JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(1), MARCH, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Feature Selection and Machine Learning Based Software Defect Prediction: A Performance Evaluation

[1]Abdulrazak Muhammad, [2]Yahaya Isah Shehu
[1]*Department of computer science Sokoto State University, Sokoto*
[2]*Department of computer science Shehu Shagari University of education, Sokoto*

ABSTRACT

Software Defect Prediction (SDP) is a vital technique for enhancing software quality and reducing development and maintenance costs by identifying fault-prone components at early stages of the software life cycle. Despite significant progress in machine learning–based approaches, limited attention has been given to understanding which software features most strongly influence prediction performance and how different algorithms compare under standardized evaluation. This study investigates feature identification and evaluates the performance of several machine learning classifiers for SDP using the JM1 dataset from the NASA PROMISE repository, which contains 10,885 software modules described by McCabe and Halstead complexity metrics. Data preprocessing was performed using correlation-based feature selection and K-means clustering to reduce redundancy, mitigate multicollinearity, and improve data representation. Four classifiers Support Vector Machine, Naïve Bayes, Random Forest, and AdaBoost together with a stacking ensemble model were implemented and evaluated using accuracy, precision, recall, and F1-score, both with and without Particle Swarm Optimization. Experimental results indicate that the Random Forest classifier consistently outperformed the other models, achieving up to 99.91% accuracy and superior precision, recall, and F1-score, while Naïve Bayes produced the lowest performance. The ensemble approach, particularly the combination of Random Forest and AdaBoost, further improved robustness. Feature analysis revealed that line count of code, cyclomatic complexity, essential complexity, total operators and operands, program volume, and program length are the most influential predictors of software defects. These findings demonstrate that careful feature selection combined with robust ensemble-based learning models can significantly improve the reliability and effectiveness of software defect prediction systems.

ARTICLE INFO
Article History
Received: December, 2025
Received in revised form: January, 2026
Accepted: February, 2026
Published online: March, 2026

KEYWORDS
Software Defect Prediction, Machine Learning; Feature Selection, Random Forest, Ensemble Learning, Software Metrics

INTRODUCTION

Meeting the end user's requirement specifications or expectations during the first initial stage of software deployment is one of the most difficult things to achieve in software development process. Therefore, detecting defects before the deployment of software helps in delivering high quality software products and minimizes development costs. Software defect prediction research endeavors to forecast components susceptible to be defective prior to the testing phase in software development. This proactive approach aids in identifying and rectifying potential issues early on, ultimately enhancing software quality while minimizing costs and development time.

A software defect is an error, flaw, or mistake originating from the development team

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(1), MARCH, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

that hinders the smooth operation of the software (M. S. G. M. K. L. K.Prashanthi, 2023). This directly impacts the software quality, which reflects how smooth and reliable the software functions. Quality in software development is crucial, with organizations striving to deliver bug-free products to meet customer requirements and stay competitive (Aimen Khalid, et al; 2023). Software defect prediction (SDP) is a pivotal technique in software development, aiming to enhance

Software quality and minimize testing expenses by leveraging diverse machine learning methods for creating predictive models. It is crucial for companies across various sectors to anticipate software issues to uphold quality standards, ensure customer satisfaction, and economize on testing efforts. SDP integrates into the software development life cycle, utilizing Machine Learning (ML) algorithms with historical data to forecast faults and defects, thereby streamlining the development process and optimizing software reliability. By employing SDP, organizations can proactively address potential software glitches, leading to improved software quality, reduced costs, and enhanced customer experiences (Pandit, Mahesha and Varma, Nitin, 2019).

