Developing a Machine Learning-based Software Fault Prediction Model using Improved Whale Optimization Algorithm

Hauwa Abubakar [1]*, Kabir Umar Phd [2], Rukayya Auwal [3], Kabir Muhammad [4], Lawan Yusuf [5]

[1] Bayero University Kano; hauwaabubakarmusa93@gmail.Com
[2] Bayero University Kano; ukabir.se@buk.edu.ng
[3] Bayero University Kano; Rukayyaauwal@gmail.Com
[4] Bayero University Kano; kabirmuhammad3@gmail.Com
[5] National Open University of Nigeria; Lyusuf@noun.edu.ng

Abstract: Software fault prediction (SFP) is vital for ensuring software system reliability by detecting and mitigating faults. Machine learning has proven effective in addressing SFP challenges. However, extensive fault data from historical repositories often leads to dimensionality issues due to numerous metrics. Feature selection (FS) helps mitigate this problem by identifying key features. This research enhances the Whale Optimization Algorithm (WOA) by combining truncation selection with a single-point crossover method to enhance exploration and avoid local optima. Evaluating the enhancement on 14 SFP datasets from the PROMISE repository reveals its superiority over the original WOA and other variants, demonstrating its potential for improved SFP.

Keywords: software fault prediction; whale optimization; feature selection; machine learning; truncation selection.

1. Introduction

SFP greatly aids in producing high-quality software at a low cost by identifying fault-prone software modules [1]. Machine learning algorithms like decision trees, Bayesian learners, neural networks, support vector machines, and rule-based learning have shown promise, as have soft computing approaches like fuzzy computing, neural networks, evolutionary computing, and swarm intelligence [2].

Feature selection is frequently used to improve the SFP performance of machine learning (ML) algorithms, intending to increase data processing effectiveness and avoid algorithmic error [3]. This is often done using metaheuristic algorithms like genetic algorithms and particle swarm optimization [4]. Among these metaheuristic approaches, the Whale Optimization Algorithm (WOA) has emerged as a promising choice for feature selection. However, WOA is susceptible to local optima trapping, a challenge in large datasets.

This study addresses the local optima problem in WOA for feature selection in SFP by introducing the truncation selection method. Building on recent advancements in WOA variants [5], the research investigates the effectiveness of truncation selection within the context of WOA selection enhancement. This novel approach aims to improve WOA's performance in SFP, offering a potential solution to the local optima challenge.

In summary, the research aims to contribute to the field of software fault prediction by leveraging metaheuristic algorithms, specifically WOA, in combination with the novel truncation selection method, building upon previous advancements to address local optima challenges and enhance the effectiveness of machine learning algorithms in software fault prediction.

2. Review of Related Work

A comprehensive review of literature related to Software Fault Prediction (SFP), Machine Learning (ML)-based SFP, feature selection, and Meta heuristic algorithms for software fault prediction is presented. This review aims to provide a foundational understanding of existing knowledge in this field to support the development and evaluation of the proposed methodology.

A study by [4] brings to the fore the traditional techniques used in SFP, encompassing software matrices, Soft Computing (SC), and Machine Learning (ML). While these approaches have significantly contributed to early fault prediction and the development of dependable software, they still have limitations in predicting certain types of faults. Additionally, they might be time-consuming, particularly when applied to complex software projects, leading to potentially diminished software testing effectiveness.

Some examples of ML-based SFP techniques include; [6], [7], [8], [9], and [10].

Feature selection (FS) has become a significant step in data mining in general and machine learning in particular since it helps to clean data by removing noisy, irrelevant, and redundant features [11]. A study by [12] developed a novel FS called evolving populations with mathematical diversification (FS-EPWMD), which uses arithmetic diversification among candidate solutions to avoid the local optimum. The guiding principle of populations evolving through crossover and mutation is the survival of the fittest. The results demonstrated that FS-EPWMD outperforms other models.

