Optimizing Software Defect Prediction: A Genetic Algorithm Based Comparative Analysis

Misbah Ali[1]

[1] Department of Computer Science, Virtual University of Pakistan, Pakistan;

*Corresponding Author: Email: talktomisbah.ali@gmail.com

Abstract: Software quality assurance is a crucial activity during the initial stages of the software development life cycle. Various frameworks have been developed over the past two decades to ensure software quality. By predicting defective modules at the initial stages, the resources available for software development can be efficiently used to ensure the timely delivery of good-quality software. Numerous software defect prediction models have been proposed and developed using supervised and unsupervised machine learning methodologies and integrating statistical methodologies. Software metrics contain hidden patterns that can be extracted and utilized to identify defective modules using a machine-learning approach. This study applies a genetic algorithm (GA) to select relevant features vital in predicting defective modules and explores supervised classification techniques by incorporating seven widely used NASA datasets. The three most used classification techniques, decision tree, support vector machine, and naïve Bayes, were selected for the analysis. Precision, accuracy, recall, Matthew's correlation coefficient, F-measure, and receiver operating characteristics were selected as the performance parameters. The results of this study can serve as a baseline for comparing and verifying the results of new models that implement GA for optimal feature selection.

Keywords: Software defect prediction, software metrics, machine learning, classification

1. Introduction

The software quality assurance process consists of several activities. Among them, predicting defective modules in the initial stages is crucial [1] because the resources involved in debugging them in the later stages of software development increase exponentially. Software testing and defect fixing incur significant costs and require numerous resources, including money, manpower, and time [2]. Researchers have extensively discussed this phenomenon over the past two decades. Developing an efficient software defect prediction (SDP) model involves several factors [3], and the most significant factor is selecting the optimal features.

from historical data that play a significant role in effectively predicting defective modules [4,5]. Previous studies conducted by researchers demonstrated the significance of machine learning techniques, including classification and feature selection, in increasing the accuracy of prediction models [6,28,29].

SDP can improve the effectiveness of software quality assurance, resulting in high-quality software while reducing the resources required for defect identification and fixing, such as manpower, money, and time. Machine learning techniques [30] extract hidden features from historical data and assist in predicting defects in the early stages of the software development life cycle (SDLC) [7]. This study provides an analysis of the three most commonly used supervised machine learning classification algorithms, namely decision tree (DT), support vector machine (SVM), and naïve Bayes (NB), implemented on seven publicly available NASA defect datasets. A genetic algorithm (GA) is applied as a preprocessing step to extract the most relevant features. Supervised machine learning techniques [31,32] require labeled or pre-classified data, called training data. These techniques generate rules by being trained on unseen data called test data. This study incorporatesseven clean NASA datasets: CM1, MC1, MC2, KC1, KC3, PC1, and PC2 [8, 9].

This study performed a detailed analysis of supervised classification techniques on benchmark datasets distributed using 10-fold cross-validation. This study aimed to validate the increased accuracy of models developed by researchers using new defect prediction techniques based on GAs.

The remainder of this paper is organized as follows: Related work is discussed in Section 2. The materials and methods used for the experiments in this study are discussed in Section 3. The conclusions derived from the experiments are presented in Section 4. Finally, the results of this study are discussed in Section 5.

2. Related Work

GA is recognized as one of the most robust optimization algorithms [19, 20]. Katoch et al. presented a review emphasizing the importance of GAs in various contexts and suggested future research focusing on fitness functions and hybrid algorithms [33]. Nguyen et al. experimented with a feature selection technique using a genetic search and an encoder (E-D) model with LSTM to forecast air pollution particulate matter PM2.5 [21]. The E-D model outperformed the other methods and achieved an improved accuracy. MATLOOB [22] developed an SDP model using

multilayer feed-forward neural networks and stacking as the ensemble technique. Various search methods were implemented for feature selection, with best-first search, greedy stepwise search, and genetic search achieving notable accuracy values on the NASA datasets.

