VOL. 05 NO. 9,
FEBRUARY, 2025
E-ISSN 3026-958X
P-ISSN 3027-1169

African Scholar Publications & Research International
www.africanscholarpub.com

Journal of Science Innovation & Technology Research (JSITR)

Significance of Support Vector Machine Classifier for Predicting Defects in Software Development

*Abdulrazak Muhammad Gatawa; & **Yahaya Isah Shehu

*Department of Computer Science, Shehu Shagari College of Education, Sokoto. **Department of Computer Science, Shehu Shagari University of Education, Sokoto

DOI: https://doi.org/10.70382/ajsitr.v7i9.010

Abstract

Software quality is a critical aspect of software development, as defects can significantly affect functionality, security, and user satisfaction. Predicting software defects early in the development cycle enables proactive measures to enhance software reliability and reduce costs. This study investigates software defect prediction using machine learning algorithms, particularly Support Vector Machine (SVM). By leveraging the JM1 dataset from the NASA Promise repository, we explore feature selection techniques, classification models, and performance evaluation metrics as the methodology in this study. The findings of the study revealed that SVM played crucial role in defect prediction in software development with 99% in both accuracy, precision, recall and F1score with optimization, 99.88% accuracy, 99.43% precision, 100% recall and 99.71% of F1 score without optimization. This is invariably enhance software quality assurance practices.

Keywords: Software, Defect, Prediction, Machine, Learning

Introduction

Meeting end-user requirements during the early stages of software deployment remains one of the most challenging aspects of software development. Software defects, if undetected before deployment, can lead to severe consequences, including security vulnerabilities, system failures, financial losses, and diminished user satisfaction. Therefore, ensuring high-quality software delivery by identifying and addressing defects at the earliest possible stage is crucial for both cost reduction and system reliability. Traditional software testing techniques, such as manual code review and dynamic testing, are often time-consuming, labor-intensive, and prone to human error. These limitations have driven the adoption of automated defect prediction techniques that leverage data-driven methodologies to enhance software quality assurance. Software Defect Prediction (SDP) has emerged as a critical research area aimed at identifying defective components before the testing phase. By accurately predicting potential faults, SDP enables software development teams to focus their efforts on high-risk components, thereby optimizing the testing process, reducing maintenance costs, and improving software robustness. SDP leverages machine learning (ML) algorithms to analyze historical defect data, extract meaningful patterns, and forecast fault-prone modules. This predictive capability not only streamlines the development lifecycle but also minimizes post-deployment failures, ultimately leading to higher customer satisfaction and improved software credibility. Machine learning techniques have proven to be highly effective in software defect prediction due to their ability to handle large-scale, high-dimensional data and identify complex relationships between software attributes and defect occurrences. Various ML algorithms, such as Support Vector Machine (SVM), Naïve Bayes (NB), and Random Forest (RF) have been widely applied in SDP research, each offering unique advantages in terms of predictive accuracy, interpretability, and computational efficiency.

This study investigates the application of ML-based defect prediction models using the JM1 dataset from the NASA Promise repository, a widely used benchmark dataset in software defect prediction research. By evaluating support vector machine classification model and performance metrics, the study aims to provide insights into the effectiveness of ML-driven SDP in enhancing software quality assurance.

Importance of Software Quality

Software quality encompasses various attributes, including reliability, usability, performance, security, portability, and maintainability. Organizations strive to develop bug-free software to meet customer expectations and remain competitive. High-quality software reduces maintenance costs, minimizes failures, and enhances user satisfaction.

Problems of Software Defects

The increasing complexity of software systems has escalated the cost and effort required for defect detection. In Krasner *(2018)*, global IT investments reached approximately $4 trillion, with $2.84 trillion attributed to poor software quality. Software failures accounted for 37% of these costs, highlighting the need for effective defect prediction models.

Support vector Machine Algorithms in SDP

Machine learning algorithms play a crucial role in defect prediction by analyzing historical data and identifying patterns. Some widely used ML techniques in SDP include:

- **Support Vector Machines (SVM):** A powerful algorithm used for classification and regression. It finds the optimal hyperplane to separate defect-prone and defect-free software components.

