symmetry

MDPI

Article

Software Defect Prediction Using Wrapper Feature Selection Based on Dynamic Re-Ranking Strategy

Abdullateef Oluwagbemiga Balogun [1,2,*], Shuib Basri [1], Luiz Fernando Capretz [3], Saipunidzam Mahamad [1], Abdullahi Abubakar Imam [1], Malek A. Almomani [4], Victor Elijah Adeyemo [5], Ammar K. Alazzawi [1], Amos Orenyi Bajeh [2] and Ganesh Kumar [1]

1 Department of Computer and Information Science, Universiti Teknologi PETRONAS, Seri Iskandar 32610, Perak, Malaysia; shuib_basri@utp.edu.my (S.B.); saipunidzam_mahamad@utp.edu.my (S.M.); imam.abubakar@utp.edu.my (A.A.I.); ammar_16000020@utp.edu.my (A.K.A.); ganesh_17005106@utp.edu.my (G.K.)
2 Department of Computer Science, University of Ilorin, Ilorin 1515, Nigeria; bajehamos@unilorin.edu.ng
3 Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada; lcapretz@uwo.ca
4 Department of Software Engineering, The World Islamic Sciences and Education University, Amman 11947, Jordan; malek.almomani@wise.edu.jo
5 School of Built Environment, Engineering and Computing, Headingley Campus, Leeds Beckett University, Leeds LS6 3QS, UK; v.adeyemo@leedsbeckett.ac.uk
* Correspondence: abdullateef_16005851@utp.edu.my or balogun.ao1@unilorin.edu.ng

Abstract: Finding defects early in a software system is a crucial task, as it creates adequate time for fixing such defects using available resources. Strategies such as symmetric testing have proven useful; however, its inability in differentiating incorrect implementations from correct ones is a drawback. Software defect prediction (SDP) is another feasible method that can be used for detecting defects early. Additionally, high dimensionality, a data quality problem, has a detrimental effect on the predictive capability of SDP models. Feature selection (FS) has been used as a feasible solution for solving the high dimensionality issue in SDP. According to current literature, the two basic forms of FS approaches are filter-based feature selection (FFS) and wrapper-based feature selection (WFS). Between the two, WFS approaches have been deemed to be superior. However, WFS methods have a high computational cost due to the unknown number of executions available for feature subset search, evaluation, and selection. This characteristic of WFS often leads to overfitting of classifier models due to its easy trapping in local maxima. The trapping of the WFS subset evaluator in local maxima can be overcome by using an effective search method in the evaluator process. Hence, this study proposes an enhanced WFS method that dynamically and iteratively selects features. The proposed enhanced WFS (EWFS) method is based on incrementally selecting features while considering previously selected features in its search space. The novelty of EWFS is based on the enhancement of the subset evaluation process of WFS methods by deploying a dynamic re-ranking strategy that iteratively selects germane features with a low subset evaluation cycle while not compromising the prediction performance of the ensuing model. For evaluation, EWFS was deployed with Decision Tree (DT) and Naïve Bayes classifiers on software defect datasets with varying granularities. The experimental findings revealed that EWFS outperformed existing metaheuristics and sequential search-based WFS approaches established in this work. Additionally, EWFS selected fewer features with less computational time as compared with existing metaheuristics and sequential search-based WFS methods.

Keywords: high dimensionality; re-ranking strategy; software defect prediction; wrapper feature method; Cuckoo search method

Citation: Balogun, A.O.; Basri, S.; Capretz, L.F.; Mahamad, S.; Imam, A.A.; Almomani, M.A.; Adeyemo, V.E.; Alazzawi, A.K.; Bajeh, A.O.; Kumar, G. Software Defect Prediction Using Wrapper Feature Selection Based on Dynamic Re-Ranking Strategy. *Symmetry* **2021**, *13*, 2166. https://doi.org/10.3390/sym13112166

Academic Editor: SeongKi Kim

Received: 12 October 2021
Accepted: 3 November 2021
Published: 12 November 2021

1. Introduction

The software development lifecycle (SDLC) is a formal framework that has been specifically planned and built for the production or development of high-quality software systems. To ensure a timely and reliable software system, the SDLC incorporates gradual steps such as requirement elicitation, software system review, software system design, and software system maintenance, which must be carefully followed and applied [1–3]. However, since the SDLC step-by-step operations are done by human professionals, failures are inevitable. Because of the large scale and dependencies in modules or parts of software systems today, these errors are common and recurring. If not corrected immediately, these errors will result in unreliable computing structures and, ultimately, software failure. That is, the occurrence of errors in information system modules or components will result in flawed and low-quality software systems. Furthermore, vulnerabilities in software systems can irritate end-users and customers when the failed software system does not work as expected after having wasted scarce resources (time and effort) [4–7]. Hence, it is important to consider early prediction and recognition of software flaws before product delivery or during the software development processes. Early detection or prediction of incorrect (defective) modules or components in a software system allows those modules or components to be automatically corrected and available resources to be used wisely [8,9].

At the unit level, testing critical source codes necessitates selecting test data from the input domain, running the source codes with the selected test data, and checking the accuracy of the computed outputs [3]. The symmetric testing strategy is an applicable example. Symmetric testing tries to test source code that does not require an oracle or any formal specification. The permutation relations between source code runs are typically used to automate the testing process. Primarily, symmetric testing uses a combination of automated test data production and symmetries, checking to identify flaws or vulnerabilities in a software system [10]. However, one disadvantage of automated testing methodologies is the difficulty in designing and troubleshooting automated test scripts [11]. Another viable approach is the use of machine learning (ML) methods to determine the defectivity of modules or components in a software system. This approach is known as software defect prediction (SDP). Specifically, SDP is the application of ML methods to software features identified by software metrics to identify faults in software modules or components [12–15]. Several researchers have suggested and applied both supervised and unsupervised ML approaches for SDP [16–21].

Nonetheless, the predictive accuracy of SDP models is entirely dependent on the consistency and inherent design of the software datasets used in their creation. The magnitude and complexities of information systems are closely related to the software metrics used to characterize the consistency and performance of software systems. To put it another way, large and scalable software systems necessitate many software metric structures to deliver functionality that best reflects the output of those software systems [22,23]. In general, software systems with many features are composed of redundant and insignificant features, resulting from the accumulation of software metrics. This can be described as a high dimensionality problem. Several studies have shown that the high dimensionality of software metrics has a negative effect on the predictive performance of SDP models [24,25]. Researchers agree that the feature selection (FS) approach is an effective method for addressing high-dimensionality problems [26–29]. For each SDP operation, these FS methods essentially extract valuable and critical software features from the initial software defect dataset [27,28,30].