Iqbal et al. analyzed twelve datasets obtained from NASA to compare the performance of several classifiers, including the DT, NB, and K-Nearest neighbor (KNN). The study revealed that accuracy and receiver operating characteristic **(**ROC) are not preferred performance measures because of their inability to handle class-distribution imbalances. The results showed varying accuracy values for the different classifiers: NB achieved 78.69, MLP 85.12, RBF 85.76, SVM 84.76, KNN 82.33, KStar 79.72, OneR 84.04, PART 85.48, DT 83.03, and RF 85.23 [10]. Cetiner et al. analyzed ten classifiers using datasets from the PROMISE repository. The findings highlighted that the random forest classifier demonstrated improved performance for the PROMISE datasets [11]. Wang et al. proposed an SDP model using LASSO-SVM on NASA datasets The identification of software defects during the Software development life cycle (SDLC) is an essential step as it involves thorough testing of specific modules. Machine learning (ML) techniques such as feature selection and classification have a significant impact on software defect prediction by enabling the early detection of defects and facilitating the creation of reliable and high-quality software.

[12]. They reduced the dimensionality of the dataset and optimized the SVM parameters using cross validation, resulting in enhanced model performance. Aftab et al. conducted a comparative analysis of four classifiers using a back-propagation strategy for SDP [13]. They utilized a fuzzy layer and found that the Bayesian regularization classifier outperformed other classifiers. Iqbal et al. developed a hybrid approach for feature classification in SDP [14]. They conducted experiments on twelve NASA datasets, employing two individual approaches: one with embedded feature selection and the other without. Both approaches explored bagging and boosting ensemble strategies using a random forest as the base classifier, resulting in improved accuracy. Balogun et al. explored the filter feature ranking (FFR) and various filter subset selection (FSS) techniques using five NASA datasets [15]. Best-first search had a significant impact on the prediction accuracy of FFR, whereas the feature selection strategy generated better overall results. Kondo et al. investigated eight feature selection techniques by using supervised and unsupervised learning models [16]. Feature selection improved the performance of both models, with neural

network-based techniques demonstrating better results for unsupervised learning, and consistency- and correlation-based methods working well for supervised learning. Xiaolong et al. explored 46 feature selection techniques with NB and DT algorithms using multiple public repositories and datasets [7]. They concluded that choosing the best feature selection method depends upon the selected classifier and dataset characteristics. Balogun et al. investigated rank-aggregation-based multifilter feature-selection techniques using NB and DT algorithms on NASA datasets [17]. The results indicated that combining filter rank techniques led to improved prediction accuracy for defect prediction. Yucalar et al. combined multiple classifiers to enhance the prediction accuracy of SDP systems [18]. The selected predictors achieved notable values regarding area under curve (AUC) and F-measure.

This study aimed to comprehensively analyze three frequently employed classifiers: DT, SVM, and NB. As a preprocessing step, a GA was implemented to extract the most appropriate features for SDP.

3. Materials and Methods

This study performed a comparative analysis based on genetic algorithm using seven NASA defect datasets for SDP. A genetic algorithm (GA) was employed for feature selection in the preprocessing step. This algorithm mimics the process of natural selection to optimize the selection of relevant features for a particular task. The genetic algorithm starts with a population of potential feature subsets and iteratively evolves them through genetic operations like crossover and mutation. Each subset's fitness is evaluated based on its performance in a predefined evaluation metric. The fittest subsets are selected for reproduction, producing offspring with characteristics inherited from their parents. This process continues for several generations, gradually improving the feature subsets' performance. Ultimately, the genetic algorithm identifies the most suitable feature subset for the given task [20, 21]. The optimal feature selection process using genetic algorithm is shown in Figure 1.