Swarm intelligence (SI) is one of computational intelligence techniques that are used to resolve complicated problems [13].Swarm intelligence algorithms have demonstrated excellent performance in lowering the running time and addressing the FS problem. For instance, In order to solve the FS problem in the area of software fault prediction, [14] proposed the island model to improve the BMFO. The EBMFO with SVM classifier produced the best results overall. These findings show that the suggested model can be a useful predictor for the software fault issue.

3. Proposed Work

This section presents the research workflow and outlines the proposed methodology employed in the study, covering the enhancement of the algorithm and subsequent performance evaluation.

3.1. The Research Workflow

The proposed model work in four phases namely; literature review, methodology, implementation and evaluation and results phase. The flow begins from literature review down to evaluation and results and each phase is represented with its activities. Figure 1 represent the entire work flow of the model.

In the first phase, we conducted a comprehensive literature review, exploring 42 articles related to Software Fault Prediction (SFP), ML-based SFP, Feature Selection (FS), and Selection schemes. The Second phase highlighted the main components of the proposed ML-based SFP model and the enhanced WOA. ML-based SFP handles prediction, while the enhanced WOA aims to enhance its performance. In the third phase, we implemented the ML-based SFP model using Google Colab and replicated baseline work in the same environment for comparison. To classify FS problems, four well-known classifiers were employed; Support Vector Machine (SVM), Decision Tree (DT), Linear Discriminant Analysis (LDA), and K-Nearest Neighbors (KNN). In the final phase, the proposed model's performance was evaluated using the following metrics; Area Under Curve (AUC), precision, recall, F1 score, and accuracy. Cross-validation technique was also used to assess the performance of the model, where 80% of the dataset was used for training and 20% for testing.

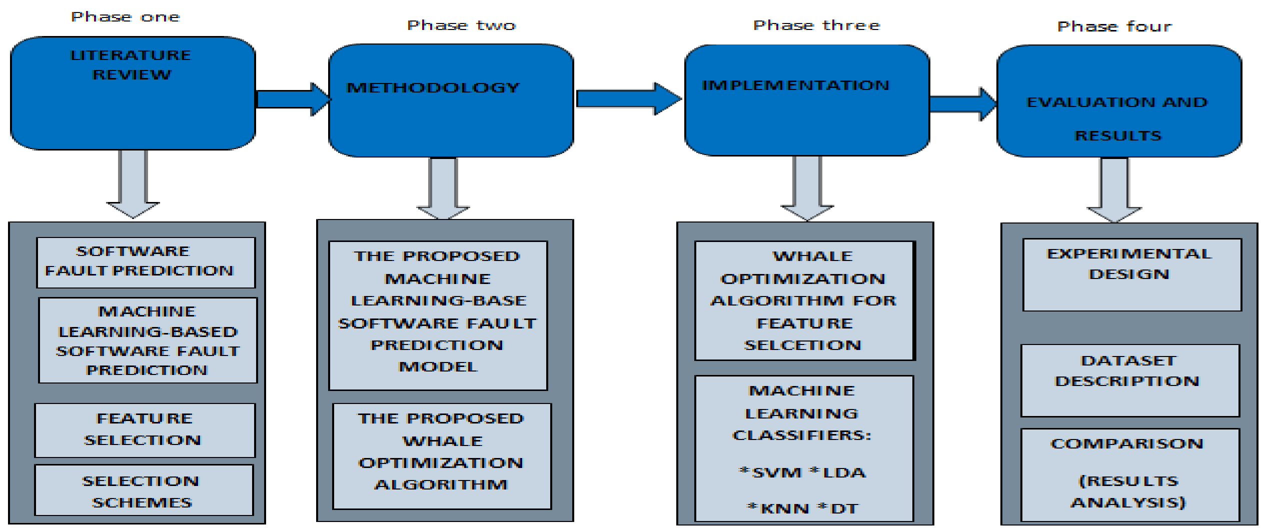
Phase one
LITERATURE REVIEW
SOFTWARE FAULT PREDICTION
MACHINE LEARNING-BASED SOFTWARE FAULT PREDICTION
FEATURE SELECTION
SELECTION SCHEMES
Phase two
METHODOLOGY
THE PROPOSED MACHINE LEARNING-BASE SOFTWARE FAULT PREDICTION MODEL
THE PROPOSED WHALE OPTIMIZATION ALGORITHM
Phase three
IMPLEMENTATION
WHALE OPTIMIZATION ALGORITHM FOR FEATURE SELCETION
MACHINE LEARNING CLASSIFIERS:
*SVM *LDA
*KNN *DT
Phase four
EVALUATION AND RESULTS
EXPERIMENTAL DESIGN
DATASET DESCRIPTION
COMPARISON
(RESULTS ANALYSIS)