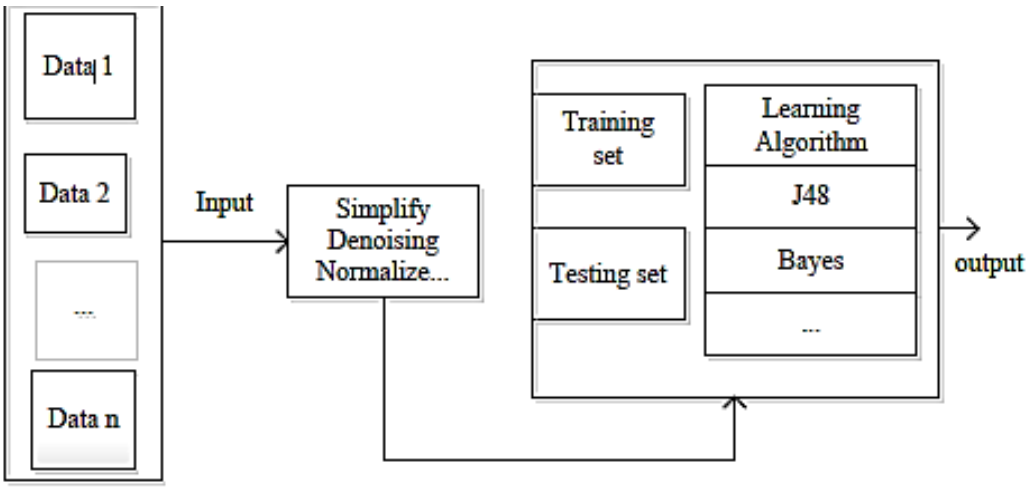

Figure 1.1 Software defects prediction model.

The primary goal of Software Defect Prediction (SDP) is to ensure high-quality software and reliability while optimizing resource utilization. By accurately predicting defects in software, organizations can enhance software quality, satisfy customers, and reduce testing costs. SDP achieves this by developing classification models through various machine learning techniques, such as support vector machine (SVM), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), and Artificial Neural Network (ANN) (I. C SOCIETY, 2024). These models help prioritize resource allocation at different stages of software development, enabling efficient testing and maintenance processes. SDP's evolution from statistical models to sophisticated machine learning approaches reflects its commitment to improving software quality and streamlining development activities.

Classification algorithms in machine learning are type of supervised learning technique used to identify the category or class of new observations based on a given set of training data. Essentially, these algorithms teach a computer to sort new information into predefined groups. The program learns from a dataset and then classifies new observations into various categories, such as "Yes" or "No," "0" or "1," "Spam" or "Not Spam," and so on. These categories are also referred to

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(1), MARCH, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

as targets, labels, or classes. Now, let's explore some of the algorithms that enable this process.

Naïve Bayes (NB): The Naive Bayes (NB) algorithm is a classification technique rooted in Bayes' theorem. It functions by calculating and comparing probabilities. The NB algorithm is particularly effective when the features in a dataset are independent meaning there is no correlation between the features. Logistic Regression: A supervised machine learning algorithm for classification that predicts categorical outcomes by fitting an "S"-shaped logistic function, providing probabilistic outputs between 0 and 1. It is ideal for binary classifications like "true or false" or "spam versus not spam."

Decision Trees: A classification method using a tree structure where internal nodes represent data features, branches represent decisions, and leaves are the outcomes. It offers a visual, flowchart-like representation, simplifying complex decision-making processes. Random Forests: An ensemble learning technique that improves accuracy by combining multiple decision trees. It makes final predictions based on the majority vote from all trees, reducing overfitting and enhancing performance. Support Vector Machines (SVM): A powerful algorithm used for classification, regression, and outlier detection. SVMs find the optimal hyperplane to separate data into classes, excelling in high-dimensional and complex datasets, such as text or image recognition tasks.

Software quality is a multifaceted concept that includes various perspectives. Fundamentally, it is defined by how well a software product aligns with its requirements and fulfills user needs. It encompasses both the characteristics of the software product and the processes employed in its development. From the product perspective, quality is measured by attributes such as reliability, usability, performance, and security. From the process perspective, it is associated with the application of structured methods and best practices that contribute to producing a higher-quality product (I.C SOCIETY, 2024).

Software bugs continue to pose significant risks across various critical domains, with real-world cases underscoring potential severe consequences of software failures when quality assurance measures are inadequate. Software quality has become a critical concern in the software engineering lifecycle due to the increasing scale and complexity of modern software. It requires huge investment and resources to come up with software system. The financial implication of software quality is very high. In year, 2018, the world invested about 4 trillion USD on IT and Telecom system. In USA alone about 2.84 trillion USD was spent purposely for poor software quality. However, approximately 37% of the cost of poor software quality was based on software lapses. Moreover, nearly 17% of the expenses was also on finding and fixing software defects (Mehmood Iqra, et al; 2023).

Despite advancements in software defect prediction (SDP) using machine learning, there is insufficient understanding of which software features most significantly influence the performance of machine learning algorithms in software defect prediction. Therefore, there is a need to applies standardized evaluation metrics to analyze the machine learning algorithms, and develops robust, interpretable, and transferable models capable of performing reliably across different software systems and contexts.