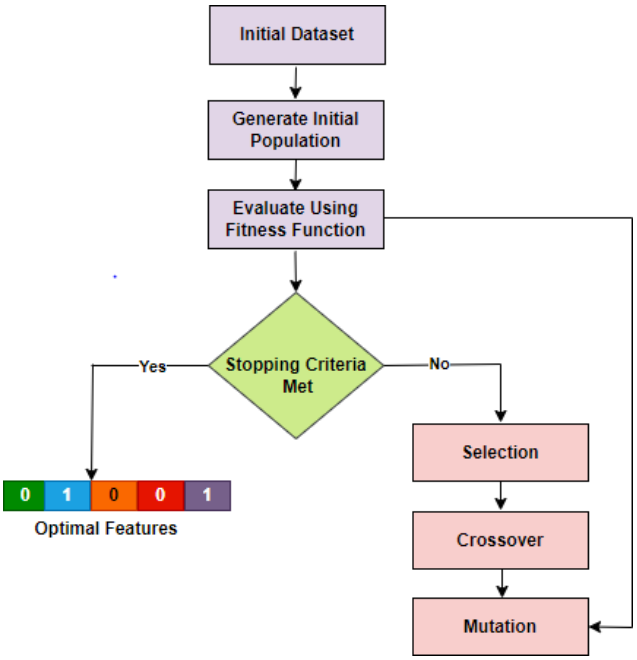

Figure 1. Feature selection using genetic algorithm

The extracted optimal features were integrated into three classification algorithms: DT, SVM, and NB. Decision tree (DT) is a renowned classifier that works effectively by capturing complex relationships between software metrics and defect occurrences. It provides interpretable rules for identifying potential defects, allowing developers to prioritize and allocate resources accordingly [23]. A support Vector Machine (SVM) is an effective classifier that separates defective and non-defective software modules by mapping software metrics into higher-dimensional feature spaces. SVM can create optimal decision boundaries, resulting in accurate defect classification and prediction [2]. Naïve Bayes is a legacy algorithm that has demonstrated its effectiveness in software defect prediction due to its simplicity and computational efficiency. It leverages the probabilistic framework to estimate the likelihood of defects based on software metrics, making it suitable for handling large datasets. Despite its assumption of feature independence, Naive Bayes can still achieve reliable defect prediction results in many software engineering contexts [14].

The NASA datasets in this study contained historical software data with various features and labeled classes. The datasets contained either Y or N as independent attributes, where Y represented a defective instance, and N represented a non-defective instance. Seven clean NASA

datasets were selected for this study: CM1, KC1, KC3, MC1, MC2, PC1, and PC2, as listed in Table 1.

Based on the study of cleaned NASA datasets, two versions were generated: D' and D''. D' contained inconsistent data, and D'' contained consistent and unique instances [8]. This research incorporates D'' clean version of NASA datasets. Researchers have widely used these datasets for defect prediction purposes [23, 24]. The cleaning standards are listed in Table 2. The process flow for the defect prediction process is shown in Figure 2.

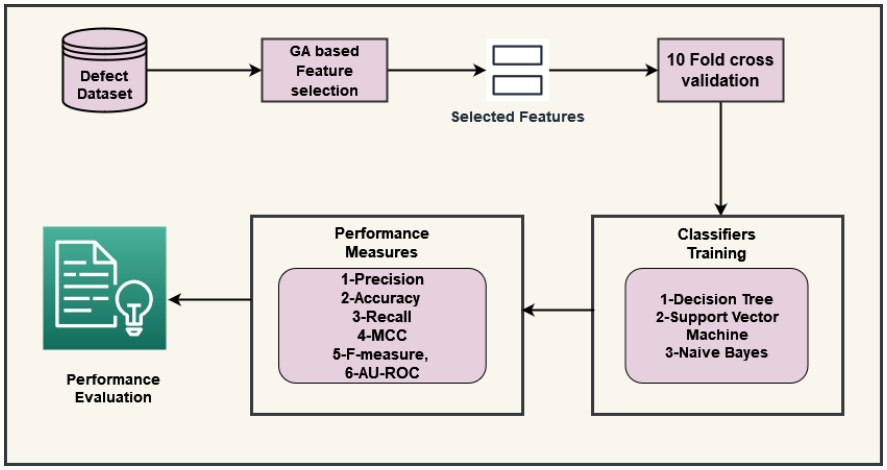

Figure 2. Process flow for software defect prediction

Experiments were performed using the WEKA tool, an extensively used data mining tool with graphical user interface (GUI), created using the JAVA programming language at the University of Waikato, New Zealand.

The description of the datasets is given in Table 1 as follows:

Table 1. NASA datasets-cleaned version

Dataset	Language	Modules	Attributes	GA-based attributes
CM1	C	327	38	13
KC1	C++	1162	22	7
KC3	JAVA	194	40	7
MC1	C++	1952	39	5
MC2	C	124	40	12

PC1	C	679	38	14
PC2	C	722	37	14

The cleaning standards of the employed datasets are presented in Table 2.

Table 2. Cleaning standards [20], [36]

Case No	Standard Type	Description
Case 1	Identical Cases	Instances Containing duplicate values for all metrics along with class label
Case 2	Inconsistent Cases	Instances that meet Case 1, but class labels differ
Case 3	Cases with missing values	Instances containing one or more missing values
Case 4	Cases with inconsistent attribute values	Instances violating referential integrity constraint
Case 5	Cases with doubtful values	Instances violating integrity constraint

4. Results and Discussion

This section presents the performance evaluation of the targeted classifiers that is analyzed using various performance measures. Figure 2 shows the confusion matrix created to derive the performance measures for analysis and evaluation. The confusion matrix contains the following attributes.