Figure 1: The Research Workflow

3.2. The Proposed ML-based SFP Model

This section details the research methodology for ML-based Software Fault Prediction (SFP). Figure 2 presents the proposed model diagram which works in five stages; data collection, data pre-processing, feature selection, machine learning classifiers, and evaluation.

Stage 1 involves gathering 14 datasets from the PROMISE dataset repository, with details provided in table 1. In Stage 2, data pre-processing was used to stabilized the dateset into a form suitable for training and validation. Stage 3 employs the Whale Optimization Algorithm (WOA) for feature selection, with a focus on improving its selection scheme using truncation selection. The stage 4 deploys four ML classifiers (DT, KNN, LDA, SVM) to predict software faults, enhancing the WOA's performance in feature selection. In the final stage we evaluated the model using the evaluation metrics mentioned above.

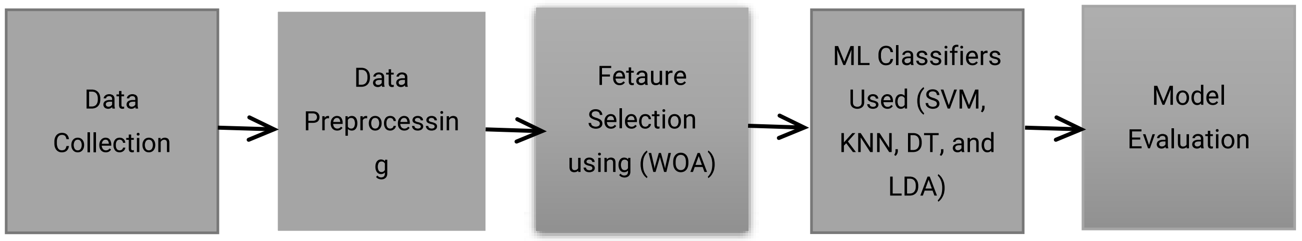
Data Collection
Data Preprocessin g
Fetaure Selection using (WOA)
ML Classifiers Used (SVM, KNN, DT, and LDA)
Model Evaluation

Figure 2: The Proposed Model

3.3 The Proposed Enhanced Whale Optimization Algorithm

This work employs the Whale Optimization Algorithm (WOA) and enhances it by incorporating truncation selection to improve its selection scheme, as illustrated in Figure 3. In Figure 3, we addressed the challenge of a stuck best solution in local optima, where we proposed a solution involving the combination of the Whale Optimization Algorithm (WOA), truncation selection, and a single-point crossover approaches. This enhancement primarily focuses on improving the selection part of WOA, where employed truncation selection. In the truncation selection process, individuals are ranked based on their fitness values, and only the best-performing individuals are chosen as parents for the next generation. This selection is governed by a primary truncation selection parameter known as the TRS threshold, which can vary between 50% and 10%.

It determines the percentage of the population that will serve as parents. Individuals falling below this threshold are eliminated as they are considered unfit for reproduction. The truncstion selection processes are indicated below.

- The population is sorted based on each individual's evaluation scores.
- The poorest-performing fraction of the population is removed.
- The eliminated individuals are replaced with variations of individuals from the top-performing fraction, with each of the best individuals creating one offspring. These offspring subsequently replace one of the previously removed, lower-performing individuals in the population.