Advancements in machine learning—particularly deep learning, ensemble methods, and transfer learning have significantly enhanced the effectiveness of software defect prediction (SDP) models. These developments help address persistent challenges such as class imbalance, feature selection, and model interpretability, thereby opening new avenues for research and practical application. This literature review examines studies, highlighting key trends, techniques, and findings in the field. Its aim is to provide a comprehensive overview of current approaches, evaluate their strengths and limitations, and suggest potential directions for future research.

Emmanuel, Gbenga, et al (2021) designed Automatic software bug prediction using adaptive golden eagle optimizer with deep

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

learning. The study analyzed five NASA dataset such as CM1, JM1, KC1, and PC1. The acronyms CM1, JM1, KC1, and PC1 most commonly refer to specific software defect prediction datasets from the NASA Metrics Data Program (MDP), widely used in software engineering research. Moreover, CM1: Data from a NASA spacecraft instrument written in the C programming language. JM1: Data from a real-time predictive ground system (using simulations to generate predictions) written in C. KC1: Data from a C++ system implementing storage management for receiving and processing ground data. PC1: Data from C functions in the flight software for an earth-orbiting satellite.

They are names assigned to data collected from different real-world software systems. The study also employed adaptive golden eagle optimizer (AGEO) for feature selection and long short-term memory (LSTM) based recurrent neural network (RNN) as the methodology to remove duplicate data instances from the dataset, which was conducted on JAVA platform environment. The output of the study revealed that, AGEO and LSTM improve bug detection accuracy in software development. However, the maximum bug detection accuracy for promise dataset stood at 80.1% and that of NASA stood at 85.5% which are not up to some bug detection accuracy.

Sharma, et al (2020) designed ensemble machine learning model for software defect prediction. The study employed ensemble machine learning with K-nearest neighbor (KNN), Generalized linear Discriminant Analysis (LDA) with random forest as based learner. The cost-sensitive Siamese parallel fully connected neural network technique was used in the study. The study utilized four NASA datasets: CM1, JM1, KC3, and PC3 from promise repository using Rstudio simulation tool. The proposed ensemble model attained accuracy of 87.69% for CM1, 81.11% of JM1, 90.70% of PC3 and 94.74% of KC3 respectively. The model outperformed seven states of art prediction models with an average prediction accuracy of 88.56%. However, LDA is sensitive to outliers and Multicollinearity and LDA values occasional fall below 0 or exceed 1. Moreover, there is need to correlation coefficient matrix to solve the problems of Multicollinearity.

METHODOLOGY

The significance of software defect prediction stems from its ability to proactively manage software quality. By properly forecasting defects, developers may better allocate resources, prioritize testing efforts, and focus on high-risk areas, minimizing the time and expense of post-release maintenance and bug repairs. This proactive strategy not only improves the dependability and robustness of software products, but it also promotes customer satisfaction by providing more reliable code.

In this research, we use the JM1 dataset from the NASA Promise repository, which is widely used in the software engineering community to assess software defect prediction methods. The JM1 dataset, derived from a NASA spacecraft sensor, contains 10,885 modules and 22 attributes, including McCabe and Halstead metrics. These measurements provide useful insights into the code's structure and complexity elements, making the dataset an excellent candidate for testing machine learning models in defect prediction.

To solve the problem of software defect prediction, we use a combination of data preparation techniques, feature selection approaches, and machine learning algorithms. The approaches are intended to handle the dataset's complexities, such as scaling and encoding, while guaranteeing that the chosen features contribute favorably to the model's performance. This project intends to increase the accuracy and reliability of defect prediction models using Support Vector Machine (SVM), Naïve Bayes (NB), and Random Forest (RF) classifiers, Ada Boost (ADT), as well as an ensemble technique. This will contribute to the improvement of software quality assurance practices.

This research was carried out through the following procedures below:

1. Data gathering
2. Data preprocessing
3. Feature selection
4. Model design

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(1), MARCH, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

5. Experimental results discussions
6. Performance evaluation

This research utilized method by (Aimen Khalid, et al; 2023) in their study software defect prediction analysis using machine learning techniques. However, the method has been updated by running two classification models simultaneously to enhance the performance of the models.