The implementation of FS methods will lead to the formation of a subset of features that contains germane and crucial features from a set of irrelevant and excessive features, thus overcoming the high dimensionality of the dataset. The application of FS methods results in the creation of a subset of features containing germane and critical features from a collection of trivial and unnecessary features, thus resolving the dataset's high dimensionality. In other words, FS methods select important features while retaining dataset accuracy. Finally, this solves the issue of the high dimensionality of software defect

datasets. FS methods select prominent features while ensuring the quality of the dataset. In the end, this solves the high dimensionality problem of software defect datasets [30,31]. There are two types of FS methods: filter feature selection (FFS), and wrapper feature selection (WFS). FFS approaches test dataset attributes by using the dataset's underlying computational or predictive properties. Following that, the top-ranked features are selected depending on the predefined threshold score. In contrast to FSS, WFS methods test dataset functionality based on their usefulness in improving the efficiency of underlining classifiers. In other words, WFS chooses features based on classifier results.

However, there are still notable issues with WFS methods. The process of subsets generation in WFS methods largely depends on the search strategy used. A key problem with WFS is how to search into the space of feature subsets. Using an exhaustive search strategy leads to high time complexity, since all possible feature subsets are considered, while a heuristic search strategy does not consider all possible feature subsets, being a stochastic process [28,32–35]. While several WFS methods based on metaheuristic and sequential searches have been proposed and developed, the local maxima stagnation problem and high computational costs due to large search space persist [30–33,36]. WFS approaches have been recognized to have significant computational costs, since the number of executions necessary for feature subset search, assessment, and selection are unknown in advance, which often leads to overfitting of prediction models owing to easy entrapment in local maxima. In the WFS subset evaluator phase, using the right search technique may resolve its entrapment in local maxima.

In addition, finding a way to reduce the evaluation time of WFS, that is, computational cost, while maintaining its performance, is imperative. As a result, this study proposes a novel re-ranking strategy-based WFS method to dynamically and iteratively select features. The proposed enhanced WFS (EWFS) method is based on incrementally selecting features while considering previously selected features in its search space.

The main contributions of this study are as follows:

1. An enhanced wrapper feature selection (EWFS) method based on a dynamic re-ranking strategy was developed.
2. The performance of the proposed EWFS was evaluated and compared with existing sequential and metaheuristic search-based WFS methods.
3. The effectiveness of EWFS as a solution for high dimensionality in SDP was validated.

The remainder of this paper is structured as follows: Reviews on existing related works are presented in Section 2. Details on proposed EWFS and experimental methods are described in Section 3. Experimental results are analyzed and discussed in Section 4, and the research is concluded with highlights of future work in Section 5.

2. Related Works

High dimensionality is a well-known data quality issue that typically harms the predictive capabilities of SDP models. In other words, a wide range of technological features or metrics affects the predictive performance of SDP models. FS methods are used to resolve high dimensionality as a data preprocessing activity by choosing appropriate and irredundant features. In this case, SDP is no exception, and many FS approaches have been used to improve the predictive performance of SDP models.

Wahono, et al. [37] used a metaheuristic-based WFS approach to improve an ensemble-based SDP model. They integrated Particle Swarm Optimization (PSO) and Genetic Algorithm as search methods for the wrapper. Their experimental results showed that the applied WFS method enhanced the predictive performance of the ensemble method.

Specifically, a hybrid metaheuristic search method based on PSO and GA was deployed as a search mechanism in the proposed WFS method. Findings from their study indicated that the proposed WFS can improve the implemented ensemble method's prediction accuracy. This observation indicates that metaheuristic search approaches may be equally as effective as conventional BFS or GSW techniques as search methods in WFS. However, the performance of metaheuristic search methods depends on their hyper-

parameterization. That is, getting the right or appropriate parameters for the metaheuristic-based WFS method is a problem.

Similarly, Song, et al. [38] utilized two WFS approaches in their study: forward-selection and backwards-elimination. Based on their findings, they concluded that both types of WFS had a beneficial impact on SDP models, asserting that there is no discernible distinction between their respective performances. Moreover, their work seems to have a drawback in that it only uses forward selection and backward removal as methods of analysis. Other search approaches, such as metaheuristics, may be as efficient as, if not more efficient than, forward-selection and backwards-elimination in terms of finding relevant results.

Muthukumaran, et al. [39] performed a detailed empirical study on 16 defect datasets using 10 diverse FS techniques. In their analysis, WFS founded on greedy stepwise search (GSS) methods outperformed other FS methods. Rodríguez, et al. [40] studied the impact of FS methods on prediction models in SDP. Particularly, they compared the effectiveness of correlation-based FS (CFS), consistency-based FS (CNS), fast correlation-based filter (FCBF), and WFS methods. The researchers claimed that selecting fewer features from datasets has greater prediction capabilities than the original dataset, and that the WFS approach outperforms the other tested FFS approaches. WFS approaches, on the other hand, are computationally expensive, perhaps as a result of relying on classical exhaustive search techniques. Moreover, testing the chosen features against a different base classifier does not ensure optimality.

Cynthia, et al. [41] looked into how FS methods affected SDP prediction models. The impact of five FS approaches on a set of classifiers was studied. From their results, they deduced that the prediction output of the selected classifiers was impacted (positively) by experimented FS methods. Despite this, the scope of their study (the number of FS methods and datasets used) was limited. Akintola, Balogun, Lafenwa-Balogun and Mojeed [2] investigated filter-based FS techniques on heterogeneous prediction models, focusing on principal component analysis (PCA), correlation-based feature selection (CFS), and filtered subset evaluation classifiers (FSE). They also observed that utilizing FS techniques in SDP is useful since it improves the prediction accuracy of selected classifiers.