TP – True positive: defective modules predicted as defective

FN – False negative: defective modules predicted as non-defective

FP – False positive: non-defective modules predicted as defective

TN – True negative: non-defective modules predicted as non-defective

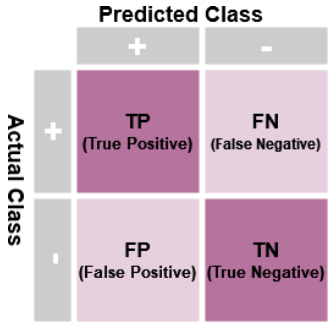

Figure 2. Confusion matrix

The following measures have been derived from the confusion matrix for evaluation purpose: precision, accuracy, recall, Matthew's correlation coefficient (MCC), F-measure, and ROC have been selected as performance parameters.

Precision: It denotes the ratio of true positives to the sum of true and false positives.

$$Precison = \frac{(TP)}{(TP) + (FP)} \quad (1)$$

Accuracy: It presents the extent to which the prediction is accurate [13,34].

$$Accuracy = \frac{(TP) + (TN)}{(TP) + (TN) + (FP) + (FN)} \quad (2)$$

Recall: It expresses the ratio of true positive instances to the sum of the true positive and false negative instances [13].

$$Recall = \frac{(TP)}{(TP) + (FN)} \quad (3)$$

MCC: It is the fraction of actual and predicted class after training the model for classification on the datasets, and it ranges from -1 to +1. The obtained results that are closer to +1 produce higher model accuracy, and the results that are near 0 or less than 0 do not show remarkable performance [23].

$$MCC = \frac{(TN) * (TP) - (FP) * (FN)}{\sqrt{(FP + TP) * (FN + TP) * (TN + FP) * (TN + FN)}} \quad (4)$$

F-Measure: This is derived from precision and recall [13].

$$F - Measure = \frac{precision * recall * 2}{(precision + recall)} \quad (5)$$

ROC curves show the efficiency of a performance measure in distinguishing between faulty and non-faulty instances, calculated using the true positive rate (TPr) and false positive rate (FPr) [26, 27].

$$ROC = \frac{1 + TPr - FPr}{2} \quad (6)$$

The tool used in this study, WEKA provides the facility to derive all the above-listed performance measures. The results calculated using these performance measures are listed in Tables 3 and 11. "?" denotes that the dataset has an imbalance class distribution issue; therefore, few performance parameters are undefined. The most significant values obtained for these parameters are indicated in bold.

Table 3 presents the results of the experiments performed using the CM1 dataset. NB produced better results in terms of precision against the non-defective (N) class, whereas SVM provided better recall and F-measure for the defective (Y) class.

Table 3. CM1 Experimental Results

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.308	0.444	0.364
	N	**0.941**	0.899	0.920
SVM	Y	0.908	**1.000**	**0.952**
	N	?	0.000	?
DT	Y	0.500	0.111	0.182
	N	0.917	0.908	0.881

The experimental results obtained from the KC1 dataset are listed in Table 4. NB yielded better results in terms of precision for the defective (Y) class, whereas SVM yielded better results for the non-defective (N) class in terms of recall. DT generated good results for the F-measure against the non-defective (N) class.

Table 4. KC1 Experimental results

Classifier	Class	Precision	Recall	F-Measure
NB	Y	**0.801**	0.371	0.420
	N	0.485	0.865	0.832
SVM	Y	0.747	0.022	0.043
	N	0.400	**0.988**	0.851
DT	Y	0.786	0.258	0.354
	N	0.561	0.931	**0.852**

Table 5 presents the experimental results obtained using the KC3 dataset. NB provided better results for the non-defective (N) class against the precision parameter. For the recall and F-measure, SVM and DT provided good results for the non-defective (N) class.

Table 5. KC3 Experimental Results

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.300	0.300	0.300

SVM	N	**0.854**	0.854	0.854
	Y	?	0.000	?
DT	N	0.828	**1.000**	**0.906**
	Y	?	0.000	?
	N	0.828	**1.000**	**0.906**

Table 6 presents the experimental results obtained using the MC1 dataset.

Both SVM and DT produced better precision, recall, and F-measure results for the non-defective (N) class.

