An Extended Software Defect Prediction Framework for Improving Accuracy

Isah Abdullateef [1], **Souley Boukari** [1], **Usman Ali Abdullahi** [2]

[1] *Abubakar Tafawa Balewa University*
Dass road, P. M. B. 0248, Bauchi, 740272, Nigeria

[2] *Federal College of Education (Technical)*
P. M. B 60, Ashaka Road, Gombe, Gombe State, Nigeria

DOI: 10.22178/pos.108-20

LCC Subject Category: T1-995

Received 26.08.2024
Accepted 28.09.2024
Published online 30.09.2024

Corresponding Author:
Isah Abdullateef
isahabdullateef7326@gmail.com

Abstract. Ensuring the quality and reliability of software systems is a critical challenge, mainly due to the difficulty in accurately predicting software defects. Traditional models often struggle with maintaining high accuracy across diverse datasets and classifiers. This study addresses this challenge by exploring the enhancement of defect prediction accuracy by integrating L2 regularisation with feature selection and dataset resampling techniques, forming the With Feature Selection and Regularization (WFS+RG) model.

The evaluation was conducted using five classifiers – Multilayer Perceptron, Bayes Net, Lazy IBK, J48, and Logistic Regression – applied to two datasets, PC1 and JM1, within the Waikato Environment for Knowledge Analysis (WEKA). The results indicate that the WFS+RG model consistently outperforms the traditional model With Feature Selection WFS only across most classifiers. Specifically, on the PC1 dataset, Lazy IBK achieved an accuracy of 97.29%, J48 reached 95.67%, and Multilayer Perceptron obtained 94.40%. Bayes Net and Logistic Regression also showed strong performances with 91.25 and 93.86% accuracy, respectively. On the JM1 dataset, Lazy IBK demonstrated significant improvement with an accuracy of 90.21%, while J48 and Multilayer Perceptron achieved 84.92 and 81.02%, respectively. Bayes Net and Logistic Regression reported 76.93 and 81.10% accuracy, respectively. However, the improvements are not uniform across all classifiers, with some showing minimal change. Statistical analysis conducted via Minitab confirms the robustness and significance of the WFS+RG model's impact, underscoring its potential as a more practical approach in software defect prediction tasks.

Keywords: Software Defect Prediction; Machine Learning Classifiers; L2 Regularization; Dataset Resampling; Feature Selection.

INTRODUCTION

Software defects, commonly referred to as bugs or errors, can significantly impact the quality and functionality of software systems. Enhancing process quality to control and minimise these faults is crucial. One effective way to achieve this is by implementing reliable software development processes and best practices [1]. Defects may arise from various sources, including programming errors and poor design choices, and can lead to malfunctions or unintended outcomes [2]. Testing is a costly aspect of software development, and addressing defects during this phase can further escalate costs and extend project timelines. Identifying defective modules during the development phase before testing can help reduce these testing costs [3]. If defects are not detected early, they may lead to software failures, which can cause user dissatisfaction, reduced efficiency, financial setbacks, and damage to a company's reputation [4]. Various quality assurance measures are employed to mitigate these risks and improve software quality, such as code reviews, unit testing, integration testing, and system testing.

However, these practices can consume up to 80% of a project's budget [5].

Software defect prediction (SDP) often utilises regression and classification techniques. Regression methods are employed to forecast the number of software defects [6]. Various regression models have been explored in the literature for SDP [7–10]. On the other hand, classification methods help determine whether a software module is defective. These methods have shown promise in optimising features and identifying patterns in software code, providing a reliable solution for accurate defect prediction across different datasets [11]. SDP uses machine learning classifiers and historical data from previous projects to identify defect-prone software modules [12]. Prediction models are trained on data from mining historical software repositories [13]. These models analyse various attributes of software modules and recognise patterns that indicate potential defects. By accurately identifying defect-prone modules, SDP enables software practitioners to allocate their limited testing resources more effectively, thereby reducing the risk of software failures. The primary goal of these prediction models is to identify as many defect-prone modules as possible [14].

The SDP process involves three primary steps: gathering a historical defect dataset, training a model by employing machine-learning techniques for classification or regression tasks with the historical data, and using the trained model to forecast the number or probability of software defects [15]. However, imbalanced learning poses a significant challenge in SDP, as software defect datasets tend to be highly skewed, with a small proportion representing defective modules and the majority comprising non-defective ones. Machine learning algorithms trained on these imbalanced datasets may become biased towards non-defective instances, leading to the misclassification of defective instances and reduced accuracy [16, 17]. To improve prediction accuracy, researchers have explored various algorithms, including Extended Nearest Neighbor (ENN), Consolidated Tree Construction (CTC), K-Nearest Neighbors (KNN) [18], Bayesian methods [19], J48 Decision Tree (DT) [20], Naïve Bayes (NB), Support Vector Machine (SVM), Random Forest (RF) [21], Linear Classifiers (LC) [22], Adaboost [7], XGBoost (XGB) and Bayesian Network Model (BNM) [23]. While these methods have significantly advanced the field of SDP, there remains room for improvement, particularly in addressing performance metrics like accuracy. Machine-learning models often struggle with highly imbalanced data in real-world scenarios, significantly impacting the SDP model's overall accuracy [17].