Balogun, Basri, Jadid, Mahamad, Al-momani, Bajeh and Alazzawi [26] studied the performance of the WFS technique using several feature search methods. The effectiveness of 13 metaheuristics and 2 sequential search-based WFS systems was specifically examined. According to their experimental observations, WFS based on metaheuristic search approaches beat sequential search-based WFS methods in the majority of conducted experiments. Similarly, Mabayoje, et al. [42] investigated the effect of different WFS methods on prediction performances of SDP models. According to their findings, using WFS approaches may improve the accuracy of SDP model predictions. Furthermore, Ding [43], in their study, applied a GA as a search method in WFS to enhance the prediction performance of an isolation forest classifier (IFC). Findings from their results revealed the effectiveness of the proposed WFS method on implemented IFC. Yet, the computing time of metaheuristic search-based WFS techniques, on the other hand, was rather significant. This can be attributed to the easy trapping of metaheuristic search-based methods in local maxima and its convergence uncertainty.

Overall, WFS techniques are quite effective in reducing the number of data characteristics while simultaneously improving the prediction model's performance. Despite this, the downsides of easily falling into local maxima and large computing costs continue to crop up. Consequently, this study proposes an EWFS method based on a dynamic re-ranking strategy to select relevant and irredundant features in SDP. Specifically, an entropy measure (in this case, Information Gain) is used to rank features, and then the ranked features are passed through an incremental wrapper method using conditional mutual information maximization (CMIM) while considering the initially selected features. CMIM balances and selects features by maximizing their respective mutual information with the class label while minimizing the co-dependency that may exist between or amongst features using a

Multi-Objective Cuckoo search method. This process will reduce the number of wrapper evaluations, as only a few features are considered during each iteration while maintaining or amplifying the prediction performance of the selected features.

3. Methodology

This section presents and discusses the implemented classification algorithms, the proposed EWFS approach, experimented datasets, performance assessment measures, and experimental procedures.

3.1. Classification Algorithm

Decision Tree (DT) and Naive Bayes (NB) algorithms were chosen and used as prediction models in this analysis. The two were chosen because of their high prediction efficiency and potential to deal with imbalanced datasets [34,44]. In addition, parameter tuning seldom affects DT and NB. Furthermore, DT and NB have been used extensively in previous SDP reports. Details on implemented DT and NB classifiers are presented in Table 1.

Table 1. Selected prediction models.

Prediction Algorithms	Parameter Settings
Decision Tree (DT)	ConfidenceFactor = 0.25; MinObj = 2
Naïve Bayes (NB)	NumDecimalPlaces = 2; NumAttrEval = Normal Dist.

3.2. Enhanced Wrapper Feature Selection Method (EWFS) Based on Dynamic Re-Ranking Strategy

The proposed EWFS method is based on incrementally selecting features while considering previously selected features in its search space. First, an entropy measure is used to rank features from a dataset, and then the ranked features are passed through an incremental wrapper method. However, it is only the first B ranked feature selected by the entropy-based on $\log_2 N$ that is passed to the incremental wrapper method. The use of $\log_2 N$ features is per existing empirical studies in SDP [25,27,28]. Thereafter, the remaining features in B are re-ranked using conditional mutual information maximization (CMIM) (see Function 1) while considering the initially selected features. CMIM balances and selects features by maximizing their respective mutual information with the class label while minimizing the co-dependency that may exist between or amongst features using a Multi-Objective Cuckoo search method (see Function 2). In particular, CMIM attempts to balance the quantity of information provided for each possible attribute A_i and class C, as well as the possibility that this information has already been captured by some feature $A_j \in S$. As a result, this technique picks characteristics based on their mutual information with the class while reducing pair-to-pair dependence. For instance, given subset S and a subset of features $\{X_1, \ldots, X_n\}$, the merit $M(X_i, C|X_j)$, i = 1, . . . , n is computed as $M(X_i, C|S) = \min I_{A_j \in S}(X_i, C|X_j)$.

Then, the incremental wrapper method is applied to the newly ranked list after having been initialized by the first selected features from B. This process is repeated until there are no changes in the selected features. This process will reduce the number of wrapper evaluations, as only a few features are considered during each iteration while maintaining or amplifying the prediction performance of the selected features. Algorithm 1 shows the pseudocode for the proposed EWFS method.

Algorithm 1 Pseudocode of proposed Enhanced WFS (EWFS)

Input:
D: Dataset
C: Classifier = $|NB, DT|$ *// to be used for subset evaluation in WFS*
E: Entropy Measure = $|IG|$ *// to be used for subset evaluation in WFS*
T: Threshold = $\log_2{}^{n}$
B: Block Size $A = \{0 \leq B \leq T\}$,
N: Number of features
CS: Multi-Objective Cuckoo Search Method
Output:
$S[\,]$—Subset of Optimal Features

```
for each feature F_i in D { do
    Rank = feature E_D (F_i, Class)
    R[ ] = F_i } // assign ranked features in ascending order into R
S = { }
S.eval = null
B = log_2 N (R) .// select top-ranked features in R as the first block
S = CMIM (D, B, C, S, CS)
continue =: True
while (continue){ do
        R'[ ] = { }
        for each feature F_i in R { do
            Rank = feature E_D (F_i, Class|S)
            R'[ ] = F_i } // assign ranked features in ascending order into R
        R[ ] = R'[ ]
        B = log_2 N (R) .// select top-ranked features in R as the first block
        S' = CMIM(D, B, C, S, CS)
        if (S == S')    P*_t[i] ← P_i[i] // append optimal features from P' based on T
            continue =: False }
        else S = S'
Return S[ ]
```

Function 1 Pseudocode of Conditional Mutual Information Maximization (CMIM) [45]

Input:
N: Number of features
C: Class labels
O: Initial set of features
Output:
$Y[\,]$—Subset of Selected Features

```
for each feature O_i in O { do
        a' ← O_i
            ps[O_i] ← Î_R(C : a')
            m[O_i] ← 0
    for i = 1 to I do
            y* ← 0
        for each feature O_i in O { do
            while (py[O_i] > y* && m[O_i] < i − 1 ) do
                m[O_i] + +
                py[O_i] ← min(py[O_i], Î_RC(C : a' | S[y[O_i]]))
            If (py[O_i] > y*) then
                y* ← py[O_i]
                Y[I] ← a'
Return Y[ ]
```

Function 2 Pseudocode of Multi-Objective Cuckoo Search Method [46]

Initialize objective functions $f_1(x), \ldots, f_k(x); x = (x_1, \ldots x_d)^T$
Generate an initial population of n host nests x_i and each with K eggs
while (t < MaxGeneration) or (stop criterion)
Get a cuckoo (say I) randomly by Levy flights
Evaluate and check if it is Pareto optimal
Choose a nest among n (say j) randomly
Evaluate K solutions for nest j
If new solutions of nest j dominate those of nest i,
Replace nest i with the new solutions set of nest j
End
Abandon a fraction (p_o) of worse nests
Keep the best solution s (or nest with non-dominated sets)
Sort and find the current Pareto optimal solutions
End