This algorithm facilitates early defect detection, reducing testing efforts and improving software quality.

Research Questions

1. How effective is Support Vector Machine (SVM) in predicting software defects in various software development processes?
2. Can SVM-based machine learning algorithms provide significantly different performance metrics in software defect prediction?

Research Objectives

1. To analyze the efficiency of SVM in defect prediction during software development cycles.
2. To explore appropriate performance metrics for software defect prediction using ML algorithms.

Review of Related Works

In recent work conducted by Alshammari (2022) which is an analysis study on software defect prediction and analysis using enhanced Random Forest (extRF) technique for business process management and improvement concept in IOT-Based application. This study used sampling with conventional machine learning for defect prediction in software component and semi supervised learning (SSL) technique like CoForest method for classification. The output of the analysis indicated that, SSL technique enhanced software defect prediction with improve precision and efficiency and the proposed NSGLP methodology outperforms the other SSL method in software defect prediction. The dataset from NASA promise repository such as KC1, PC1, KC2, and CM1 were used in the analysis. In spite of these, it is very tedious to apply the models to different programming language and also lack finer defect forecasting studies.

Oloduowo et al. (2020) designed software defect prediction using metaheuristic-based feature selection and classification algorithms. The metaheuristic-based correlation feature selection with support vector machine (SVM) and random forest (RF) classifier, Firefly algorithm (FA) and wolf search algorithm (WSA) for feature selection were utilized as the method. Accuracy, precision, recall and F-measure were used as evaluation measures in the study. The JM1, PC5, and MC1 datasets from NASA repository were analyzed in the study. The results of the study showed that, metaheuristic-base FS method improved SVM and RF prediction performance significantly, SVM with WSA based CFS method showed the best prediction performance and also FA and WSA improve feature selection for SVM and RF classifier. However, there is limited comparison with other metaheuristic method such as particle swarm optimization, Tabu search, Genetic Algorithm etc. furthermore, there is need for further evaluation of model on different dataset.

Kaliraj et al. (2023) designed Automatic software bug prediction using adaptive golden eagle optimizer with deep learning. The study analyzed five NASA dataset such as CM1, JM1, KC1, and PC1. 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.

Al-fraihat et al. (2024) performed hyper parameter optimization for software bug prediction using ensemble learning. The study utilized Ensemble Learning and tuning hyper parameter which was run on WEKA software platform. JM1 dataset comprises of 10,885 instances with 20 attributes were in the study. the findings of the study indicated that, Ensemble Learning models outperform single hypothesis learning models in software bug prediction and also hyper parameter optimization enhances machine learning model accuracy in software bug prediction. Despite like of scaling the dataset for the enhancement of the accuracy, there is also need to include future work in the study. Rahim et al. (2021) performed software defect prediction with Naïve Bayes classifier. The study utilized correlation based feature selection for normalization of data and handling of missing values. Naïve Bayes and linear regression were used as model for classification, JM1 dataset from NASA promise Repository was used in the study. The machine learning models classifiers attained accuracy of 95.7% for linear regression and 98.7% for Naïve Bayes classifier. Nevertheless, there was no optimization of classifiers before evaluation. Dhavakumar and Kumar (2024) worked on enhancing software reliability through Naïve Bayes based defect prediction. Complexity evaluation was used to classified feature inform of successful or redesign to enhanced pattern or trend for improving predictive model. Pearson correlation coefficient was utilized to identify relationship between the attributes of the dataset. Naïve Bayes classifier was implemented in building the predictive classification model. Accuracy, precision, Recall, Fi-score, and AUC-ROC were used as evaluation criteria or metrics which achieved the following results of 98.16%, 93%, 94%, 94% and 0.965 of AUC respectively. However, the specific name of the dataset used was not mentioned and also clustering techniques was not used in enhanced the efficiency of predictive classification algorithms. Daoud et al. (2021) have framework on machine learning empowered software defect prediction system. Backpropagation with Bayesian regularization (BR), Artificial Neural Network, Scaled Conjugate Gradient (SCG), Broyden_Fletcher_Goldferb_Shano (BFGS), Quasi Newton (QN) and Levenberg Marquardt (LM) were used in analyzing the dataset. Fuzzy logic for selecting best training function was utilized. The results of the study indicate that, fused ANN_BR outperformed the other algorithms with highest accuracy on various datasets; LM algorithm achieved lower mean square error (MSE) in several datasets. BR excelled in specificity, precision, recall and accuracy on specific datasets. Nevertheless, feature selection and correlation

coefficient was used to enhance the performance of cloud based software defect prediction framework.