Literature Review

Authors [24] explored defect identification using five NASA datasets (JM1, CM1, KC1, KC2, and PC1). Various machine-learning algorithms, such as Bayesian Net, Logistic Regression, Multilayer Perceptron, Ruler Zero, J48, Lazy IBK, Support Vector Machine, Neural Networks, Random Forest, and Decision Stump were employed for feature selection to achieve optimal accuracy. The results of their findings show that Logistic Regression demonstrated the highest accuracy at 93%, and Bayesian Net exhibited an average accuracy increase of 8% through feature selection. Thus, the problem of imbalanced data is not considered to ensure the reliability of the model's accuracy.

Authors [23] examined various machine learning (ML) methods for predicting software defects across seven commonly utilised datasets. The ML techniques considered encompassed Multilayer Perceptron (MLP), Support Vector Machine (SVM), Decision tree (J48), Radial Basis function (RBF), Random Forest (RF), Hidden Markov Model (HMM), Credal Decision Tree (CDT), K-nearest neighbour (KNN), Average one dependency estimator (ADE), and Naïve Bayes (NB). Performance evaluation involves multiple metrics, including accuracy. The findings revealed that Random Forest (RF) achieved the highest average accuracy at 88.32%, followed by Support Vector Machine (SVM) with an average accuracy of 87.99%. However, the study focuses on a predefined set of ML techniques, potentially overlooking newer or specialised algorithms that may offer superior performance in defect prediction tasks.

Authors [25] compared different machine learning models for Software Defect Prediction (SDP) using NASA Metrics Data Program (MDP) datasets. Their study evaluated traditional machine learning models such as K-nearest neighbours (k-NN), Decision Trees (DT), Logistic Regression (LR), Linear Discriminant Analysis (LDA), Single Hidden Layer Multilayer Perceptron (SHL-MLP), and Support Vector Machine (SVM) for their ability to predict software defects accurately. The experimental results showed that K-NN, SVM, and SHL-MLP are the most effective algorithms for SDP, achieving the highest accuracy, precision,

recall, and F1 scores on multiple datasets. However, the effectiveness of hyper-parameter tuning and SMOTE oversampling may depend on dataset characteristics and may not always result in improved model performance.

Authors [26] investigate different techniques for Software Fault Prediction (SFP) and evaluate the performance of seven ensemble techniques: Dagging, Decorate, Grading, MultiBoostAB, RealAdaBoost, and Rotation Forest. The experiment utilises three classification algorithms (naive Bayes, logistic regression, and J48) as base learners for the ensemble techniques. Rotation Forest with J48 as the base learner achieves the highest accuracy. However, these methods may be impacted by the fault datasets' attributes and the foundational learner's selection.

The author [18] employed a stacked ensemble with a hierarchical approach, combining multiple models to handle class imbalance and enhance the performance of Software Defect Prediction (SDP) classifiers. The stacked ensemble comprised three weak classifiers (KNN, NB, DT) and two robust classifiers (ANN, SVM). The experiment was conducted using the publicly available NASA corpus. The findings revealed that Goyal's proposed stacked ensemble model demonstrated superior performance to the baseline models, specifically in the Area Under the Curve (AUC) and accuracy metrics. However, the findings are constrained to a singular dataset, posing challenges in drawing comprehensive conclusions.

Authors [27] conducted a systematic literature review on studies from 1990 to June 2019 focusing on software fault prediction (SFP) utilising machine learning and statistical methods. The study reveals that machine-learning techniques generally outperform classical statistical models in SFP. However, the focus on studies from 1990 to June 2019 does not encompass all recent developments and advancements in SFP.

Authors [14] conducted a survey focusing on deep learning methodologies for defect prediction, reviewing recent research in the field. They observed the lack of universally recognised standard benchmarks for defect prediction. They highlighted the challenge posed by variations in performance metrics and datasets across different studies, making it challenging to determine the most effective models.

Authors [28] used a regularisation model and optimisation to predict the hourly air pollution concentration based on meteorological data of previous days by formulating the prediction over 24 has a Multi-Task Learning (MTL) problem. The experiments have shown that their proposed method achieved outstanding results.

Authors [29] have applied regularisation techniques in conjunction with other models in the medical domain to predict cancer diagnosis based on descriptions of nuclei extracted from breast masses. The trained algorithms achieved a high accuracy rate of approximately 94% to 96% in classifying cell nuclei.

Authors [30] applied various Software Defect Prediction (SDP) approaches, including the K-Means methodology, Support Vector Machines (SVM) linear, Random Forest (RF), and Multilayer Perceptron (MLP) algorithms. The evaluation of their proposed defect prediction models involved a comprehensive assessment using metrics such as false alarm rate, precision, and detection rate. The outcomes revealed consistently high accuracy across all the techniques utilised.

Authors [31] examined the effect of the Synthetic Minority Over-sampling Technique (SMOTE) as a base classifier on the Adaptive boost ensemble system with J48. The result showed that SMOTE improved the ensemble method on four NASA datasets by boosting accuracy and other metrics. Additionally, they noted that highly imbalanced class distributions within datasets had a detrimental impact on the effectiveness of classification methods. The problem with this method is the issue of overfitting and inflation of performance metrics.

Authors [19] examined the effectiveness of Minority Clustering SMOTE (MC-SMOTE), which involves clustering minority class samples to enhance imbalance classification performance. Their classification results indicated that MC-SMOTE demonstrates superior performance compared to traditional SMOTE. But, suffer from overfitting problem.

The author [20] evaluated hybrid ensemble and machine learning classifiers across eight PROMISE datasets. The findings indicated that AdaBoost support vector machines and bagging SVM outperformed other models in terms of metrics such as Accuracy, Area under the Curve, recall, and F-measure. However, the study does not provide detailed insights into the specific characteristics of the PROMISE datasets used for evaluation.