Table 6. MC1 Experimental Results

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.143	0.286	0.190
	N	0.982	0.958	0.970
SVM	Y	?	0.000	?
	N	**0.976**	**1.000**	**0.988**
DT	Y	?	0.000	?
	N	**0.976**	**1.000**	**0.988**

Table 7 presents the experimental results obtained using the MC2 dataset. NB produced good results in terms of precision for the defective (Y) class, whereas it produced good results for the f-f-measure against the non-defective (N) class. However, SVM produced good results for the non- defective (N) class regarding recall.

Table 7. MC2 Experimental Results

Classifier	Class	Precision	Recall	F-Measure
NB	Y	**0.833**	0.385	0.526
	N	0.742	0.958	**0.836**
SVM	Y	?	0.000	?
	N	0.649	**1.000**	0.787
DT	Y	0.500	0.154	0.235

	N	0.667	0.917	0.772

Table 8 lists the experimental results obtained on the PC1 dataset. NB obtained good results in terms of precision for the non-defective (N) class. SVM produced good results in terms of recall for the non-defective (N) class, whereas the DT generated promising results in terms of the f-measure for the non-defective (N) class.

Table 8. PC1 Experimental results

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.250	0.700	0.368
	N	**0.983**	0.892	0.935
SVM	Y	?	0.000	?
	N	0.951	**1.000**	0.975
DT	Y	0.667	0.400	0.500
	N	0.970	0.990	**0.980**

The experimental results obtained on the PC2 dataset are presented in Table 9. The SVM and DT showed good precision, recall, and F-measure results for the non-defective (N) class.

Table 9. PC2 Experimental Results

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.000	0.000	0.000
	N	0.976	0.953	0.964
SVM	Y	0.000	?	0.000
	N	**0.977**	**1.000**	**0.988**
DT	Y	?	0.000	?
	N	**0.977**	**1.000**	**0.988**

The models' performances in terms of accuracy are listed in Table 10. SVM produced achieved higher accuracy on four datasets: KC3, MC1, MC2, and PC2. DT

produced good results on five datasets: KC3, MC1, MC2, PC1, and PC2. On the other hand, NB did not achieve high accuracy for any of the seven datasets. The results obtained regarding MCC and AU-ROC are listed in Tables 11 and 12, respectively. This is a sensitive evaluation measure, as it does not produce results for many datasets due to class imbalance problems. NB produced good results on two datasets, MC2 and PC1, whereas DT did not achievehigh results on any of the seven datasets. SVM also did not achieve satisfactory results on any of the datasets. These results indicate that all performance measures are critical when a class- imbalance issue exists, as they do not produce any results while implementing the classification algorithms. The AU-ROC area and accuracy were not influenced by the class imbalance issue in the datasets, making them non-critical of the problem. Therefore, researchers should use other evaluation measures, such as recall and F-measure, which respond to class imbalance issues. MCC is a highly critical measure because it responds the most to the class imbalance problem. Although SVM and DT perform well in binary classification, the classifier's results are tool dependent. The tool used in this study encountered challenges while processing specific NASA datasets, particularly in the case of Support Vector Machine (SVM) and Decision Tree (DT) algorithms. The issue stemmed from a class imbalance within these datasets,leading to non-computed results. As a result, the outcomes for these scenarios are undefined and have been denoted as "?" in our analysis.

These techniques were discussed in [25] to address class-imbalance issues. This study incorporated the cleaned version D'' of the NASA datasets, and consequently, no attention was given to the class-imbalance issue.

Table 10. Performance in terms of accuracy

Dataset	NB	SVM	DT
CM1	85.7143	90.8163	**90.8163**
KC1	73.9255	74.212	75.9312
KC3	75.8621	82.7586	82.7586
MC1	94.198	97.6109	97.6109
MC2	75.6757	64.8649	64.8649
PC1	88.2353	95.098	96.0784

PC2	93.0876	97.6959	97.6959

Table 11. Performance in terms of MCC

Dataset	NB	SVM	DT
CM1	0.292	?	0.204
KC1	0.260	0.040	0.256
KC3	0.154	?	?
MC1	0.175	?	?
MC2	**0.444**	?	0.108
PC1	**0.498**	?	0.371
PC2	-0.034	?	?