Zhang et al. (2024) performed software defect prediction using Deep Q-learning Network Based Feature Extraction. 11 NASA MDP and 11 Promise Repository datasets were used in the study. the datasets are: CM1, JM1, KC1, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. Balanced cascade sampling. Expected calibration (ECE) for metric ranking, Relation matrix construction based on random matrix theory (RMT), Training with decision tree, SVM, and KNN on datasets were used as methods in the study. The proposed DQN based feature selection extraction method enhances SDP precision, and effectiveness. F1, MCC, precision and AUC were utilized as evaluation indicators or performance measures. Moreover, Deep learning methods lack software metric relationship examination.

Khalid et al. (2023) designed software defect prediction analysis using machine learning techniques. K-means clustering was used for class label categorization, particle swarm optimization was used optimized machine learning models, scaling techniques and feature selection were used to scale and extracted the most relevant variables of the dataset. The optimized and scaled variables were used in building models. The findings of the study revealed that SVM and optimized SVM models achieved highest accuracy of 99.80%, RF models achieved precision of 100% and accuracy 98.70%. CM1 dataset was used in the study analysis, all machine learning models had lower error rates performance evaluation. However, limited dataset size was used for classification models and also there is need to analyze more ensemble classifiers with data balancing techniques.

Based on the articles reviewed, all the research papers encountered setbacks, either in their study methodologies or in the performance evaluation measures. However, the software defect prediction analysis using machine learning techniques demonstrated a well-designed methodology covering data preprocessing, feature scaling, correlation matrix, feature clustering, and model building and achieved the highest performance evaluation measures among all the studies. Nevertheless, the dataset used in that study was very limited. Therefore, this study aims to utilize the JM1 dataset, which contains 10,885 features significantly more than the CMI dataset, which has 489 features to compare the performance evaluation measures of the studies.

Methodology

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 paper intends to increase the accuracy and reliability of defect prediction models using Support Vector Machine (SVM), classifier. 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
5. Experimental results discussions
6. Performance evaluation

This research adopts method utilized by Khalid et al. (2023) in their study software defect prediction analysis using machine learning techniques.