3.3. Software Defect Datasets

This research employed defect datasets from four software repositories for experimentation. Specifically, twenty-five defect datasets were culled from PROMISE, NASA, AEEEM, and ReLink repositories. For the NASA repository, the Shepperd, et al. [47] version of defect datasets were selected in this research. The datasets consist of software properties produced by static code analysis. Static code analysis is derived from the source code size and complexity [25,27]. The PROMISE repository contains defect datasets derived from object-oriented metrics and additional information from software modules. This additional information is derived from Apache software [24,27,48]. Concerning the ReLink repository, datasets from this repository are derived from source code information taken from version control. These datasets were created by Wu, et al. [48] as linkage data, and these datasets have been frequently employed in previous SDP experimental studies [49–51]. Lastly, the AEEEM datasets comprise software features derived from software source code analysis that is dependent on software change metrics, entropy, and software code churn [25,27,39,48]. The description of these datasets is presented in Table 2.

3.4. Experimental Procedure

In this section, details on the experimental procedure followed in this research as depicted in Figure 1 are described.

To evaluate the impact and effectiveness of the proposed EWFS on the predictive performance of SDP models, software defect datasets were used to construct SDP models based on NB and DT classification algorithms. Various scenarios were tested and investigated with non-biased and consistent performance comparative analyses of the resulting SDP models.

- Scenario A: In this case, the effectiveness of the proposed EWFS method is tested and correlated with the metaheuristic search-based WFS methods used in this study. The essence of this scenario is to evaluate and validate the performance of the EWFS against BAT, BEE, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Cuckoo search-based WFS methods.
- Scenario B: In this scenario, the EWFS method's performance is evaluated against sequential search-based WFS approaches as proposed in [52,53]. Findings from this scenario are used to validate the effectiveness of the proposed EWFS against Best First Search (BFS), GreeDy Step-Wise Search (GDS), Linear Forward Search (LFS), and Sequential Forward Search (SFS)-based WFS methods.

Experimental results and findings based on scenarios A and B are used to answer the following research questions.

- RQ1. How effective is the EWFS method in comparison to existing metaheuristic search-based WFS methods?

- RQ2. How effective is the EWFS method in comparison to existing sequential search-based WFS methods?

Table 2. Description of software defect datasets.

Datasets	Number of Attributes	Number of Instances
EQ	62	324
JDT	62	997
ML	62	1862
PDE	62	1497
CM1	38	327
KC1	22	1162
KC2	22	522
KC3	40	194
MW1	38	250
PC1	38	679
PC3	38	1077
PC4	38	1287
PC5	39	1711
ANT	22	292
CAMEL	21	339
JEDIT	22	312
REDKITOR	21	176
TOMCAT	22	852
VELOCITY	21	196
XALAN	22	797
SAFE	27	56
ZXING	27	399
APACHE	27	194
ECLIPSE	19	1065
SWT	18	1485

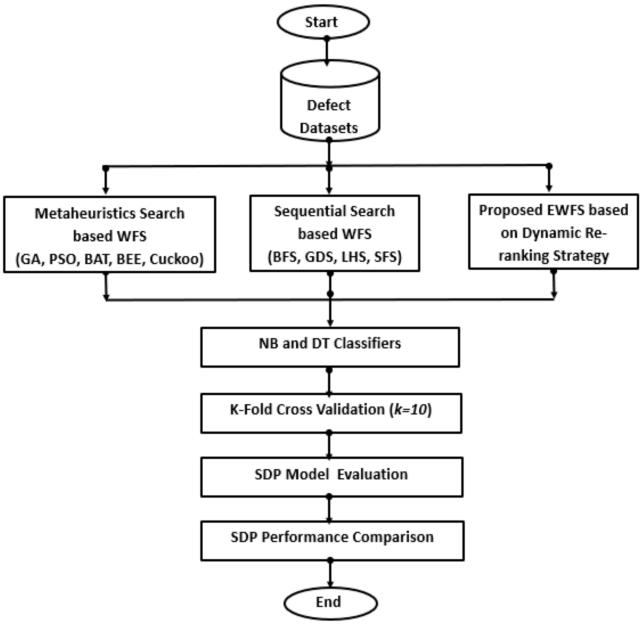

Figure 1. Flowchart for the experimental procedure.

SDP models generated based on the above-listed scenarios are trained and tested using the 10-fold cross-validation (CV) technique. The CV technique guides against data variability issues that may occur in defect datasets. Moreover, the CV technique has been known to produce models with low bias and variance [54–56]. The prediction performances of generated SDP models are assessed using performance evaluation metrics such as

accuracy, AUC, and f-measure. The Scott–Knott ESD statistical rank test is also used to ascertain the significant differences in the prediction performances of the various models studied. The Waikato Environment for Knowledge Analysis (WEKA) machine learning library [57], R programming language [58], and Origin Plot are used for the experimentation on an Intel(R) Core™ machine equipped with i7-6700, running at speed 3.4 GHz CPU with 16 GB RAM.

3.5. Performance Evaluation Assessment

In terms of performance evaluation, SDP models based on the proposed and other methods were analyzed using accuracy, the area under the curve (AUC), and f-measure values, metrics most often used in existing SDP studies to assess the performance of SDP models [9,59].

i. Accuracy is the amount of data accurately estimated out of the actual number of data and can be represented as shown in Equation (1).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \quad (1)$$

ii. F-measure is computed based on the harmonic mean of precision and recall values of observed data. Equation (2) presents the formula for calculating the f-measure value.

$$\text{F} - \text{measure} = 2 \times {}^{\text{Precision}} \times \text{Recall}/_{\text{Precision} + \text{Recall}} \quad (2)$$

iii. The area under curve (AUC) signifies the trade-off between true positives and false positives. It demonstrates an aggregate output assessment across all possible classification thresholds.

Recall $= \left(\frac{\text{TP}}{\text{TP+FN}}\right)$, precision $= \left(\frac{\text{TP}}{\text{TP+FP}}\right)$, TP = true positive (represents accurate prediction); FP = false positive (represents inaccurate prediction); TN = true negative (represents accurate mis-prediction); and FN = false negative (represents inaccurate mis-prediction).