Table 12. Performance in terms of AU-ROC

Dataset	NB	SVM	DT
CM1	0.720	0.500	0.500
KC1	0.693	0.505	0.595
KC3	0.769	0.500	0.619
MC1	0.832	0.500	0.500
MC2	0.833	0.500	0.535
PC1	0.930	0.500	0.873
PC2	0.779	0.500	0.500

5. Conclusions

Over the past twenty years, one of the emergent fields in software engineering has been SDP using classification algorithms provided by machine learning. Using SDP, software with better-quality can be delivered using fewer resources, including money, manpower, and time. This study performed a GA-based detailed performance analysis on seven cleaned versions of NASA defect datasets. The application of GA for attribute selection in SDP has demonstrated remarkable effectiveness and has made significant contributions to the field. This study aimed to analyze the efficiency of classification algorithms by selecting the most relevant features, and the findings highlight that the effectiveness of the GA in achieving this objective is significant. Three

Classification algorithms were implemented: DT, NB, and SVM. The effectiveness of the classifiers was analyzed using precision, accuracy, recall, MCC, F-measure, and ROC. The results indicate that the AU-ROC area and accuracy did not respond to the class imbalance issue and, therefore, cannot be considered as efficient performance measures. However, MCC, precision, F-Measure, and recall respond to the class imbalance problem. This study serves as a foundational benchmark for researchers, as any new technique proposed for SDP can be compared and validated with the technique used in this research. In the future, the scope of this research can be enhanced by addressing the class imbalance issue and analyzing the performance of classification algorithms using ensemble learning techniques.

References

1. Shafiq, M., Alghamedy, F. H., Jamal, N., Kamal, T., Daradkeh, Y. I., & Shabaz, M. (**2023**). Scientific programming using optimized machine learning techniques for software fault prediction to improve software quality. IET Software, sfw2.12091.
2. Wang, K., Liu, L., Yuan, C., & Wang, Z. (**2021**). Software defect prediction model based on LASSO–SVM. Neural Computing and Applications, 33(14), 8249–8259.
3. Zhu, K., Ying, S., Zhang, N., & Zhu, D. (**2021**). Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network. Journal of Systems and Software, 180, 111026.
4. S. Moustafa, M. Y. ElNainay, N. El Makky, and M. S. Abougabal, ―Software bug prediction using weighted majority voting techniques, Alexandria Eng. J. **2018**, vol. 57, no. 4, pp. 2763–2774.
5. Singh, S., & Haider, T. U. (**2022**). Selection of best feature reduction method for module-based software defect prediction. Journal of Physics: Conference Series, 2273(1), 012002.
6. Yang, Z., Jin, C., Zhang, Y., Wang, J., Yuan, B., & Li, H. (**2022**). Software Defect Prediction: An Ensemble Learning Approach. Journal of Physics: Conference Series, 2171(1), 012008.
7. Xiaolong, X., Wen, C., & Xinheng, W. (**2021**). RFC: A feature selection algorithm for software defect prediction. Journal of Systems Engineering and Electronics, 32(2), 389–398.

8. M. Shepperd, Q. Song, Z. Sun, and C. Mair, ―Data quality: Some comments on the NASA software defect datasets, ‖ IEEE Trans. Softw. Eng. **2013**, vol. 39, no. 9, pp. 1208–1215.
9. ―NASA – Software Defect Datasets [Online]. Available: https://nasa softwaredefectdatasets.wikispaces.com. [Accessed: 01-July-**2023**].
10. Iqbal, A., Aftab, S., Ali, U., Nawaz, Z., Sana, L., Ahmad, M., & Husen, A. (**2019**). Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets. International Journal of Advanced Computer Science and Applications, 10(5).
11. Cetiner, M., & Sahingoz, O. K. (**2020**). A Comparative Analysis for Machine Learning based Software Defect Prediction Systems. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-7.
12. Rath, S. K., Sahu, M., Das, S. P., Bisoy, S. K., & Sain, M. (**2022**). A Comparative Analysis of SVM and ELM Classification on Software Reliability Prediction Model. Electronics, 11(17), 2707.
13. Sh. Daoud, M., Aftab, S., Ahmad, M., Adnan Khan, M., Iqbal, A., Abbas, S., Iqbal, M., & Ihnaini, B. (**2022**). Machine Learning Empowered Software Defect Prediction System. Intelligent Automation & Soft Computing, 31(2), 1287–1300.
14. Iqbal, A., Aftab, S., Ullah, I., Salman Bashir, M., & Anwaar Saeed, M. (**2019**). A Feature Selection based Ensemble Classification Framework for Software Defect Prediction. International Journal of Modern Education and Computer Science, 11(9), 54-64.
15. Balogun, A. O., Basri, S., Abdulkadir, S. J., & Hashim, A. S. (**2019**). Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach. Applied Sciences, 9(13), 2764.
16. Kondo, M., Bezemer, C.-P., Kamei, Y., Hassan, A. E., & Mizuno, O. (**2019**). The impact of feature reduction techniques on defect prediction models. Empirical Software Engineering, 24(4), 1925-1963.
17. Balogun, A. O., Basri, S., Mahamad, S., Abdulkadir, S. J., Capretz, L. F., Imam, A. A., Almomani, M. A., Adeyemo, V. E., & Kumar, G. (**2021**). Empirical Analysis of Rank