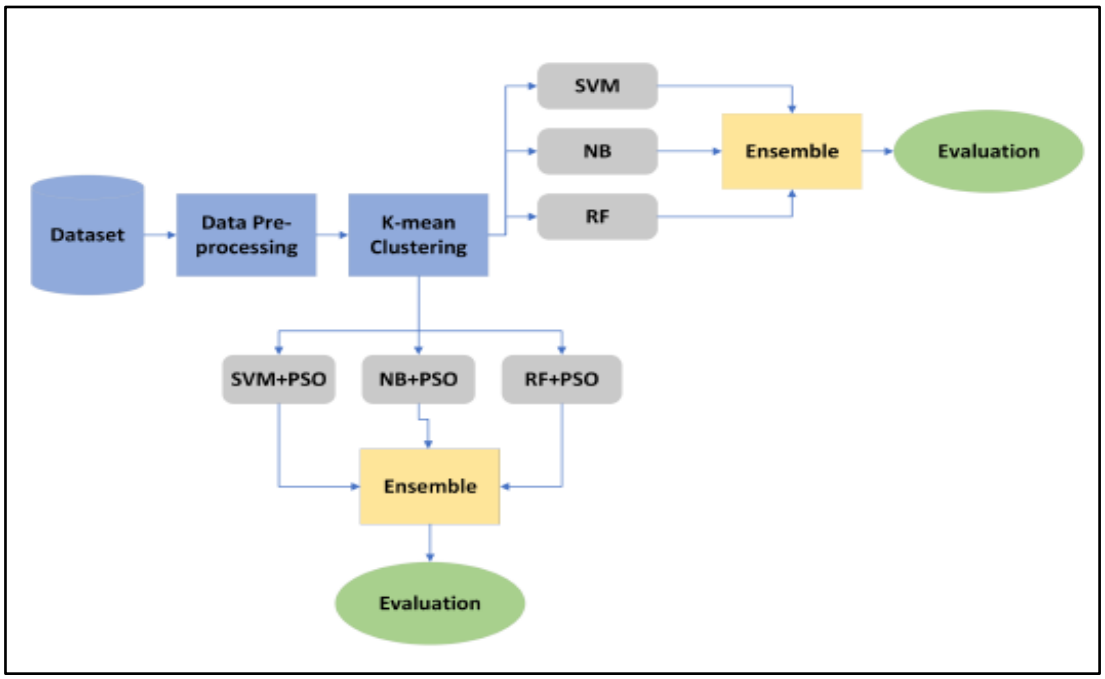


Figure 1. Proposed Framework (Khalid et al. (2023))

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. Table 1 below clearly show the attributes of the dataset.

Title	Language	Source code	Rows	Columns	Defective	Non defective	Defects rate
JM1	C	NASA Space craft instrument	10885	22	2106 = (1)	8779 = (0)	19.34%

1 = True, 0 = False Table 1. Dataset Attribute

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. Table 2 provides a list of the dataset metrics.

Dataset attribute description.

McCabe Metrics		
0	LOC:	Line count of code
1	V(g):	cyclomatic complexity
2	EV(g)	Essential complexity
3	IV(g)	Design complexity
Halstead metrics		
4	N:	Total operators of operands
5	V:	Volume
6	L:	Program length
7	D:	Difficulty
8	I:	Intelligence
9	E:	Effort to write program
10	B:	Delivered Bugs
11	T:	Time Estimator
12	IOCode:	Line count of code
13	IOComment	Count lines of comments
14	IOblank:	count of blank line
15	IOCode& comment	Lines of code and comments
Other Metrics		
16	UniqOp:	Unique Operators
17	UniqOpnd:	Unique operands
18	TotalOp:	Total operators
19	TotalOpnd	Total operands
20	BranchCount:	Flow of Graph
21	D:	Modules has defects or not

Table 2. Descriptions of the dataset attribute

Justification for Using the JM1 Dataset

The JM1 dataset from the NASA Promise software repository was chosen for this research due to its comprehensive nature and widespread acceptance in the software engineering community. This dataset is particularly advantageous because it encompasses a significant number of software modules (10,885) and a diverse range of attributes (22), making it robust for training and evaluating machine learning models. The dataset includes both McCabe and Halstead metrics, which are critical for understanding code complexity and maintainability. These metrics provide a solid foundation for building predictive models that can generalize well across different software projects. Additionally, the JM1 dataset's origin from a NASA spacecraft instrument project implies a high level of reliability and relevance, as NASA's rigorous software development practices ensure the quality and integrity of the data.

Model Design and Classification Algorithms

Classification Algorithms

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 67% of the total, while the testing dataset comprises 33%. Initially, the training data includes output class labels, and the testing data consists of unseen data without output class labels. We employed SVM (Linear SVC classifier). The classifiers were analyzed both with and without Particle Swarm Optimization (PSO) techniques:

Selecting classifiers like Support Vector Machine (SVM),This classifier demonstrated significant performance in terms of accuracy, robustness, and interpretability, making them suitable for this task.

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 Nevendra and Singh (2021) 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.

Experimental Results and Discussion

Performance Metrics

The performance metrics provide a comprehensive overview of the classification model's performance, allowing you to compute various evaluation metrics, such as:

1. Accuracy

The proportion of correct predictions (both positive and negative) out of the total number of predictions. Represented by the Formula below:

Accuracy = (TP + TN)/ (TP +TN + FP + TN)

Measures how often the classifier is correct. It is useful when the dataset is balanced (i.e., there are an equal number of positive and negative samples). However, it can be misleading for imbalanced datasets where one class is significantly more frequent than the other.

2. Precision (Positive Predictive Value)

The proportion of true positive predictions out of all positive predictions made by the model.