4. Results and Discussion

In this section, findings from experiments concerning the experimental procedure as outlined in Section 3.4 are reported and discussed. Performance metrics such as accuracy, AUC, and f-measure values were used to test the predictive efficiency of NB and DT classifiers based on proposed EWFS, metaheuristic search-based WFS, and sequential search-based WFS. Table 3 presents the experimental results of the proposed EWFS method with NB and DT classifiers on twenty-five defect datasets based on the selected performance metrics. SDP models using EWFS on selected classifiers (NB and DT) were generated utilizing the 10-fold CV method and each experimental process was repeated 10 times. As depicted in Table 3, the proposed EWFS with NB and DT classifier had an average accuracy value of 82.57% and 83.07%, respectively, which shows that models (NB and DT) based on EWFS have a high chance of correctly predicting the average defects in SDP, which translates to a good prediction performance of models based on EWFS. Concerning AUC values, EWFS with NB and DT recorded average AUC values of 0.783 and 0.723, respectively. The high AUC values of models based on EWFS signifies that the prediction process is effective. In addition, the high average AUC values of EWFS on NB (0.768) and DT (0.708) further support its high accuracy value such that the developed models can identify defective and non-defective modules or components. Concerning the average f-measure value, the proposed EWFS had high average f-measure values on NB (0.807) and DT (0.820), which means that the models based on EWFS models have good precision and recall values. That is, the high f-measure of EWFS with NB and DT indicates that the developed models are precise and robust in identifying defective modules or components. Additionally, EWFS selects an average of five features at a relatively low average computational time of 3.318 s. This can be attributed to the dynamism of the re-ranking

strategy which selects features iteratively and reduces the sunset search evaluation method without compromising the performance of the ensuing prediction model.

Table 3. Analysis of experimental performance results of proposed EWFS method.

Datasets	Accuracy Value (%)		AUC Value		F-Measure Value		Selected Features		Time (in Seconds)	
	NB	DT	NB	DT	NB	DT	NB	DT	NB	DT
	EWFS	EWFS	EWFS	EWFS	EWFS	EWFS	EWFS	EWFS	EWFS	EWFS
EQ	74.7	75	0.798	0.782	0.71	0.723	6	2	4.45	1.28
JDT	84.85	84.85	0.848	0.797	0.844	0.819	6	5	5.55	7.73
ML	86.9	86.57	0.743	0.682	0.853	0.835	6	2	4.83	7.39
PDE	86.24	86.64	0.749	0.667	0.832	0.82	7	4	4.94	6.41
CM1	87.16	85.63	0.722	0.509	0.81	0.809	4	7	1.65	7.19
KC1	75.3	75.9	0.683	0.64	0.714	0.713	2	4	3.09	3.99
KC2	83.14	84.48	0.816	0.786	0.821	0.828	5	3	4.14	1.66
KC3	82.47	85.41	0.662	0.601	0.755	0.811	3	3	2.42	1.71
MC2	75.2	72.8	0.674	0.569	0.731	0.655	5	1	6.74	1.3
MW1	90	89.2	0.756	0.536	0.884	0.884	3	3	3.31	1.58
PC1	91.9	91.9	0.826	0.726	0.892	0.889	5	6	4.35	4.75
PC3	84.59	86.54	0.806	0.687	0.934	0.934	3	5	2.77	21.12
PC4	82.67	88.89	0.845	0.878	0.837	0.864	6	3	3.18	2.58
PC5	74.87	76.04	0.711	0.698	0.721	0.726	3	6	3.21	12.51
Safe	76.79	71.43	0.853	0.74	0.752	0.728	3	4	1.7	4.61
Zxing	67.42	68.17	0.592	0.627	0.621	0.622	2	3	3.22	1.76
Apache	74.23	74.74	0.709	0.7	0.742	0.722	4	4	2.09	2.4
Eclipse	100	100	1	1	1	1	1	1	1.86	1.73
SWT	83.64	89.9	0.887	0.915	0.832	0.894	3	6	2.05	8.03
ANT	90.07	88.36	0.768	0.685	0.953	0.863	4	4	2.5	4.79
JEDIT	82.69	80.45	0.793	0.707	0.85	0.792	4	3	2.86	2.44
REDKITOR	90.34	89.77	0.733	0.709	0.962	0.885	5	3	3.24	1.3
TOMCAT	90.96	92.02	0.824	0.706	0.953	0.896	6	4	2.69	4.09
VELOCITY	88.27	85.71	0.796	0.675	0.871	0.823	4	2	2.67	1.67
XALAN	59.97	66.25	0.611	0.688	0.65	0.675	3	4	3.44	6.1
Average	82.57	83.07	0.768	0.708	0.821	0.826	4.120	3.680	3.318	4.805

The experimental results analysis shows the applicability and performance of the proposed EWFS in selecting optimal features in a relatively low computational time without limiting its performance. The following sub-sections focus on the performance comparison of the proposed EWFS against existing metaheuristic search-based WFS and sequential search-based WFS. The essence of the comparison is to compare the performance of the proposed EWFS against existing WFS methods with different subset search approaches.

4.1. Comparison of Proposed EWFS Method against Metaheuristic Search-Based WFS Methods

In this sub-section, the performance of the proposed EWFS is compared with metaheuristic search-based WFS. Specifically, the prediction performance of NB and DT models based on the proposed EWFS and WFS based on Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Bat Search (BAT), Bee Search (BEE), and Cuckoo Search (Cuckoo) are compared and contrasted.

Figure 2 presents box-plot representations based on average accuracy values of NB and DT models with EWFS and metaheuristic search-based WFS methods. In terms of average accuracy values, the proposed EWFS with NB and DT classifiers had superior average accuracy values when compared with NB and DT models based on the metaheuristic search-based WFS method. EWFS with NB and DT recorded average accuracy values of 82.57% and 83.07%, respectively, compared with the metaheuristic search-based WFS method based on GA (NB: 81.57%, DT: 81.58%), PSO (NB: 81.7%, DT: 81.55%), BAT (NB: 81.4%, DT: 81.64%), BEE (NB: 81.79%, DT: 81.9%), and Cuckoo (NB: 81.66%, DT: 81.84%). Specifically, NB and DT models based on EWFS outperformed models based on GA by

+1.23% and +1.83%, PSO by +1.06% and +1.49%, BAT by +1.44% and +1.75%, BEE by +0.95% and +1.43%, and Cuckoo by +1.11% and +1.5%, respectively, based on average accuracy values. Thus, these analyses indicate the performance and superiority of EWFS over metaheuristic search-based WFS (GA, PSO, BAT, BEE, and Cuckoo) methods based on average accuracy values.