Authors [28] proposed a defect prediction framework named Defect Prediction via Convolutional

Neural Network (DP-CNN). It employs Convolutional Neural Networks (CNN) to automatically extract features from source code while retaining both semantic and structural information. Their scheme also incorporates word embedding to integrate CNN-learned features with traditional handcrafted features to enhance software defect prediction. Their results indicate DP-CNN enhances defect prediction compared to DBN-based and traditional features-based techniques by 12% and 16%, respectively, in terms of accuracy and F-measure. However, the study lacks a detailed comparison with existing defect prediction, making it challenging to assess its novelty.

Authors [21] used five datasets, namely PC3, MW1, KC1, PC4, and CM1, with the weka simulation tool and hybridised approach that includes the Principal Component Analysis (PCA), Random Forest, Naïve Bayes and the Support Vector Machine to address the problem of imbalance datasets. Evaluation metrics such as accuracy, precision, recall, and recognition accuracy, among others, were analysed, suggesting that these methods offer valuable predictive capabilities. Still, their approach did not consider the data imbalance issue, making it difficult to assess their metrics objectively.

Authors [21] conducted a study to enhance the performance of software defect prediction models by employing various machine learning (ML) techniques and optimising them using a publicly available dataset. The findings revealed significant performance improvements in both ML and optimised ML models. Support Vector Machine (SVM) and optimised SVM models exhibited exceptional accuracy rates. However, the study failed to address the impact of imbalanced data on the model performance.

Authors [33] utilised the Genetic Algorithm (GA) for feature selection from the dataset. They implemented a decision tree (DT) framework to forecast software defects within their hybrid machine learning-based approach to software defect prediction. The classification performance, particularly concerning accuracy and recall, demonstrated an improved result. Nevertheless, data imbalance remains an unresolved challenge in their work.

Authors [34] used a fuzzy rule-based approach to predict software modules with defects by applying different classifiers. They evaluated the performance of these classifiers and determined that the prediction accuracy varied across different datasets. However, the study does not provide detailed insights into the specific datasets used for experimentation or the characteristics of these datasets.

Authors [35] introduced a novel methodology. They employed a feature selection technique alongside machine-learning methods on five datasets from the NASA repository: JM1, KC1, PC1, CM1 and KC2. The objective was to reduce complexity and eliminate irrelevant or erroneous data, ultimately aiming to augment prediction accuracy. The findings from their research indicated an enhancement in the accuracy of defect prediction when utilising feature selection (WFS) compared to the accuracy without feature selection (WOFS). However, a significant limitation in this study is the absence of consideration for the challenges posed by imbalanced data in their datasets and potential overfitting.

Authors [36] significantly improved software defect prediction accuracy. They employ a hybrid heterogeneous ensemble approach for software defect prediction, utilising a diverse set of classifiers including k-Nearest Neighbour (k-NN), Naive Bayes (NB), Decision Trees (DT), Bagging, Adaptive Boosting (AdaBoost), Random Forest (RF), and XGBoost (XGB). Additionally, the study incorporates the k-means clustering algorithm to segment the training data into clusters. Furthermore, the cross-validation methodology is utilised to train and assess the classification algorithms on each cluster, aiming to identify the most appropriate and specialised model for each cluster. But, despite the great strides recorded, the clustering algorithm used in the study has apparent inadequacy. The techniques cannot automatically identify the optimal quantity of clusters for each dataset without manual intervention. This can affect the accuracy, efficiency, and robustness of clustering results.

METHODOLOGY

This segment outlines the working principle of the existing model and its shortcomings and introduces the proposed framework along with its operational principles.

Working principle of the existing model. In their innovative method for enhancing software defect prediction accuracy, [36] used the feature selection method that combines Supervised, Unsupervised, and Semi-Supervised learning principles within a SDP framework, aiming to identify and

select the most relevant subset of features from a dataset, to improve model performance. The Supervised Component uses labelled data to evaluate the relevant features through a filtering technique. Likewise, the Unsupervised Component operates on the complete dataset, identifying features that capture inherent structure using a method based on clustering-based selection and utilising the integrated components results in a subset that improves the learning process. The final element of their framework, the Semi-Supervised component, leverages labelled (Supervised) and unlabeled (Unsupervised) data to create or reconstruct new data points from the original datasets. By optimising and integrating the outputs from all these components, the SDP framework aims to improve predictive accuracy. However, integrating multiple components, such as the supervised, unsupervised, and semi-supervised methods, can increase computational complexity, create high costs, and increase the potential for overfitting. Additionally, the model did not consider the critical aspect of data imbalance, a crucial factor in attaining dependable predictive accuracy. The operational mechanism of the current model is depicted in Figure 1.