Precision = TP / TP+FP

Precision answers the question, "Of all the cases predicted as positive, how many were actually positive?" It is particularly useful when the cost of false positives is high (e.g., spam detection or medical diagnosis where false alarms should be minimized).

3. Recall (Sensitivity or True Positive Rate)

The proportion of true positive predictions out of all actual positive cases in the dataset.

Recall = TP / TP+FN

Recall answers the question, "Of all the actual positive cases, how many were correctly predicted as positive?" It is crucial when the cost of false negatives is high (e.g., in disease detection, where missing a positive case can be costly).

4. F1 Score

The harmonic mean of precision and recall, providing a single metric that balances both concerns.

F1 score = 2 * Precision * Recall / Precision + Recall

The F1 Score is useful when you need to balance precision and recall, especially in cases where there is an uneven class distribution. It is more informative than accuracy when classes are imbalanced.

The output results of the conducted experiment on JMI dataset implementing SVM machine learning algorithm is presented in tables and graphs below. The results of analyzed ML models assess by comparing all the models with and without optimization to make the idea obvious and comprehensive. The tables below show the evaluation metrics for the machine learning techniques experiment with and without optimization respectively. All the analyzed ML models obtained the best results.

Without optimization

Dataset	Model	Accuracy	Precision	Recall	F1 score
JM1	SVM	99.88	99.43	100	99.71

Table 2. Non-Optimized Results

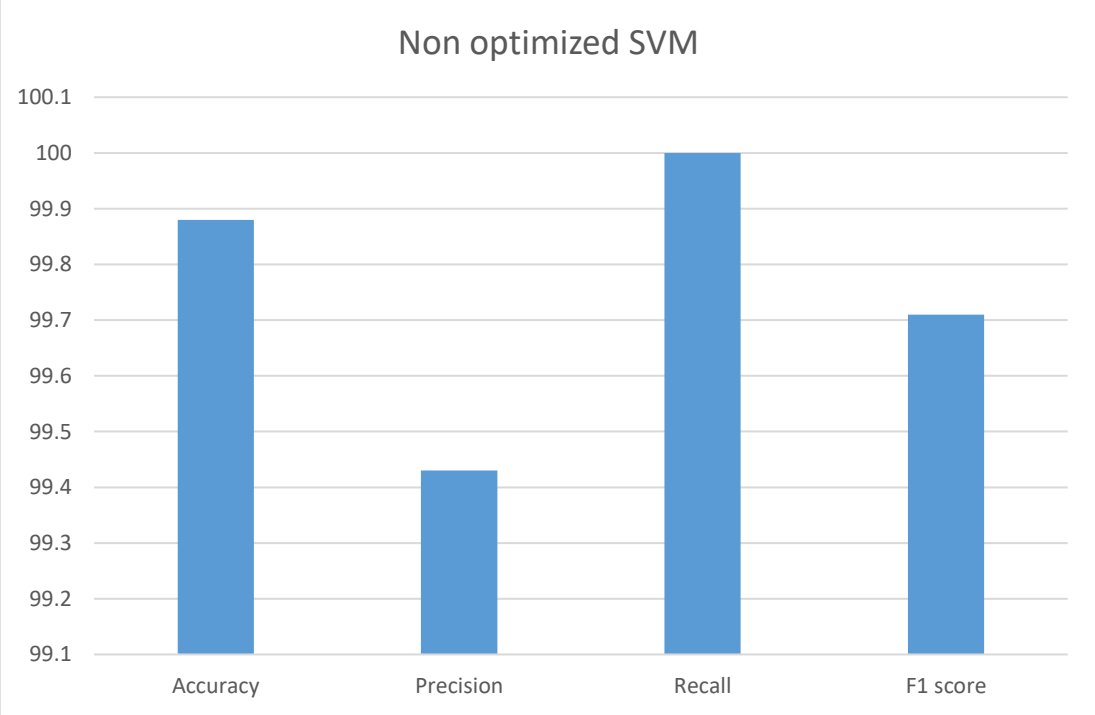


Figure 2. Non-optimized SVM

With optimization

Dataset	Model	Accuracy	Precision	Recall	F1 score
JM1	SVM	99	99	99	99

Table 3. Optimized Results

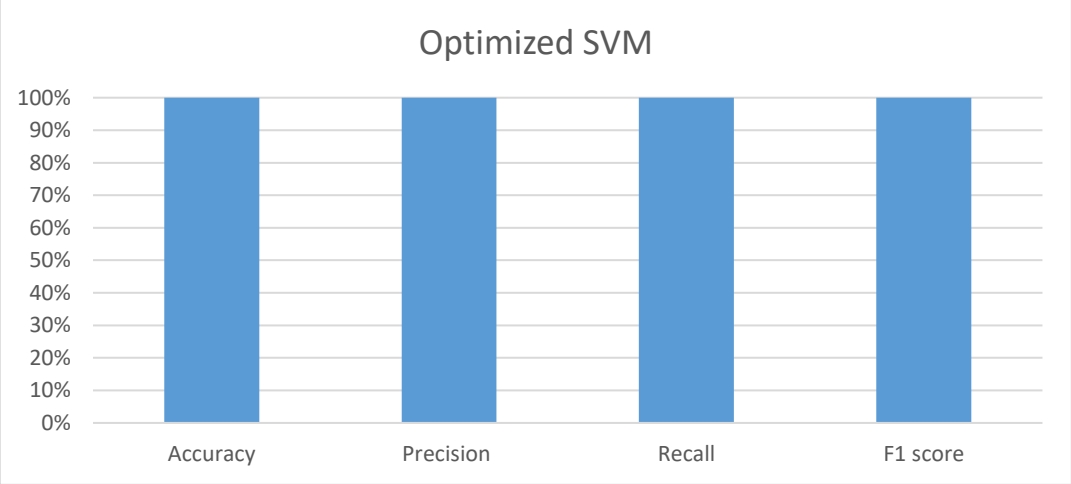


Figure 3. Optimized SVM

Findings and Analysis

This section present the detailed outcomes of the experiment conducted on the JMI dataset using machine learning (ML) algorithms: Support Vector Machine (SVM). The results are systematically organized in tables and visualized through graphs to provide a clear comparison of the performance metrics such as accuracy, precision, recall and F1 score of this algorithms, both with and without optimization.

Evaluation of Machine Learning Classifiers without Optimization