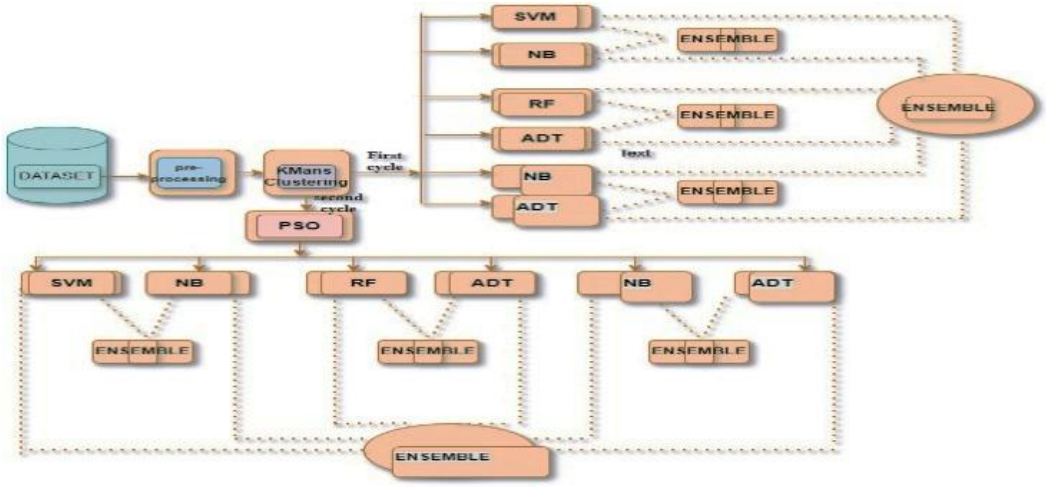

Fig 3.1 Methodology diagram

Data Gathering

The JM1 dataset from NASA Promise software repository was used in the research. JM1 is a C base NASA space craft instrument which is freely available and part of NASA metric Data program. The dataset consists of 10885 rows or modules and 22 columns or properties. The dataset for this study includes four McCabe metrics, twelve Halstead measurements, and additional metrics. McCabe metrics are method-level indicators that concentrate on programming principles and can be easily extracted from source code. Halstead metrics, on the other hand, are numerical and can be readily collected using various software tools (Aimen Khalid, et al; 2023). Table 2 provides a list of the dataset metrics.

Dataset Preprocessing

Preprocessing is a vital step in machine learning and data analysis, significantly affecting the quality and performance of predictive models. Raw data often contains noise, missing values, and inconsistencies, which can lead to inaccurate or biased outcomes. Techniques like data cleaning, normalization, and standardization are used to ensure each feature contributes appropriately, preventing any single feature from skewing the predictions. Additionally, steps such as feature selection and scaling techniques are crucial for extracting meaningful patterns and reducing computational complexity. Proper handling of categorical data and addressing class imbalances are also important for preventing overfitting and enhancing model robustness. Overall, preprocessing not only improves model accuracy but also contributes to more reliable and interpretable results, making it an essential component of the machine learning process (Kechao Wang, et al; 2020).

In this data was preprocessed through the following techniques: Label Encoding to convert the categorical variables in to numerical because machine learning models with numerical values, correlation feature selection this research used this method which is known as Correlation-based Feature Selection (CFS). It is specifically aimed at identifying and removing highly correlated features to reduce redundancy. The

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

assumption is that features or columns with a high correlation contain similar information, so removing one of them can help in reducing the multicollinearity and potentially improving model performance.

Clustering, K-means clustering is used to discover class labels, as well as inertia and the elbow approach to select the best number of clusters (K). In K-means clustering, the number of clusters created is denoted by the symbol K. Inertia is a metric for how effectively a data collection is clustered and is derived by squaring the distance between each data point and its centroid, then adding the squares throughout one cluster. The elbow approach makes use of the Within Cluster Sum of Squares idea (WCSS). In this experiment the value of K is 6 and the value of inertia is 750000.00 as indicated by the figures below:

Model Design

Classification supervised machine learning algorithm utilized data with output class labels. The dataset is divided into two parts: training and testing. The training dataset comprises 70% of the total, while the testing dataset comprises 30%. Initially, the training data includes output class labels, and the testing data consists of unseen data without output class labels. We employed four machine learning classification models for analysis: SVM (Linear SVC classifier), NB (Gaussian NB classifier), RF (Random Forest classifier) and ADT (Ada Boosting classifier). We used an ensemble method, stacking classifier, to combine the results of these three models, with NB as the base model and SVM, RF and ADT as member models. The RF model uses 1000 trees, and the random state for all classifiers is set to 42. The classifiers were analyzed both with and without Particle Swarm Optimization (PSO) techniques.