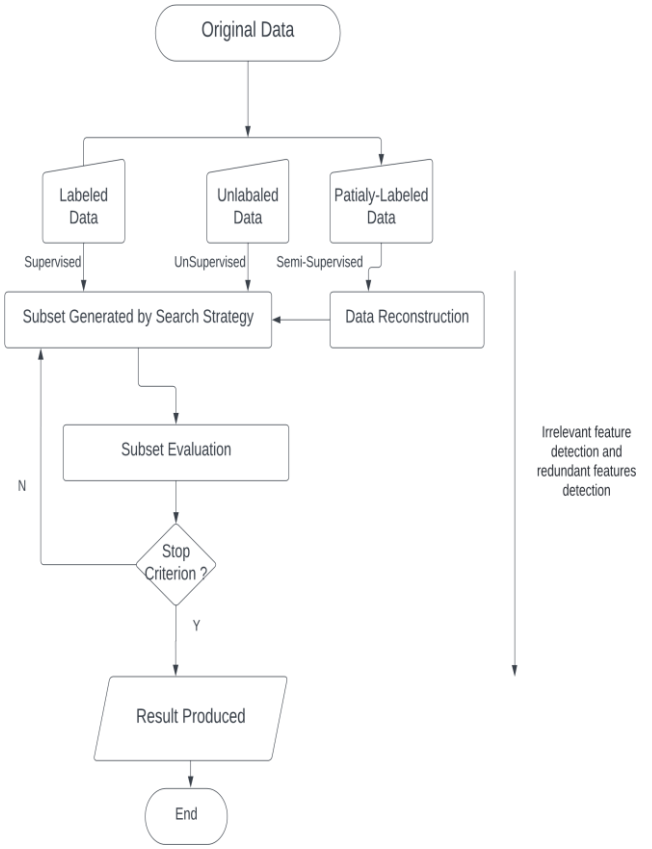

Figure 1 – Feature Selection Framework [36]

Weaknesses of the existing model. An extensive review of the existing model revealed certain deficiencies in the current model, indicating areas that warrant further investigation. While existing models have made significant strides in enhancing predictive accuracy, they are not without weaknesses. Specifically, they include:

The feature selection framework proposed by [36] specifically used Logistic Regression, Bayes Net, Multilayer Perceptron (MLP), J48, Decision Stump, Random Forest (RF), Support Vector Machine (SVM), Lazy IBK, and Rule ZeroR algorithms. This diverse set of algorithms contributed to increased computational complexity, high costs and the potential for overfitting.

The model did not consider the critical aspect of data imbalance, a crucial factor in attaining dependable predictive accuracy.

The Proposed Framework Analysis. As illustrated in Figure 2, the proposed framework serves as a comprehensive blueprint for the subsequent discussion and analysis in this context.

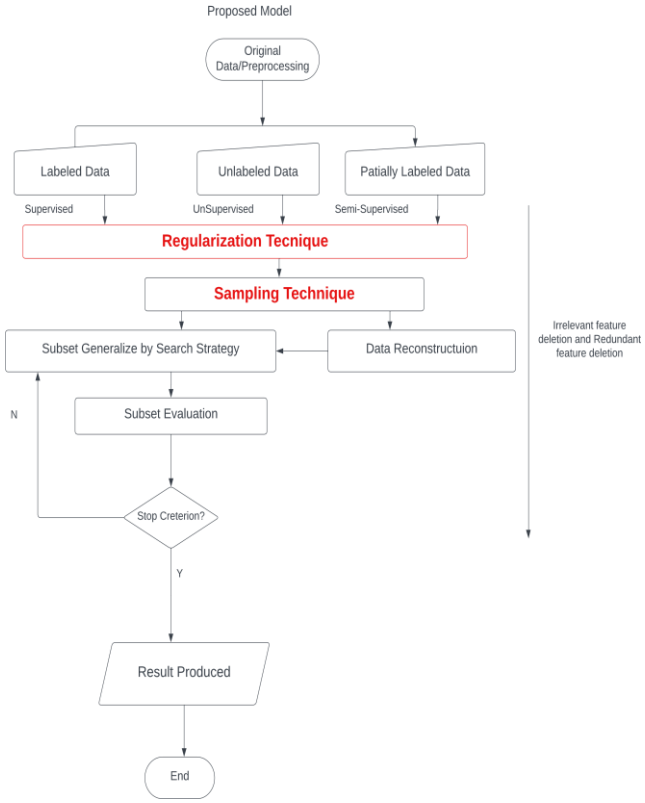

Figure 2 – Proposed Framework

Working principle of the proposed framework. This section presents the operational framework outlined in Figure 2. Building upon the existing model

by [36], our proposed framework extends its work by incorporating two key components: regularisation and resampling techniques. The goal of this integration is to enhance the accuracy of the existing model. The methodology involved using a machine learning tool called the Waikato Environment for Knowledge Analysis (WEKA) to integrate L2 regularisation with feature selection and dataset resampling. Statistical analysis of the results was then conducted using the Minitab tool. Bold caption boxes visually represent this augmentation in the diagram. The proposed framework begins with an input dataset (Original data) comprising labelled, unlabeled, and partially labelled samples. These techniques evaluate feature relevance based on labelled data, identify features capturing essential data structures, and select the most relevant features.

Next, the framework is extended by incorporating regularisation with resampling methods into the feature selection process. Regularisation adds a penalty term to the loss function, which helps manage model complexity and reduce overfitting. The penalty takes the form of the absolute value of the coefficient magnitudes (L1 regularisation), as shown in equation 1 below:

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^{n} (\text{yi} - y\text{^}\,)\,2 + \lambda \sum_{i=1}^{M} |\text{Wi}| \qquad (1)$$

where M – quantity of features; n – number of samples in the dataset; y^ = predicted target value for the i-th sample; y_i = actual target value for the i-th sample; λ = regularisation parameter.

Additionally, regularisation includes the squared magnitude of the coefficients (L2 regularisation), as shown in equation 2 below:

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^{n} (\text{yi} - y\text{^}\,)\,2 + \lambda \sum_{i=1}^{M} W_i^2 \qquad (2)$$

L2 regularisation mitigates the problem of overfitting, which occurs when a model becomes too complex and captures noise in the training data rather than the underlying pattern. Overfitting can lead to poor generalisation of unseen data. This issue is addressed by adding a penalty proportional to the sum of the squared coefficients of the features in the model. The penalty discourages the model from assigning large weights to any one feature, effectively shrinking the coefficients of less essential features. As a result, the model becomes more straightforward and less prone to overfitting, leading to better generalisation and improved accuracy on new data.