The initial phase of the experiment involved evaluating the performance of SVM machine learning model without any optimization. The evaluation metrics considered include accuracy, precision, recall, and F-measure. The results indicate that the models performed exceptionally well. Specifically, SVM achieved accuracy of 99.88%, precision of 99.43%, recall of 100%, and F-measure of 99.71%, this model is demonstrating its robustness in handling the JMI dataset.

Evaluation of Machine Learning Classifiers with Optimization

The second phase of the experiment involved applying optimization techniques to SVM machine learning model to assess the impact on their performance. The results of this analysis as presented in the above table show the same noticeable

improvement in the all performance metrics of 99% for the all metrics. Interestingly, the optimization had little impact on the performance of the SVM techniques, which maintained their high accuracy of 99% across all metrics. This suggests that these models were already near optimal and that further optimization provided marginal gains.

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

1. How effective support Vector Machine (SVM) is in predicting software defects in various software development processes?

   The experimental results indicated that support vector machine classification machine learning algorithms can be efficiency used in software defect prediction analysis with over 99% in all performance metrics without optimization and 99% of all performance metrics with optimization.
2. Can support vector machine (SVM) machine learning algorithms provide significantly different results of performance metric in software defect prediction analysis?

   The output of the experimental results indicates that, obviously there is significant differences in the results of support vector machine classification model in their performance metrics without optimization with accuracy of 99.88%, precision 99.43%, recall 100% and F1-measure of 99.71%. However, the experimental output with optimization achieved the same results in all performance metrics of 99% across the all metrics.

The findings of this study provide strong evidence that support vector machine (SVM) algorithms are highly effective for software defect prediction in various software development processes. The experimental results demonstrate that SVM can achieve outstanding performance with over 99% accuracy, precision, recall, and F1-measure, even without optimization. This highlights the robustness and reliability of SVM in predicting software defects efficiently.