This project used four machine learning classifiers and ensemble method or approach based on the following facts below:Selecting classifiers like Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), (ADT) and an ensemble approach for software defect prediction can be justified based on various studies and research findings. These classifiers and approaches have demonstrated significant performance in terms of accuracy, robustness, and interpretability, making them suitable for this task, the selection of SVM, NB, RF, and ensemble approaches for software defect prediction is justified based on their respective strengths: SVM's effectiveness in high-dimensional spaces, NB's computational efficiency, RF's robustness and accuracy, and the enhanced performance provided by ensemble methods. These approaches have been validated in recent studies, proving their suitability for the task at hand.

Naive Bayes (NB) NB is a probabilistic classifier based on Bayes' theorem, which assumes independence among predictors. Despite this simplification, it performs well with large datasets and is efficient in terms of computation, which is beneficial for real-time defect prediction. According to study by (Suresh, P. and Kavitha, V, 2021) Naïve Bayes was found to be highly effective for software defect prediction due to its simplicity and speed, which are crucial for handling large datasets in software engineering.

Random Forest (RF) RF is an ensemble learning method that constructs multiple decision trees and merges their results to improve the predictive performance and control overfitting. This makes it robust and accurate for software defect prediction. A research conducted by (S.M, Rifat, et al; 2023) Demonstrated that RF outperformed other classifiers in terms of accuracy and stability, highlighting its suitability for defect prediction in various software projects.

Ada Boost (ADT) Ada Boost is a boosting technique that converts weak classifiers into a single strong one. It does this by iteratively training new models that focus on previous errors and then combining them through a weighted vote. Ensemble Technique Ensemble methods combine the predictions of multiple models to improve the overall performance and robustness of the prediction system. Techniques like bagging, boosting, and stacking can leverage the strengths of individual classifiers and mitigate their weaknesses. A study (K. Jan, 2024) indicated that ensemble approaches significantly improved the

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(1), MARCH, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

prediction accuracy and reliability of software defect detection models by combining different classifiers.

Support Vector Machine (SVM) SVM is effective in high-dimensional spaces and can handle large numbers of features, which is common in software defect datasets. It constructs hyperplanes in a multidimensional space to separate different classes, making it suitable for binary classification problems like defect prediction. A study by [12] showed that SVM performed well in predicting software defects, providing high accuracy and better generalization capabilities compared to other classifiers.

Experimental Setup

The classification experiment for machine learning models was conducted on HP laptop computer with windows11 with an Intel® Core™ i5-1035G1 processor, 12 GB of RAM, and 1 terabyte of secondary storage. The models were implemented in Jupyter Notebook via Anaconda 3 using the Python programming language. Python is widely used in predictive analytics and data science for handling both qualitative and quantitative data. In this experiment, Python libraries such as pandas, numpy, sklearn, seaborn, and matplotlib were utilized to design and evaluate the predictive machine learning models.

Evaluating the performance of classification models involves several metrics that help assess how well the model is performing in terms of predicting the correct class labels. Here are the most commonly used metrics for evaluating classification models. A confusion matrix is a table that summarizes the performance of a classification model by comparing its predicted outcomes with the actual outcomes. It provides a detailed breakdown of how well a model is performing in terms of correctly and incorrectly predicting each class.

Structure of a Confusion Matrix: For a binary classification problem (where there are only two possible classes, such as "positive" and "negative", True or False), a confusion matrix is a 2x2 table that looks like this:

Table 3.3 confusion matrix

	Predicted positive	**Predicted negative**
Actual Positive	True positive (TP)	False positive (FP)
Actual Negative	False positive (FN)	True negative (TN)

Explanation of the Terms:

1. True Positive (TP): The number of cases that are correctly predicted as positive (the model predicted "positive," and the actual label is also "positive").
2. False Positive (FP): The number of cases that are incorrectly predicted as positive (the model predicted "positive," but the actual label is "negative"). This is also known as a Type I error.
3. True Negative (TN): The number of cases that are correctly predicted as negative (the model predicted "negative," and the actual label is also "negative").
4. False Negative (FN): The number of cases that are incorrectly predicted as negative (the model predicted "negative," but the actual label is "positive"). This is also known as a Type II error.