On the other hand, the resampling technique is applied to improve model accuracy by addressing the issue of class imbalance, which is common in software defect prediction datasets. This integration aims to manage model complexity, reduce overfitting during feature selection, and address class imbalance. The framework then generates a subset of selected features based on the regularised model, promoting selecting informative features while discarding less relevant ones. The subsequent phase involves evaluating the performance of the selected feature subset using the target accuracy metrics and validation techniques on the WEKA software environment. The feature selection process iterates with different regularisation techniques to optimise performance, continuing until a predefined stopping criterion is met. Finally, the framework concludes by finalising the selected feature subset for subsequent modelling, resulting from the feature selection process supported by regularisation and resampling techniques.

RESULTS AND DISCUSSION

This section discusses the performance of the proposed feature selection, L2 regularisation, and resampling techniques to enhance the accuracy of software defect prediction models. The evaluation was conducted using various classifiers within the WEKA (Waikato Environment for Knowledge Analysis) framework, and the results were analysed using Minitab software. The proposed technique was first tested on benchmark datasets from the NASA PROMISE repository. The results were then compared with those of existing techniques. This research employed five classifiers - Multilayer Perceptron, Bayes Net, Lazy IBK, J48, and Logistic Regression – and two datasets from the PROMISE repository: JM1 and PC1. Based on the specified classifiers, the results of defect prediction accuracy using feature selection and L2 regularisation for the PC1 and JM1 datasets are presented in Tables 1 and 2. Figures 1 and 2 provide performance charts for each dataset. The accuracy results for the Feature Selection called With Feature Selection WFS model are taken from [36]. In contrast, the proposed model, which integrates feature selection, regularisation, and resampling techniques, is referred to as WFS+RG.

Table 1 – Defect prediction accuracy WFS+RG for PC1

Classifier	Dataset	Accuracy WFS+RG in %
Multilayer Perceptron	PC1	94.40
Bayes Net	PC1	91.25
Lazy IBK	PC1	97.29
J48	PC1	95.67
Logistic Regression	PC1	93.86

Table 1 presents the accuracy results of defect prediction for different classifiers using the PC1 dataset, as detailed in Table 1. The results demonstrate a significant accuracy improvement when applying the proposed feature selection, L2 regularisation (WFS+RG), and resampling techniques on the PC1 dataset. Lazy IBK emerged as the top performer with an accuracy of 97.29%, followed by J48 at 95.67% and Multilayer Perceptron at 94.40%. Bayes Net also delivered a solid performance, achieving 91.25% accuracy. These findings highlight the effectiveness of the proposed method in enhancing the predictive accuracy of SDP models across various classifiers.

Table 2 – Defect prediction accuracy WFS+RG for JM1

Classifier	Dataset	Accuracy WFS+RG in %
Multilayer Perceptron	JM1	81.02
Bayes Net	JM1	76.93
Lazy IBK	JM1	90.21
J48	JM1	84.92
Logistic Regression	JM1	81.10

Based on the results in Table 2 Lazy IBK once again demonstrates the best performance, achieving an accuracy of 90.21% on the JM1 dataset, underscoring its robustness across different datasets when paired with the WFS+RG model. Following Lazy IBK, J48 and Logistic Regression also show strong results, with 84.92% and 81.10% accuracy, respectively. However, the Bayes Net classifier exhibits the lowest performance on this dataset, with an accuracy of 76.93%.

Table 3 compares results for the PC1 dataset using the WFS technique. The results indicate modest accuracy improvements across most classifiers when using the WFS+RG technique compared to WFS alone.

Table 3 – Comparison with (WFS) on PC1

Classifier	Data Set	Accuracy WFS	Accuracy WFS+RG
Multilayer Perceptron	PC1	93.77	94.40
Bayes Net	PC1	91.70	91.25
Lazy IBK	PC1	91.88	97.29
J48	PC1	93.32	95.67
Logistic Regression	PC1	93.32	93.86

For example, the WFS+RG model slightly outperforms the base WFS model, with a 0.63% accuracy improvement in the Multilayer Perceptron classifier. This suggests that the resampling and feature selection in the WFS+RG model enhance the MLP's generalisation ability by reducing overfitting and emphasising more relevant features. However, the Bayes Net classifier experiences a slight decrease in accuracy with the WFS+RG model, dropping by 0.45%. This decrease may result from the resampling technique introducing variations that do not align well with the model.

In contrast, the Lazy IBK classifier substantially improved 5.41% with the WFS+RG model. Similarly, the J48 classifier benefits from a 2.35% accuracy increase, likely due to better generalisation and reduced overfitting after feature selection and resampling. Finally, Logistic Regression sees a modest improvement of 0.54% with the WFS+RG model.

Table 4 – Comparison with (WFS) on JM1

Classifier	Data Set	Accuracy WFS	Accuracy WFS+RG
Multilayer Perceptron	JM1	81.08	81.02
Bayes Net	JM1	75.17	76.93
Lazy IBK	JM1	76.38	90.21
J48	JM1	81.11	84.92
Logistic Regression	JM1	80.94	81.1

The comparison of results for the JM1 dataset is presented in Table 4. The MLP classifier shows a negligible decrease in accuracy, dropping by just 0.06% with the WFS+RG model. In contrast, Bayes Net sees a 1.76% improvement with the WFS+RG model. Notably, Lazy IBK experiences a dramatic 13.83% accuracy increase, underscoring the effectiveness of the WFS+RG approach in refining

the dataset. The J48 classifier also benefits from a 3.81% improvement with the WFS+RG model. Lastly, Logistic Regression shows a minor 0.16% improvement, consistent with its performance on the PC1 dataset.