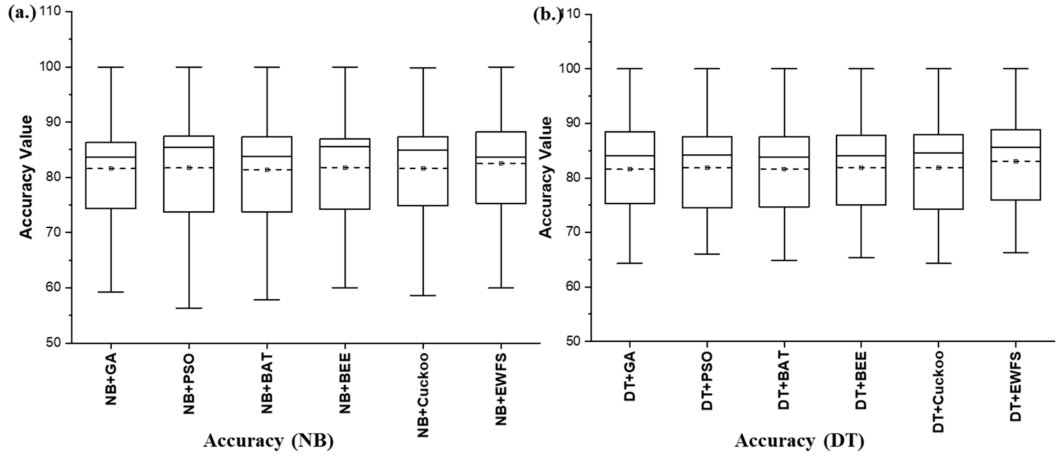

Figure 2. Box-plot representations of the performance (accuracy) of EWFS and metaheuristic search-based WFS methods on NB and DT classifiers. (**a**) Average accuracy values of NB (**b**) Average accuracy values of DT.

Figure 3 presents the box-plot representations based on average AUC values of NB and DT models with EWFS and metaheuristic search-based WFS methods. NB and DT classifiers with the EWFS method recorded higher average AUC values when compared against models based on metaheuristic search-based WFS methods. EWFS with NB and DT recorded average AUC values of 0.768 and 0.708, respectively, compared with the metaheuristic search-based WFS method based on GA (NB: 0.739, DT: 0.683), PSO (NB: 0.722, DT: 0.683), BAT (NB: 0.722, DT: 0.68), BEE (NB: 0.728, DT: 0.683), and Cuckoo (NB: 0.725, DT: 0.686). In particular, NB and DT models based on EWFS outperformed models based on GA by +3.92% and +3.66%, PSO by +6.37% and +3.66%, BAT by +6.37% and +4.11%, BEE by +5.49% and +3.66%, and Cuckoo by +5.93% and +3.21%, respectively, based on average AUC values.

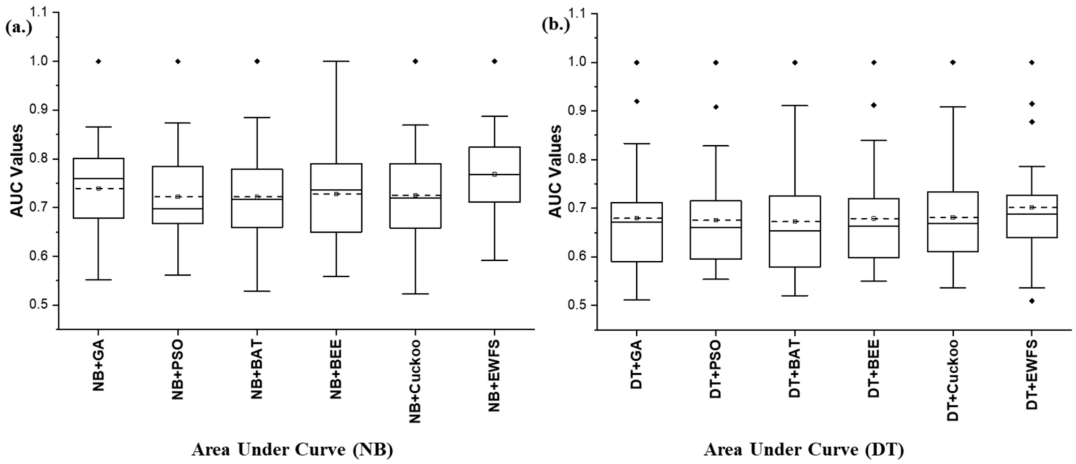

Figure 3. Box-plot representations of the performance (AUC) of EWFS and metaheuristic search-based WFS methods on NB and DT classifiers. (**a**) Average AUC values of NB (**b**) Average AUC values of DT.

Additionally, Figure 4 shows box-plot representations based on average f-measure values of NB and DT models with EWFS and metaheuristic search-based WFS methods. Prediction models (NB and DT) with EWFS methods recorded average f-measure values of 0.821 and 0.826, respectively, which are superior to the average f-measure values of models

based on metaheuristic search-based WFS methods such as GA (NB: 0.791, DT: 0.796), PSO (NB: 0.794, DT: 0.799), BAT (NB: 0.797, DT: 0.796), BEE (NB: 0.793, DT: 0.8), and Cuckoo (NB: 0.793, DT: 0.798).

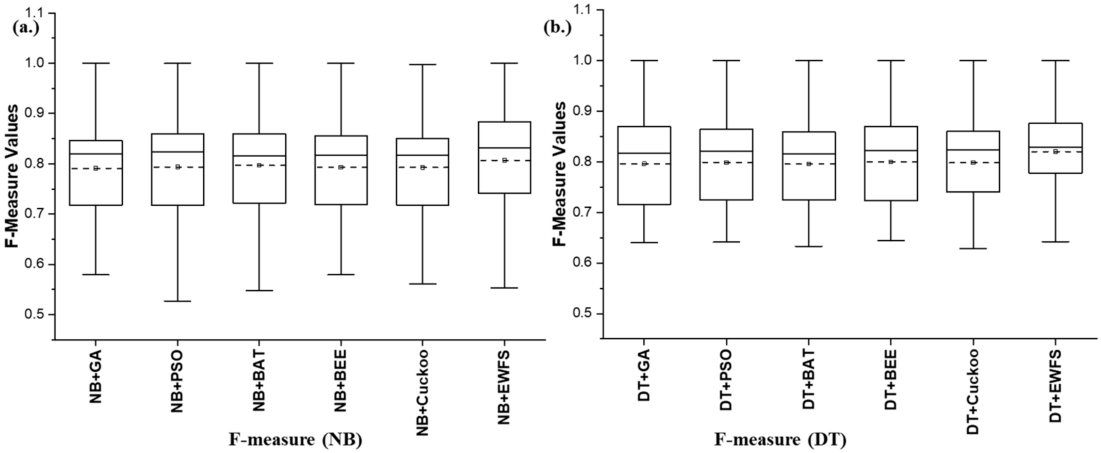

Figure 4. Box-plot representations of the performance (f-measure) of EWFS and metaheuristic search-based WFS methods on NB and DT classifiers. (**a**) Average F-measure values of NB (**b**) Average F-measure values of DT.

