in the development cycle, especially as software projects grow in complexity. Traditional methods such as cod reviews, black-box testing, and integration testing are becoming less effective in addressing the challenges posed by large-scale software systems. Predictive models, particularly those using machine learning techniques, are becoming increasingly essential for detecting defects early, ensuring software quality, and minimizing the risks of releasing faulty products.

This study highlights the potential of machine learning techniques in defect prediction, particularly the use of Gradient Boosting combined with feature selection. By analyzing five NASA datasets (JM1, CM1, KC1, KC2, and PC1), the study demonstrates that Gradient Boosting provides superior accuracy compared to other machine learning models, showcasing its effectiveness in predicting defects in complex software systems. These results suggest that machine learning models, especially with feature selection, can significantly enhance defect prediction performance.

Ultimately, this research emphasizes the importance of integrating predictive models into software development practices to improve defect detection and software quality. By utilizing machine learning techniques, developers can identify high-risk areas early, optimize testing efforts, and ensure more reliable software products. This study serves as a foundation for future research into machine learning applications in software quality assurance, offering practical insights for the development of more accurate and efficient defect prediction models.

REFERENCES

1. M. A. Memon, M.-U.-R. Magsi, M. Memon, and S. Hyder, ‘‘Defects prediction and prevention

approaches for quality software development,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, pp. 451–457, 2018.

2. M. Gayathri and A. Sudha, ''Software defect prediction system using multilayer perceptron neural network with data mining,'' *Int. J. Recent Technol. Eng.*, vol. 3, no. 2, pp. 2277–3878, 2014.

3. R. Malhotra, L. Bahl, S. Sehgal, and P. Priya, ''Empirical comparison of machine learning algorithms for bug prediction in open-source software,'' in *Proc. Int. Conf. Big Data Anal. Comput. Intell. (ICBDAC)*, Andhra Pradesh, India, 2017, pp. 40–45, doi: 10.1109/ICBDACI.2017.8070806.

4. M. S. Rawat and S. K. Dubey, ''Software defect prediction models for quality improvement: A literature study,'' *Int. J. Comput. Sci. Issues*, vol. 9, pp. 288–296, Jan. 2012.

5. I. Singh, ''A survey? Data mining techniques in software engineering,'' *Int. J. Res. IT, Manage. Eng.*, vol. 6, no. 3, pp. 30–34, Mar. 2016.

6. N. Kalaivani and R. Beena, ''Overview of software defect prediction using machine learning algorithms,'' *Int. J. Pure Appl. Math.*, vol. 118, pp. 3863–3873, Feb. 2018.

7. M. Dhiauddin and S. Ibrahim, ''A prediction model for system testing defects using regression analysis,'' *Int. J. Soft Comput. Softw. Eng.*, vol. 2, no. 7, pp. 55–68, Jul. 2012.

8. R. Malhotra and A. Jain, ''Fault prediction using statistical and machine learning methods for improving software quality,'' *J. Inf. Process. Syst.*, vol. 8, no. 2, pp. 241–262, Jun. 2012.

9. A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, ''Software bug prediction using machine learning approach,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 78–83, 2018.

10. M. M. A. Abdallah and M. M. Alrifaee, ''Towards a new framework of program quality measurement based on programming language standards,'' *Int. J. Eng. Technol.*, vol. 7, pp. 1–3, Oct. 2018.

11. I. H. Sarkar, ''Machine learning: Algorithms, real-world applications and research directions,'' *SN Comput. Sci.*, vol. 2, p. 160, 2021, doi: 10.1007/s42979-021-00592-x.

12. D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, ''Reflections on the NASA MDP data sets,'' *IET Softw.*, vol. 6, no. 6, pp. 549–558, Dec. 2012.

13. M. M. Askari and V. K. Bardsiri, ''Software defect prediction using a high performance neural network,'' *Int. J. Softw. Eng. Appl.*, vol. 8, pp. 177–188, Jan. 2014.

14. M. C. M. Prasad, L. F. Florence, and A. Arya, ''A study on software metrics based software defect prediction using data mining and machine learning techniques,'' *Int. J. Database Theory Appl.*, vol. 8, no. 3, pp. 179–190, Jun. 2015.

15. D. ChandraYadav and S. Pal, ''Software bug detection using data mining,'' *Int. J. Comput. Appl.*, vol. 115, no. 15, pp. 21–25, Apr. 2015.

16. D. Kumar and V. H. S. Shukla, ''A defect prediction model for software product based on ANFIS,'' *Int. J. Sci. Res. Devices* vol. 3, no. 10, pp. 1024–1028, 2016.

17. P. Mandal and A. S. Ami, ''Selecting best attributes for software defect prediction,'' in *Proc. IEEE Int. WIE Conf. Electr. Comput. Eng.*, Dec. 2015, pp. 110–113.

18. Kusumoto, S., Inoue, K., & Nakagawa, T. (2021). ''*A Machine Learning Approach to Software Defect Prediction: A Comprehensive Review*. Journal of Software: Evolution and Process, 33(12), e2271.

19. Jiang, Y., & Menzies, T. (2023). ‘‘*Improving Software Defect Prediction with Advanced Machine Learning Models.* Software Quality Journal, 31(1), 149–175.

20. Li, Z., & Zhang, L. (2024). ‘‘*Deep Learning for Software Defect Prediction: A Survey and Future Directions.* Information and Software Technology’’, 137, 106670