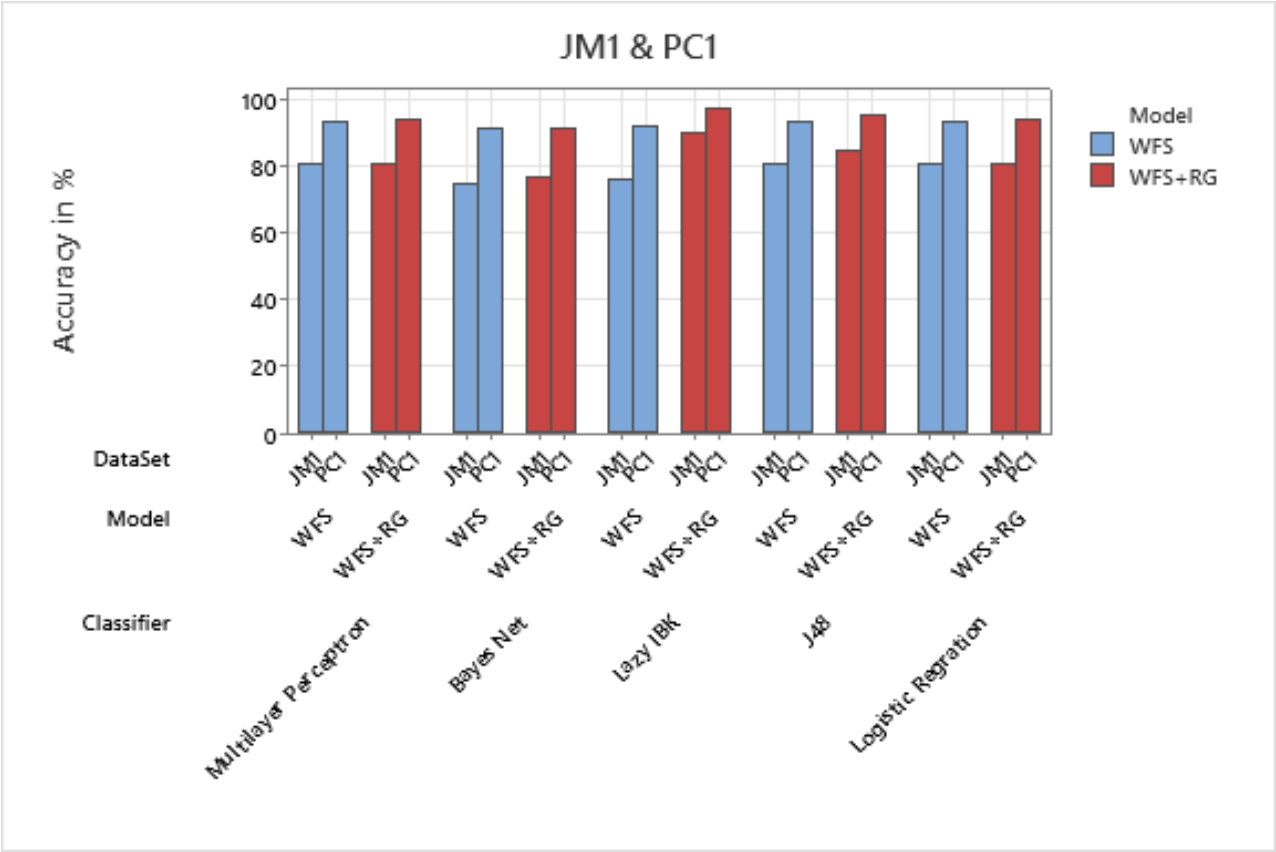

Figure 5 – Comparison on JM1&CM1 data set utilising different classifiers (proposed method (WFS+RG) and (WFS))

The comparison between the base WFS model and the proposed WFS+RG model across two datasets (PC1 and JM1) reveals several key insights into the efficacy of the WFS+RG approach in enhancing Software Defect Prediction (SDP) accuracy. The most notable improvement is observed with the Lazy IBK classifier, where the proposed WFS+RG model boosts accuracy from 91.88% to 97.29% on the PC1 dataset and by an impressive 13.83% on the JM1 dataset. The J48 classifier also shows considerable improvement, gaining 2.35% accuracy on the PC1 dataset and 3.81% on the JM1 dataset.

Multilayer Perceptron and Logistic Regression demonstrate modest improvements with the WFS+RG model, indicating that while the model provides some benefit, the gains are not as substantial as those seen with Lazy IBK or J48. Bayes Net presents mixed results, with a slight decrease in accuracy on the PC1 dataset and a minor improvement on the JM1 dataset.

In general, the WFS+RG model surpasses the WFS model in performance across most classifiers, especially with the PC1 dataset, where improved feature selection and resampling techniques contribute to better generalisation and reduced overfitting. Although the WFS+RG model typically enhances accuracy on the JM1 dataset, the results are more inconsistent. The significant improvement observed with Lazy IBK, compared to the minimal changes in other classifiers, indicates that the dataset's complexity or noise might impact the effectiveness of the WFS+RG model. In conclusion, the WFS+RG model outperforms the base WFS model, particularly for classifiers like Lazy IBK and J48. This indicates that the integrated approach of L2 regularisation, feature selection, and resampling in WFS+RG enhances the predictive accuracy of these classifiers.