RESULTS PRESENTATION

The output results of the conducted experiment without and with particle swarm optimization (PSO) is presented inform of tables and graphs: Evaluation of ML classifier without optimization

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 14(1), MARCH, 2026
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Table 3.4 Evaluation of performance classifiers without PSO for JM1

DATASET	EVALUATIONS MEASURES	SVM	NB	RF	ADT	ENSEMBLE
JMI	Accuracy	99.47	93.2	99.91	94	98.85
	Precision	99	93	100	94	99
	Recall	99	93	100	94	98.51
	F1_score	99	93	100	94	98.9

Table 3.5 Performance evaluation of Ensemble models

DATASET	EVALUATION METRICS	SVM+NB	RF+ADT	NB+ADT
JMI	Accuracy	94.00	97.00	92.00
	Precision	94.00	98.00	92.00
	Recall	94.00	97.00	92.00
	F1_score	94.00	97.00	92.00

Based on the results of the analysis, the research questions of the study can be answer as follows:

Q1. How effective are machine learning algorithms in predicting software defects in various software development processes? The experimental results showcase classification machine learning algorithms can be efficiency used in software defect prediction analysis more specifically Random Forest classifier.

Q2. Can different machine learning algorithms provide significantly different results in software defect prediction analysis? The output of the experiment results indicates that, obviously there is significant differences in the results of machine learning models used in the research. The RF has highest score in both evaluation metrics such as accuracy, precision, recall and F1 score. Meanwhile NB has lower scores in all evaluation metrics. The SVM is the second highest score meanwhile ADT is better than NB but slightly lower than SVM.

Q3. What are the most influential features for accurate software defect prediction using machine learning algorithms? Based on feature selection and correlation coefficient techniques, the most influential features or metrics are McCabe software metrics and some few Halstead metrics, which are: line count of code, cyclomatic complexity, essential complexity, Total operators of operand, volume and program length.

Generally, the experiment demonstrated that while all the ML models tested could achieve high levels of accuracy, the Random Forest classifier, particularly, provided the most reliable and accurate predictions for the JMI dataset.

Experimental Results Discussion

In this section, we present the detailed outcomes of the experiment conducted on the JMI dataset using Five distinct machine learning (ML) algorithms: Support Vector Machine (SVM), Naïve Bayes (NB), Random Forest (RF) AdaBoost (ADT), and an Ensemble technique. The results are systematically organized in tables and visualized through graphs to provide a clear comparison of the performance metrics of these algorithms, both with and without optimization.

Evaluation of Machine Learning Classifiers Without Optimization, the initial phase of the experiment involved evaluating the performance of each ML model without any optimization. The evaluation metrics considered include accuracy, precision, recall, and F-

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

measure. The results indicate that all models performed exceptionally well, with the Random Forest (RF) classifier emerging as the top performer.

Specifically, RF achieved 99.91% accuracy and 100% for precision, recall, and F-measure, demonstrating its robustness in handling the JMI dataset. The Support Vector Machine (SVM) achieved 99.47% accuracy and 99% for precision, recall, and F-measure, making it the second most efficient model. Furthermore, AdaBoost (ADT) also performed well, attaining 94% accuracy, precision, recall, and F-measure. Finally, the Gaussian Naïve Bayes (NB) classifier achieved 93.20% accuracy and 93% for precision, recall, and F-measure.

On the other hand, the results of the classification models on the CM1 dataset after the experiment indicate that the Random Forest (RF) model was the top performer, achieving the highest efficiency across all evaluation metrics 98.66% accuracy and 99.00% for precision, recall, and F1-score. Meanwhile, both the Support Vector Machine (SVM) and AdaBoost (ADT) models achieved identical performance, with 98.00% accuracy, precision, recall, and F1-score. The Naïve Bayes (NB) model performed moderately well, recording 87.00% for accuracy, precision, and recall, and 83.00% for F1-score. Furthermore, among the ensemble models, the combination of RF and ADT achieved the highest performance, scoring 99.00% across all evaluation metrics. The SVM+NB and NB+ADT ensemble models both attained 88.00% accuracy and recall, 80.00% precision, and 84.00% F1-score, respectively.