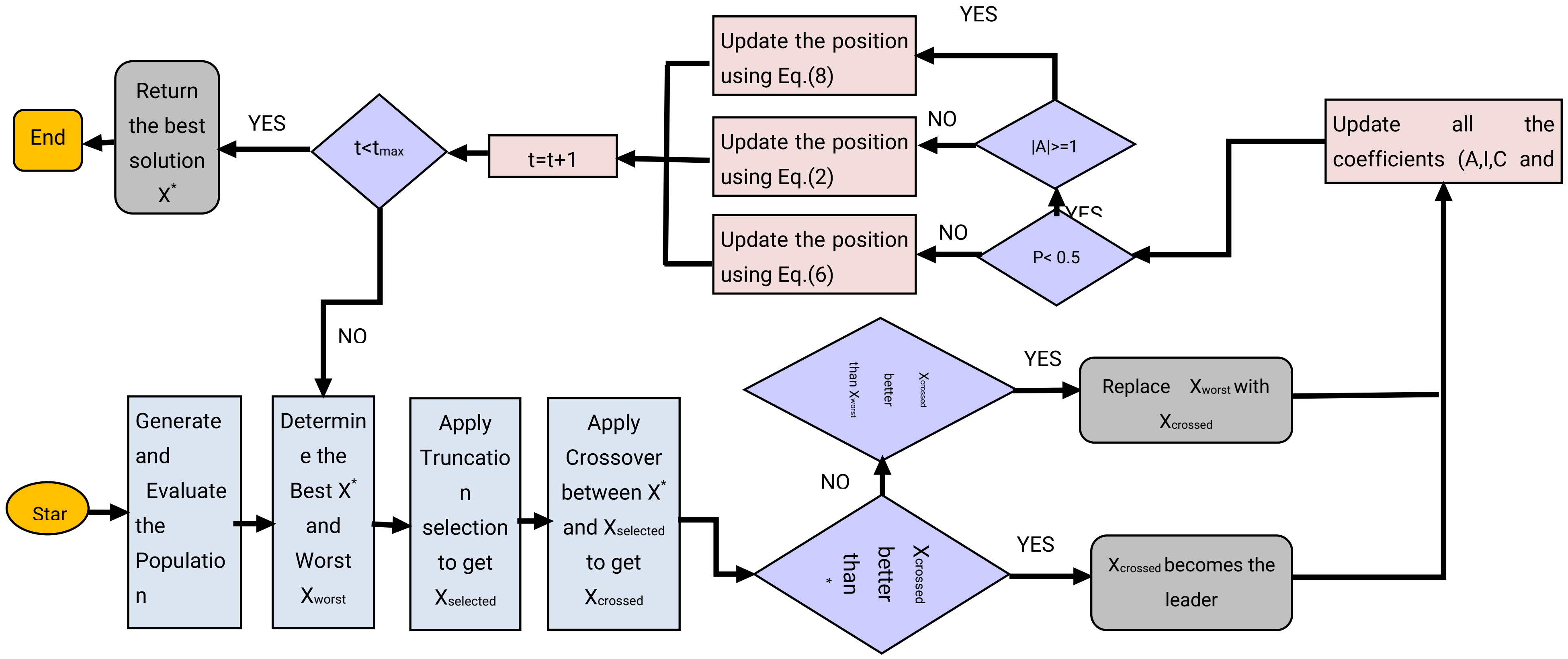
Star
Generate and Evaluate the Population
Determine the Best X* and Worst X_worst
Apply Truncation selection to get X_selected
Apply Crossover between X* and X_selected to get X_crossed
X_crossed better than *
YES
X_crossed becomes the leader
NO
X_crossed better than X_worst
YES
Replace X_worst with X_crossed
Update all the coefficients (A,I,C and
P< 0.5
YES
|A|>=1
YES
Update the position using Eq.(8)
NO
Update the position using Eq.(2)
NO
Update the position using Eq.(6)
t=t+1
t<t_max
NO
YES
Return the best solution X*
End

Figure 3: flowchart of the proposed enhanced WOA

4. Discussion of Results

This section devoted to the presentation and of the experimental results.