CONCLUSIONS

The integration of L2 regularisation, feature selection, and resampling techniques within the WFS+RG model has shown considerable promise in enhancing software defect prediction accuracy across various classifiers and datasets. This combined approach provides a more robust framework for addressing the complexities and variabilities inherent in software defect datasets. Key findings highlight that WFS+RG is particularly effective in improving the performance of classifiers such as Lazy IBK and J48. The model's ability to refine feature selection and balance datasets through resampling results in more accurate predictions, as evidenced by the significant accuracy gains observed with these classifiers. This suggests that the WFS+RG model effectively reduces noise and eliminates irrelevant features, leading to better generalisation and a lower risk of overfitting. However, the effectiveness of the WFS+RG model is not uniform across all classifiers and datasets. While it consistently enhances performance, the degree of improvement varies depending on the classifier and the specific dataset. For example, Bayes Net exhibits mixed results, indicating that the benefits of WFS+RG may be context-dependent, requiring careful consideration of the underlying data structure and the characteristics of the classifier.

Overall, the WFS+RG model offers a promising approach to software defect prediction by providing a balanced and refined feature set that enhances predictive accuracy. Its successful application across multiple classifiers and datasets underscores its versatility and potential as a standard approach in software defect prediction tasks. Future research could explore further optimisations and extensions of this model, potentially incorporating additional machine-learning techniques to improve its adaptability and performance across diverse datasets.

REFERENCES

1. Harizaj, M., Harizaj, A., Idrizi, O., & Tomco, V. (2023). Analysis And Potential Improvement Of Software Fault Prediction Approaches. *Journal of Southwest Jiaotong University, 58*(4).

2. Gupta, R., Rajkumar, A., & N., B. (2023). Predicting software defects with swarm-intelligence-based machine learning algorithm for improved process quality. *Multidisciplinary Science Journal, 5*, 2023ss0311. doi: 10.31893/multiscience.2023ss0311

3. Ali, U., Aftab, S., Iqbal, A., Nawaz, Z., Salman Bashir, M., & Anwaar Saeed, M. (2020). Software Defect Prediction Using Variant based Ensemble Learning and Feature Selection Techniques. *International Journal of Modern Education and Computer Science, 12*(5), 29–40. doi: 10.5815/ijmecs.2020.05.03

4. Thota, M. K., Shajin, F. H., Rajesh, P. (2020). Survey on software defect prediction techniques. *International Journal of Applied Science and Engineering, 17*, 331–344.

5. Qiao, L., Li, X., Umer, Q., & Guo, P. (2020). Deep learning based software defect prediction. *Neurocomputing, 385*, 100–110. doi: 10.1016/j.neucom.2019.11.067

6. Kalaivani, N., & Beena, R. (2018). Overview of software defect prediction using machine learning algorithms. *International Journal of Pure and Applied Mathematics, 118*(20), 3863–3873.

7. Meng, F., Cheng, W., & Wang, J. (2021). Semi-supervised Software Defect Prediction Model Based on Tri-training. *KSII Transactions on Internet and Information Systems, 15*(11). doi: 10.3837/tiis.2021.11.009

8. Chen, H., Chen, Z., Zheng, H., Ge, L., & Gao, X. (2022). Policy shock effect of SDP on environmental total factors productivity: 53 coal cities versus 165 non-resource-based cities. *Environmental Science and Pollution Research, 29*(30), 46145–46160. doi: 10.1007/s11356-022-19163-5

9. Fanuel, M., & Tyagi, H. (2022). *Denoising modulo samples: k-NN regression and tightness of SDP relaxation*. Retrieved from https://arxiv.org/abs/2009.04850

10. Saheed, Y. K., Longe, O., Baba, U. A., Rakshit, S., & Vajjhala, N. R. (2021). An Ensemble Learning Approach for Software Defect Prediction in Developing Quality Software Product. *Advances in Computing and Data Sciences*, 317–326. doi: 10.1007/978-3-030-81462-5_29

11. Shankar, B. M., Sivakumar, S. A., Dhabliya, D., Sundari, P. A., Asmitha, M., & Shree, S. M. G. (2023). Software Defect Prediction using ANN Algorithm. *2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 682–686. doi: 10.1109/i-smac58438.2023.10290523

12. Aljamaan, H., & Alazba, A. (2020). Software defect prediction using tree-based ensembles. *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*, 1–10. doi: 10.1145/3416508.3417114

13. Bashir, K., Li, T., Yohannese, C. W., & Yahaya, M. (2020). SMOTEFRIS-INFFC: Handling the challenge of borderline and noisy examples in imbalanced learning for software defect prediction. *Journal of Intelligent & Fuzzy Systems, 38*(1), 917–933. doi: 10.3233/jifs-179459

14. Akimova, E. N., Bersenev, A. Yu., Deikov, A. A., Kobylkin, K. S., Konygin, A. V., Mezentsev, I. P., & Misilov, V. E. (2021). A Survey on Software Defect Prediction Using Deep Learning. *Mathematics, 9*(11), 1180. doi: 10.3390/math9111180

15. Feng, S., Keung, J., Yu, X., Xiao, Y., Bennin, K. E., Kabir, M. A., & Zhang, M. (2021). COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction. *Information and Software Technology, 129*, 106432. doi: 10.1016/j.infsof.2020.106432