Moreover, the study reveals that optimizing the SVM model yields consistent results, maintaining a 99% performance across all metrics, while the un-optimized model slightly outperformed in terms of recall (100%) and accuracy (99.88%). The performance differences between the optimized and un-optimized models are statistically significant, suggesting that while optimization can standardize results, the un-optimized model may offer better performance in some cases.

Overall, SVM proves to be a highly capable and effective machine learning algorithm for software defect prediction, with the ability to achieve excellent predictive performance both with and without optimization.

Conclusion and Future Work

This study demonstrated the effectiveness of SVM (Linear classifier), in detecting defects in software development as the JM1 dataset provided a robust foundation for model evaluation. By leveraging the JM1 dataset from the NASA Promise repository, we explore feature selection techniques, classification models, and performance evaluation metrics as the methodology in this study. The findings of the study revealed that SVM played crucial role in defect prediction in software development with 99% in accuracy, precision, recall and F1score with optimization, 99.88% accuracy, 99.43% precision, 100% recall and 99.71% of F1 score without optimization.

Future research could explore:

- Expanding datasets to include more diverse software projects.
- Integrating deep learning techniques for improved defect prediction.
- Investigating hybrid models that combine ML with statistical techniques.

By advancing defect prediction models, organizations can enhance software quality, reduce costs, and improve overall reliability.

References

Al-Fraihat, D., Sharrab, Y., Al-Ghuwairi, A.-R., Alshishani, H., & Algarni, A. (2024). Hyperparameter optimization for software bug prediction using ensemble learning. *IEEE Access, 12*, 51869–51878. https://doi.org/10.1109/ACCESS.2024.3380024

Alshammari, F. H. (2022). Software defect prediction and analysis using enhanced RandomForest (extRF) technique: A business process management and improvement concept in IoT-based application processing environment. *Mobile Information Systems, 2022*, Article ID 2522202. https://doi.org/10.1155/2022/2522202

Daoud, M. S., Aftab, S., Ahmad, M., & Khan, M. A. (2021). Machine learning empowered software defect prediction system. *Journal of Intelligent Automation & Soft Computing, 31*(2), 1288–1300. https://doi.org/10.32604/jasc.2021.017123

Dhavakumar, & Kumar, L. (2024). Enhancing software reliability through Naive Bayes-based defect prediction. *International Journal of Soft Computing and Artificial Intelligence (IJSCAI), 10*, 32–37. https://doi.org/10.21203/rs.3.rs-4268707/v1

Kaliraj, R. S., Hariharan, B., & Premkumar, N. (2023). Automatic software bug prediction using adaptive artificial jelly optimization with long short-term memory. *Wireless Personal Communications, 132*, 1975–1998. https://doi.org/10.1007/s11277-023-10685-w

Khalid, A., Badshah, G., Ayub, N., Shiraz, M., & Ghouse, M. (2023). Software defect prediction analysis using machine learning techniques. *Sustainability, 15*(6), 15517. https://doi.org/10.3390/su152115517

Krasner, H. (2018). *The cost of poor quality software in the US: A 2018 report*. Consortium for IT Software Quality.

Nevendra, M., & Singh, P. (2021). Software defect prediction using deep learning. *Acta Polytechnica Hungarica, 18*(10), 172–189. https://doi.org/10.12700/APH.18.10.2021.10.10

Oloduowo, A. A., Raheem, M. O., Ayinla, F. B., & Ayeyemi, B. M. (2020). Software defect prediction using metaheuristic-based feature selection and classification algorithms. *Ilorin International Journal of Computer Science and Information Technology (ILJCSIT), 3*(1), 23–39.

Rahim, A., Hayat, Z., Abbas, M., Rahim, A., & Rahim, M. A. (2021). Software defect prediction with Naïve Bayes classifier. In *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)* (pp. 1–6). IEEE. https://doi.org/10.1109/IBCAST51254.2021.9393278

Zhang, Q., Zhang, J., Feng, T., Xue, J., Zhu, X., Zhu, N., & Li, Z. (2024). Software defect prediction using deep Q-learning network-based feature extraction. *IET Software, 2024*, 1–34. https://doi.org/10.1049/sfw2.12145