4.1 Implementation Environment

The work was implemented on Google Colab environment, using Python, Pandas and Tensor Flow libraries.

4.2 Proposed Model Performance

The proposed enhance WOA was evaluated by iteratively enhancing a population of candidate solutions using truncation selection method, crossover, and coefficient update procedures. The results obtained from the experiments revealed promising outcomes as shown in Table 2.

In table 2, Ant-1.7 performed best with an accuracy of 0.834, while Lucene-2.2 scored the lowest at 0.534. Log4j-1.2 had the highest precision (0.789), Log4j-1.0 the highest recall (0.727), and

Log4j-1.0 also achieved the highest F1 score (0.643) and AUC (0.790). Conversely, Camel-1.6 consistently had lower scores across these metrics.

Table 2: Truncation based-WOA (TRBWOA) with SVM

Datasets	Accuracy	Precision	Recall	F1 score	AUC
Ant-1.7	0.839	0.650	0.533	0.519	0.687
Camel-1.2	0.706	0.600	0.102	0.239	0.604
Camel-1.4	0.817	0.04	0.05	0.561	0.493
Camel-1.6	0.877	0.433	0.224	0.440	0.605
Jedit-3.2	0.763	0.900	0.528	0.680	0.699
Jedit-4.0	0.922	0.800	0.386	0.421	0.734
Jedit-4.1	0.730	0.875	0.486	0.513	0.819
Jedit-4.2	0.978	0.0	0.500	0.500	0.892
Log4j-1.0	0..740	0.433	0.661	0.322	0.736
Log4j-1.1	0.829	0.725	0.625	0.625	0.705
Log4j-1.2	0.902	0.902	0.950	0.948	0.500
Lucene-2.0	0.767	0.300	0.873	0.518	0.676
Lucene-2.2	0.580	0.650	0.850	0.701	0.850
Lucene-2.4	0.806	0.588	0.753	0.740	0.849

In table 3 we summarized the dataset performance. Ant-1.7 had the best accuracy (0.832), while Camel-1.6 scored lowest (0.878). Camel-1.4 achieved perfect precision (1.0), while Camel-1.6 had the lowest accuracy (0.333). Camel-1.2 had the highest recall (0.504), and Jedit-4.2 had the highest F1 score (0.500) and AUC (0.863). Camel-1.4 had the lowest AUC (0.517) and Camel-1.6 had the lowest F1 score (0.244)

Table 3: Truncation based-WOA (TRBWOA) with KNN

Datasets	Accuracy	Precision	Recall	F1 score	AUC
Ant-1.7	0.832	0.581	0.600	0.533	0.745
Camel-1.2	0.715	0.656	0.504	0.275	0.623
Camel-1.4	0.834	1.0	0.333	0.365	0.517
Camel-1.6	0.878	0.333	0.438	0.244	0.605
Jedit-3.2	0.781	0.946	0.523	0.647	0.863
Jedit-4.0	0.874	0.500	0.387	0.463	0.850
Jedit-4.1	0.777	0.827	0.481	0.540	0.732
Jedit-4.2	0.846	1.0	0.450	0.500	0.801
Log4j-1.0	0..919	0.767	0.725	0.400	0.766
Log4j-1.1	0.815	0.950	0.777	0.867	0.625
Log4j-1.2	0.864	0.902	0.433	0.949	0.743
Lucene-2.0	0.902	0.444	0.876	0.381	0.812
Lucene-2.2	0.766	0.756	0.629	0.812	0.600
Lucene-2.4	0.618	0.657	0.753	0.690	0.749

4.3 Results Comparison

Regarding AUC, Table 4 shows that TRBWOA performed better than TBWOA and all other variations when Decision Tree was applied. The TRBWOA did remarkably well in dataset like ant-1.7 with 0.803 while the Random-Based Whale Optimization Algorithm (RBWOA) had the worst performance with log4j-1.2 with 0.486 dataset. Table 4 shows the results comparison of the models.