CONCLUSION

The results of the analysis of JMI Dataset indicate that machine learning algorithms, particularly the Random Forest classifier, are highly effective in predicting software defects. RF consistently achieved the highest scores in accuracy, precision, recall, and F1 score, making it the most robust model in this study. While SVM and the Ensemble method also showed strong performance, they appeared to be near-optimal without significant gains from optimization.

Moreover, AdaBoost relatively performed well. Naïve Bayes, however, exhibited limitations that optimization did not address, indicating it may not be suitable for this specific dataset or task. These findings demonstrate significant differences in the effectiveness of various machine learning algorithms for software defect prediction. Moreover, NASA software features such as: line count of code, Cyclomatic complexity, essential complexity, Total operators of operand, volume and program length. These are the features that influence machine learning models performance.

RECOMMENDATIONS

1. **Leverage Random Forest for Software Defect Prediction**: Due to its outstanding performance across key metrics such as accuracy, precision, recall, and F1 score, the Random Forest classifier is recommended as the primary algorithm for software defect prediction tasks.
2. **Software features**: such as metrics are McCabe software metrics and some few Halstead metrics, which are: line count of code, cyclomatic complexity, essential complexity, Total operators of operand, volume and program length are the most accurate features of NASA Dataset for software defect predictions analysis in machine learning.

FUTURE RESEARCH DIRECTION

In this paper, software defect prediction analysis was conducted using machine learning algorithms. four machine learning classification models were analyzed both without and with Particle Swarm Optimization (PSO), as well as with an ensemble method to detect defects in software. These models were applied exclusively to the JM1and CM1 dataset. However, further research is needed using multiple datasets from the NASA Promise software repository and advance algorithms to analyze and compare performance evaluation metrics across different datasets.

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.

REFERENCES

Aimen, Khalid, Gran Badshah, Nasir Ayub, Muhammad Shiraz, Mohamed Ghouse, 2023. "Software Defect Prediction Analysis Using Machine Learning Techniques," Sustainability, vol. 15, no. 6, pp. 15,5517.

Emmanuel, Gbenga Dada, David Opeoluwa Oyewola, , Stephen Bassi Joseph,and Ali Baba Dauda, 2021. "Ensemble Machine Learning Model for Software Defect Prediction," Advances in Machine Learning & Artificial Intelligence 2021, vol. 2, pp. 11-21.

I C., SOCIETY, 2024. "What is software quality, "IEEE COMPUTER SOCIETY, New York city,

K. Jan, (2024). "Ensemble Methods: Combining multiple models toimprove prediction accuracy and robustness," Research gate, p. 11,.

Kechao, Wang, Lin Liu, Chengjun Yuan and M. S. G. M. K. L. K. Prashanthi, 2023. "Software Defect Prediction Survey Introducing Innovations with Multiple Techniques," in Advances in Cognitive Science and Communications, CHAP, pp. (pp.783-793).

Mehmood,Iqra, Shahid Sidra, Hussain Hameed, Khan Inayat, Ahmad Shafiq Rahman, Shahid Ullah, Najeeb Huda, Shamsul, 2023. "A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning," IEEE Access, Vols. pp(99):1-1, p. 99,

Pandit, Mahesha and Varma, Nitin, 2019. A Deep Introduction to AI Based Software Defect Prediction (SDP) and its Current Challenges, TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON).

S.M. Rifat, Akil Uddin Bhuyain, Md. Shamim Hossain, Md. Solaiman Mia, Mahmuda Rhman, 2023.A Systematic Approach for Enhancing Software Defect Prediction Using Machine Learning," in 2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM), Gazipur, Bangladesh, 16-17 June 2023.

Sharma, S, Khatter K Soni M., and Sharma R, 2020. "Software defect prediction: do different levels of data relevance effect the prediction accuracy," international journal of information technology, vol. 12(3), no. 41870-020-00418-7, pp. 1073-1084.

Suresh, P., & Kavitha, V, 2021 "Naive Bayes for Software Defect Prediction: A Study on Its Effectiveness in Handling Large Datasets," Journal of Software Engineering Research and Development, vol. 9(3), pp. 123-135.

Zhifei Wang, 2020. "Software defect prediction model based on LASSO–SVM," Springer Link, vol. 33, pp. 8249–8259.

Corresponding author: Abdulrazak Muhammad
abddurrazaq4343@gmail.com
Department of computer science Sokoto State University, Sokoto.