Specifically, NB and DT models based on EWFS outperformed models based on GA by +3.79% and +3.77%, PSO by +3.4% and +3.38%, BAT by +3.01% and +3.77%, BEE by +3.53% and +3.25%, and Cuckoo by +3.53% and +3.51%, based on average f-measure values. Hence, NB and DT models with EWFS recorded superior f-measure values when compared against models with metaheuristic search-based WFS methods. Table 4 presents and compares the number of selected features by the proposed EWFS and metaheuristic search-based WFS methods. It can be observed that the proposed method selected, on average, fewer features compared with the average number of features selected by the metaheuristics search-based WFS methods. Additionally, as shown in Table 5, the average computational time (in seconds) taken by the proposed EWFS in selecting features was less than the computational time of other methods. Specifically, EWFS took an average computational time of 3.32 and 4.8 s for selecting on average five and four features (approximately) for NB and DT models respectively. As compared with GA (NB: (7.2 features, 3.65 s), DT (9.2 features, 22.15 s)), PSO (NB: (6.44 features, 2.85 s), DT: (5.84 features, 18.55 s)), BAT (NB: (6.44 features, 5.84 s), DT: (7.36 features, 29.74 s)), BEE (NB: (5.44 features, 6.37 s), DT: (5.64 features, 31.92 s)), and Cuckoo (NB: (6.2 features, 2.76 s), DT: (7.44 features, 12.1 s)). This analysis indicates that the proposed EWFS method can select better and important features in a reasonable computational time. The relatively low computational time for EFWS may be attributed to the re-ranking strategy deployed at the subset evaluation stage for dynamically and iteratively selecting relevant features, thereby reducing the wrapper evaluation cycle and subsequently the computational time.

Based on the aforementioned experimental results and analyses, it is evident that the proposed EWFS method outperformed existing metaheuristic search-based WFS methods. For further analyses, the performance of EWFS and the existing metaheuristic search-based WFS methods were subjected to the Scott–Knott ESD statistical rank test to determine the statistically significant differences in their respective performances.

Figure 5 presents the Scott–Knott ESD statistical rank test results of the proposed EWFS method and the metaheuristics search-based WFS methods we evaluated on NB and DT, based on average accuracy values. From Figure 5, it can be observed that average accuracy performances of NB and DT based on the EWFS method, although superior, are not statistically superior to models based on metaheuristics search-based WFS methods. That is, although models based on EWFS had superior average accuracy values, there is not much difference between the average accuracy values of the proposed method when compared to existing metaheuristic search-based WFS methods. The average accuracy

value may not be sufficient for the evaluation. Consequently, other metrics such as AUC and f-measure can be used for performance evaluations.

Table 4. Comparison of average selected features by the EWFS and metaheuristic-search-based WFS methods.

Dataset	NB+GA	NB+PSO	NB+BAT	NB+BEE	NB+Cuckoo	NB+EWFS	DT+GA	DT+PSO	DT+BAT	DT+BEE	DT+Cuckoo	DT+EWFS
EQ	22	15	26	16	23	6	23	9	7	14	21	2
JDT	20	18	16	8	15	6	10	5	7	8	11	5
ML	8	12	5	3	3	6	9	6	9	4	10	2
PDE	16	8	7	6	8	7	11	2	12	9	11	4
CM1	3	3	3	1	1	4	5	9	6	2	3	7
KC1	3	4	3	3	2	2	5	4	5	2	2	4
KC2	4	3	2	2	3	5	4	4	5	5	10	3
KC3	4	3	6	3	3	3	15	9	17	3	10	3
MC2	15	8	16	5	16	5	5	1	3	4	2	1
MW1	5	5	4	4	5	3	14	12	12	7	11	3
PC1	6	1	4	3	1	5	17	6	8	6	5	6
PC3	4	3	3	5	1	3	2	1	4	4	6	5
PC4	10	6	7	7	7	6	12	6	5	6	14	3
PC5	10	19	9	18	21	3	10	14	10	6	7	6
Safe	5	4	4	2	2	3	9	5	4	5	6	4
Zxing	5	6	10	9	2	2	14	13	14	14	12	3
Apache	11	11	8	9	9	4	7	4	6	4	6	4
Eclipse	1	1	1	1	2	1	6	2	4	1	4	1
SWT	2	2	2	2	2	3	8	10	10	11	9	6
ANT	3	3	2	5	4	4	8	6	10	6	6	4
JEDIT	7	7	7	5	6	4	11	4	6	6	6	3
REDKITOR	5	1	4	2	2	5	4	2	4	2	2	3
TOMCAT	1	3	1	1	1	6	7	1	1	1	1	4
VELOCITY	6	6	6	7	7	4	2	2	2	2	2	2
XALAN	4	9	10	9	9	3	12	9	13	9	9	4
Average	7.20	6.44	6.64	5.44	6.20	4.12	9.20	5.84	7.36	5.64	7.44	3.68

Table 5. Comparison of average computational time (in seconds) by the EWFS and metaheuristic-search-based WFS methods.