Aggregation-Based Multi-Filter Feature Selection Methods in Software Defect Prediction. Electronics, 10(2), 179.

18. Yucalar, F., Ozcift, A., Borandag, E., & Kilinc, D. (**2020**). Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability. Engineering Science and Technology, an International Journal, 23(4), 938–950.
19. Nguyen, M. H., Le Nguyen, P., Nguyen, K., Le, V. A., Nguyen, T.-H., & Ji, Y. (**2021**). PM2.5 Prediction Using Genetic Algorithm-Based Feature Selection and Encoder-Decoder Model. IEEE Access, 9, 57338–57350.
20. Hamdia, K. M., Zhuang, X., & Rabczuk, T. (**2021**). An efficient optimization approach for designing machine learning models based on genetic algorithm. Neural Computing and Applications, 33(6), 1923-1933.
21. Katoch, S., Chauhan, S. S., & Kumar, V. (**2021**). A review on genetic algorithm: Past, present, and future. Multimedia Tools and Applications, 80(5), 8091-8126.
22. Matloob, F. (**2020**). *Software defect prediction model using multi-layer feed forward neural networks* [Thesis, Virtual University of Pakistan]. https://vspace.vu.edu.pk/details.aspx?id=342.
23. Alazba, A., & Aljamaan, H. (**2022**). Software Defect Prediction Using Stacking Generalization of Optimized Tree-Based Ensembles. Applied Sciences, 12(9), 4577. https://doi.org/10.3390/app12094577
24. Alkhasawneh, M. S. (2022). Software Defect Prediction through Neural Network and Feature Selections. Applied Computational Intelligence and Soft Computing, 2022, 1–16. https://doi.org/10.1155/2022/2581832
25. Liu, Y., Zhang, W., Qin, G., & Zhao, J. (**2022**). A comparative study on the effect of data imbalance on software defect prediction. Procedia Computer Science, 214, 1603-1616. https://doi.org/10.1016/j.procs.2022.11.349
26. Liu, W., Wang, B., & Wang, W. (**2021**). Deep Learning Software Defect Prediction Methods for Cloud Environments Research. Scientific Programming, 2021, 1-11. https://doi.org/10.1155/2021/2323100
27. Pan, C., Lu, M., & Xu, B. (**2021**). An Empirical Study on Software Defect Prediction Using CodeBERT Model. Applied Sciences, 11(11),4793. https://doi.org/10.3390/app11114793
28. Ahmed, F., Asif, M. and Saleem, M., 2023. Identification and Prediction of Brain Tumor Using

VGG-16 Empowered with Explainable Artificial Intelligence. International Journal of Computational and Innovative Sciences, 2(2), pp.24-33.

29. Saleem, M., Khan, M.S., Issa, G.F., Khadim, A., Asif, M., Akram, A.S. and Nair, H.K., 2023, March. Smart Spaces: Occupancy Detection using Adaptive Back-Propagation Neural Network. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-6). IEEE.
30. Athar, A., Asif, R.N., Saleem, M., Munir, S., Al Nasar, M.R. and Momani, A.M., 2023, March. Improving Pneumonia Detection in chest X-rays using Transfer Learning Approach (AlexNet) and Adversarial Training. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-7). IEEE.
31. Abualkishik, A., Saleem, M., Farooq, U., Asif, M., Hassan, M. and Malik, J.A., 2023, March. Genetic Algorithm Based Adaptive FSO Communication Link. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-4). IEEE.
32. Sajjad, G., Khan, M.B.S., Ghazal, T.M., Saleem, M., Khan, M.F. and Wannous, M., 2023, March. An Early Diagnosis of Brain Tumor Using Fused Transfer Learning. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-5). IEEE.