16. Thi-Mai-Anh, B., & Nhat-Hai, N. (2023). On the Value of Code Embedding and Imbalanced Learning Approaches for Software Defect Prediction. *Proceedings of the 12th International Symposium on Information and Communication Technology*, 510–516. doi: 10.1145/3628797.3628963

17. Gong, L., Jiang, S., & Jiang, L. (2019). Tackling Class Imbalance Problem in Software Defect Prediction Through Cluster-Based Over-Sampling With Filtering. *IEEE Access, 7*, 145725–145737. doi: 10.1109/access.2019.2945858

18. Goyal, S., & Bhatia, P. K. (2021). Heterogeneous stacked ensemble classifier for software defect prediction. *Multimedia Tools and Applications, 81*(26), 37033–37055. doi: 10.1007/s11042-021-11488-6

19. Yi, H., Jiang, Q., Yan, X., & Wang, B. (2021). Imbalanced Classification Based on Minority Clustering Synthetic Minority Oversampling Technique With Wind Turbine Fault Detection Application. *IEEE Transactions on Industrial Informatics, 17*(9), 5867–5875. doi: 10.1109/tii.2020.3046566

20. Qu, Y., Li, Z., Zhao, J., & Li, H. (2022). Unbalanced data processing for software defect prediction. *2022 4th International Conference on Data-Driven Optimization of Complex Systems (DOCS)*, 1–6. doi: 10.1109/docs55193.2022.9967755

21. Khalid, A., Badshah, G., Ayub, N., Shiraz, M., & Ghouse, M. (2023). Software Defect Prediction Analysis Using Machine Learning Techniques. *Sustainability, 15*(6), 5517. doi: 10.3390/su15065517

22. Deep Singh, P., & Chug, A. (2017). Software defect prediction analysis using machine learning algorithms. *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*, 775–781. doi: 10.1109/confluence.2017.7943255

23. Khan, B., Naseem, R., Shah, M. A., Wakil, K., Khan, A., Uddin, M. I., & Mahmoud, M. (2021). Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques. *Journal of Healthcare Engineering, 2021*, 1–16. doi: 10.1155/2021/8899263

24. Sheth, V., Tripathi, U., & Sharma, A. (2022). A Comparative Analysis of Machine Learning Algorithms for Classification Purpose. *Procedia Computer Science, 215*, 422–431. doi: 10.1016/j.procs.2022.12.04

25. Siswantoro, M. Z. F. N., & Yuhana, U. L. (2023). Software Defect Prediction Based on Optimised Machine Learning Models: A Comparative Study. *Teknika, 12*(2), 166–172. doi: 10.34148/teknika.v12i2.634

26. Rathore, S. S., & Kumar, S. (2020). An empirical study of ensemble techniques for software fault prediction. *Applied Intelligence, 51*(6), 3615–3644. doi: 10.1007/s10489-020-01935-6

27. Pandey, S. K., & Tripathi, A. K. (2021). Class Imbalance Issue in Software Defect Prediction Models by various Machine Learning Techniques: An Empirical Study. *2021 8th International Conference on Smart Computing and Communications (ICSCC)*, 58–63. doi: 10.1109/icscc51209.2021.9528170

28. Li, J., He, P., Zhu, J., & Lyu, M. R. (2017). Software Defect Prediction via Convolutional Neural Network. *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 318–328. doi: 10.1109/qrs.2017.42

29. Sidey-Gibbons, J. A. M., & Sidey-Gibbons, C. J. (2019). Machine larning in medicine: a practical introduction. *BMC Medical Research Methodology, 19*(1). doi: 10.1186/s12874-019-0681-4

30. Shaikh, S. Y., Qureshi, N. A., Khan, M. Z., Khan, M. A., Imroz, A., & Kalwar, M. A. (2023). Performance Analysis of Classification Algorithms for Software Defects Prediction by Mathematical Modelling & Simulations. *Journal of Software Engineering, 2*(1), 1–28.

31. Alsawalqah, H., Faris, H., Aljarah, I., Alnemer, L., & Alhindawi, N. (2017). Hybrid SMOTE-Ensemble Approach for Software Defect Prediction. *Software Engineering Trends and Techniques in Intelligent Systems*, 355–366. doi: 10.1007/978-3-319-57141-6_39

32. Zubair Khan, M. (2020). Hybrid Ensemble Learning Technique for Software Defect Prediction. *International Journal of Modern Education and Computer Science, 12*(1), 1–10. doi: 10.5815/ijmecs.2020.01.01

33. Manjula, C., & Florence, L. (2018). Hybrid Approach for Software Defect Prediction Using Machine Learning with Optimisation Technique. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 12*, 28-32.

34. Singh, P., Pal, N. R., Verma, S., & Vyas, O. P. (2017). Fuzzy Rule-Based Approach for Software Fault Prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47*(5), 826–837. doi: 10.1109/tsmc.2016.2521840

35. Mehmood, I., Shahid, S., Hussain, H., Khan, I., Ahmad, S., Rahman, S., Ullah, N., & Huda, S. (2023). A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning. *IEEE Access, 11*, 63579–63597. doi: 10.1109/access.2023.3287326

36. Alsawalqah, H., Hijazi, N., Eshtay, M., Faris, H., Radaideh, A. A., Aljarah, I., & Alshamaileh, Y. (2020). Software Defect Prediction Using Heterogeneous Ensemble Classification Based on Segmented Patterns. *Applied Sciences, 10*(5), 1745. doi: 10.3390/app10051745