Table 4: Comparison of results of WOA implemented with different selection schemes with DT classifier in terms of AUC.

Datasets	LRBWOA	RBWOA	PBWOA	TBWOA	SUSBWOA	TRWOA
Ant-1.7	0.690	0.687	0.664	0.688	0.664	0.987
Camel-1.2	0.635	0.609	0.604	0.606	0.612	0.608
Camel-1.4	0.531	0.585	0.582	0.587	0.589	0.555
Camel-1.6	0.593	0.575	0.575	0.567	0.576	0.679
Jedit-3.2	0.803	0.744	0.735	0.736	0.722	0.955
Jedit-4.0	0.569	0.560	0.571	0.550	0.567	0.655
Jedit-4.1	0.569	0.641	0.634	0.625	0.620	0.855
Jedit-4.2	0.782	0.718	0.701	0.725	0.730	0.665
Log4j-1.0	0.777	0.644	0.640	0.653	0.657	0.638
Log4j-1.1	0.562	0.605	0.713	0.704	0.712	0.777
Log4j-1.2	0.597	0.486	0.643	0.660	0.627	0.925
Lucene-2.0	0.504	0.560	0.499	0.496	0.762	0.751
Lucene-2.2	0.533	0.650	0.528	0.551	0.507	0.654
Lucene-2.4	0.642	0.634	0.615	0.634	0.536	0.761

Regarding AUC, Table 5 shown that TRBWOA outperformed all other variations of WOA including the TBWOA when KNN was applied. The TRBWOA performed well in dataset like Jedit-3.2 with 0.863 while the linear ranked-based whale optimization algorithm (LRBWOA) had the worst performance with camel-1.6 with 0.474 dataset. Figure 4 shown the graphical presentation of the results compared.

Table 5: Comparison of results of WOA implemented with different selection schemes with KNN classifier in terms of AUC

Datasets	LRBWOA	RBWOA	PBWOA	TBWOA	SUSBWOA	TRBWOA
Ant-1.7	0.657	0.657	0.682	0.691	0.704	0.745
Camel-1.2	0.524	0.582	0.519	0.525	0.517	0.623
Camel-1.4	0.555	0.499	0.511	0.517	0.511	0.517
Camel-1.6	0.474	0.556	0.496	0.505	0.505	0.605
Jedit-3.2	0.703	0.780	0.735	0.756	0.719	0.863
Jedit-4.0	0.544	0.569	0.552	0.634	0.557	0.850
Jedit-4.1	0.619	0.654	0.650	0.500	0.647	0.732
Jedit-4.2	0.679	0.804	0.662	0.669	0.689	0.801
Log4j-1.0	0.488	0.761	0.607	0.679	0.604	0.766
Log4j-1.1	0.660	0.633	0.691	0.516	0.677	0.625
Log4j-1.2	0.625	0.500	0.520	0.516	0.524	0.743
Lucene-2.0	0.546	0.601	0.545	0.531	0.551	0.812
Lucene-2.2	0.640	0.476	0.593	0.715	0.586	0.600
Lucene-2.4	0.639	0.619	0.573	0.792	0.583	0.749

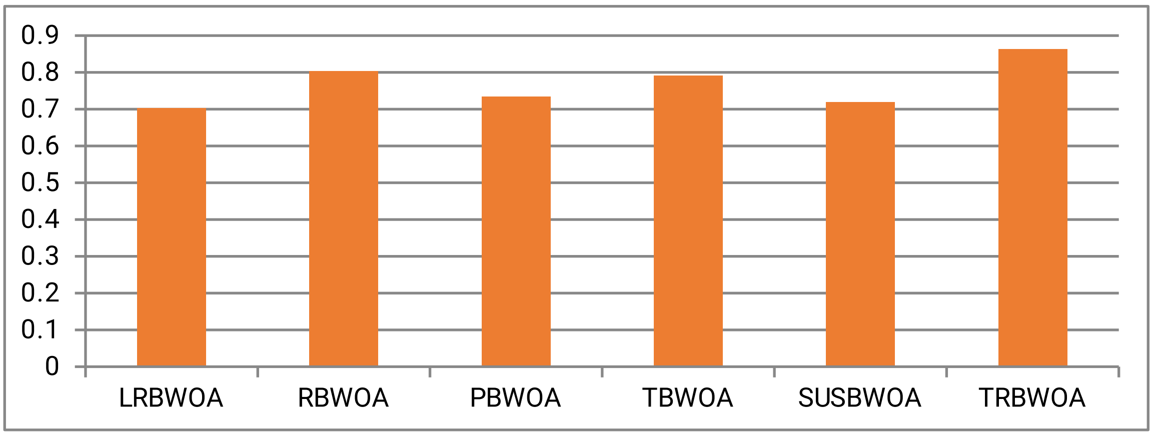

Figure 4: Comparison of results of WOA implemented with different selection schemes with KNN classifier in terms of AUC

Conclusion

Remarkably, the proposed work demonstrated significant advancements over the previous work across all evaluated metrics and datasets. The results showcased the superiority performance of the proposed work, consistently outperforming the previous work in terms of various evaluation measures. These findings affirm the efficiency and effectiveness of the proposed approach in addressing the research objectives and achieving improved outcomes.

References

1. Rathore, S. S., & Kumar, S. (2019). A study on software fault prediction techniques. *Artificial Intelligence Review, 51*(2), 255-327.
2. Singh, P. D., & Chug, A. (2017). *Software defect prediction analysis using machine learning algorithms.* Paper presented at the 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence.
3. Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing, 300*, 70-79.
4. Hassouneh, Y., Turabieh, H., Thaher, T., Tumar, I., Chantar, H., & Too, J. (2021). Boosted whale optimization algorithm with natural selection operators for software fault prediction. *IEEE Access, 9*, 14239-14258
5. Heidari, A. A., Aljarah, I., Faris, H., Chen, H., Luo, J., & Mirjalili, S. (2020). An enhanced associative learning-based exploratory whale optimizer for global optimization. *Neural Computing and Applications, 32*(9), 5185-5211.
6. Bowes, D., Hall, T., & Petrić, J. (2018). Software defect prediction: do different classifiers find the same defects? *Software Quality Journal, 26*(2), 525-552.
7. Pandey, S. K., Mishra, R. B., & Tripathi, A. K. (2021). Machine learning based methods for software fault prediction: A survey. *Expert Systems with Applications, 172*, 114595.
8. Suryadi, A. (2019). Integration of feature selection with data level approach for software defect prediction. *Sinkron: jurnal dan penelitian teknik informatika, 4*(1), 51-57.
9. Bahaweres, R. B., Suroso, A. I., Hutomo, A. W., Solihin, I. P., Hermadi, I., & Arkeman, Y. (2020). *Tackling feature selection problems with genetic algorithms in software defect prediction for optimization.* Paper presented at the 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS).
10. Alkhasawneh, M. S. (2022). Software Defect Prediction through Neural Network and Feature Selections. *Applied Computational Intelligence and Soft Computing, 2022.*
11. Chantar, H., Mafarja, M., Alsawalqah, H., Heidari, A. A., Aljarah, I., & Faris, H. (2020). Feature selection using binary grey wolf optimizer with elite-based crossover for Arabic text classification. *Neural Computing and Applications, 32*(16), 12201-12220.
12. Goyal, S. (2022). Software fault prediction using evolving populations with mathematical diversification. *Soft Computing, 26*(24), 13999-14020.
13. Shrivastava, D., Sanyal, S., Maji, A. K., & Kandar, D. (2020). Bone cancer detection using machine learning techniques *Smart Healthcare for Disease Diagnosis and Prevention* (pp. 175-183): Elsevier.