Dataset	NB+GA	NB+PSO	NB+BAT	NB+BEE	NB+Cuckoo	NB+EWFS	DT+GA	DT+PSO	DT+BAT	DT+BEE	DT+Cuckoo	DT+EWFS
EQ	6.67	5.42	9.99	11.37	5.3	4.45	32.6	17.96	19.23	48.7	17.61	1.28
JDT	12.1	8.44	24.95	23.48	12.91	5.55	46.62	23.88	37.06	80.56	19.2	7.73
ML	8.9	5.41	12.78	15.59	5.38	4.83	72.02	81.06	91.86	100.73	43.11	7.39
PDE	6.49	6.17	10.46	15.6	5	4.94	81.58	42.83	87.2	124.7	46.77	6.41
CM1	2.22	1.41	3.5	2.52	1.37	1.65	5.73	6.14	8	8.21	3	7.19
KC1	2.16	1.96	3.12	2.72	1.16	3.09	12.76	12.01	20.05	13.44	5.62	3.99
KC2	1.61	1.29	2.46	2.04	1.21	4.14	4.54	7.83	16.73	17.1	4.77	1.66
KC3	1.62	1.26	2.58	2.63	1.62	2.42	7.92	5.28	11.18	12.06	5.09	1.71
MC2	1.77	1.48	2.91	2.58	2.09	6.74	2.82	2.02	3.7	4.22	1.83	1.3
MW1	1.69	1.47	2.36	2.97	1.84	3.31	9.9	10.9	12.43	14.53	6.81	1.58
PC1	3.92	2.31	5.34	5.43	1.86	4.35	18.28	9.81	17.09	26.26	7.43	4.75
PC3	6.45	3.45	6.53	10.16	2.6	2.77	16.92	12.7	17.31	36.03	9.29	21.12
PC4	10.7	7.51	16.27	14.61	5.29	3.18	51.15	30.99	62.89	64.38	23.36	2.58
PC5	7.45	9.12	11.25	22.16	5.51	3.21	80.27	110.31	112.94	93.08	31.67	12.51
Safe	0.97	0.56	0.96	0.83	0.72	1.7	1.4	1.33	2.13	2.49	1.08	4.61
Zxing	1.81	2.14	4.33	3.5	1	3.22	11.94	12.05	19.09	21.01	6.47	1.76
Apache	1.56	2.31	2.46	2.3	1.1	2.09	3.63	2.78	4.83	4.68	2.54	2.4
Eclipse	3.15	1.58	3.98	3.53	1.96	1.86	1.76	5.25	6.08	4.31	2.83	1.73
SWT	2.08	1.01	4.63	2.21	1.51	2.05	31.77	22.8	77.04	52.72	14.59	8.03
ANT	1.1	0.74	1.8	2.06	0.95	2.5	4.9	5.06	9.69	7.74	5.09	4.79
JEDIT	1.66	1.43	2.57	1.98	1.3	2.86	4.86	6.01	7.87	7.87	7.84	2.44
REDKITOR	0.84	0.8	1.81	1.26	1.19	3.24	1.75	1.77	9	2.56	8.5	1.3
TOMCAT	1.71	1.12	2.03	2.11	1.53	2.69	10.45	5.26	7.53	6.64	5.2	4.09
VELOCITY	1.14	0.83	2.51	1.76	1.9	2.67	2.71	2.2	4.98	3.15	2.4	1.67
XALAN	1.57	1.95	4.49	3.73	2.71	3.44	35.56	25.45	77.52	40.76	20.5	6.1
Average	3.65	2.85	5.84	6.37	2.76	3.32	22.15	18.55	29.74	31.92	12.10	4.80

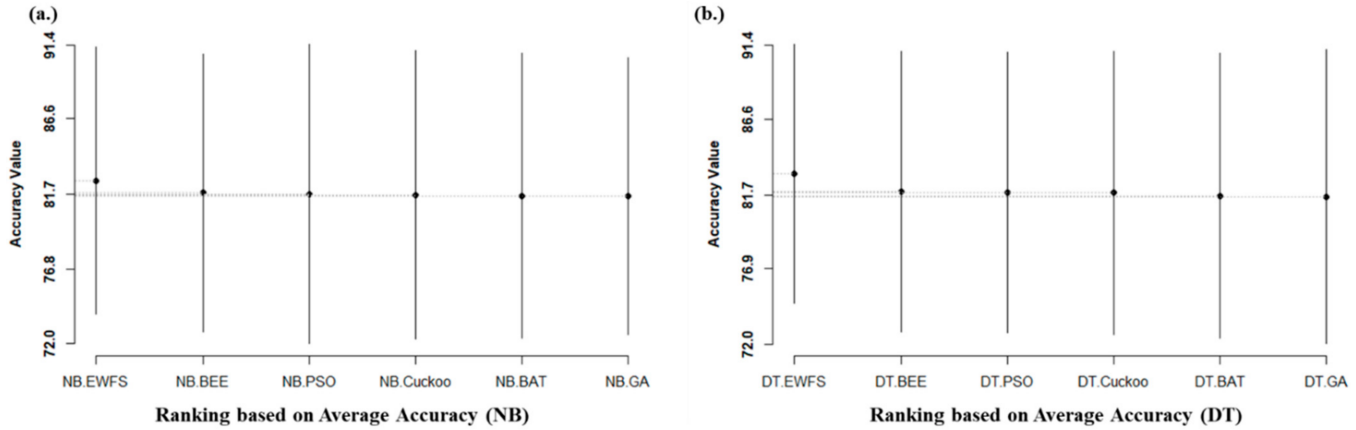

Figure 5. Scott–Knott ESD statistical rank test representations of the performance (accuracy) of EWFS and metaheuristic search-based WFS methods on NB and DT classifiers. (**a**) Average accuracy values of NB (**b**) Average accuracy values of DT.

Figure 6 shows the Scott–Knott ESD statistical rank test results of the proposed EWFS method and the metaheuristics search-based WFS methods we studied on NB and DT based on average AUC values. Significant statistical differences in the average AUC values of NB and DT models based on EWFS methods when compared with metaheuristic search-based WFS were observed. Specifically, NB and DT models based on EWFS outrank and outperform the metaheuristic search-based WFS methods examined. This analysis showed that the high accuracy and low AUC values of models based on metaheuristic search-based WFS might be a result of overfitting, which is one of the inherent problems in metaheuristic search-based WFS, as they can easily get trapped in the local minima. Additionally, similar findings are observed in the case average f-measure value. As indicated in Figure 7, NB and DT models based on the EWFS method outranked and outperformed metaheuristic search-based WFS, as evidenced by a significant statistical difference in the average f-measure values in favor of NB and DT models based on EWFS. Table 6 presents a summary of the Scott–Knott ESD statistical rank test results of NB and DT models based on EWFS and metaheuristic-search-based WFS (GA, PSO, BEE, BAT, and Cuckoo) methods.

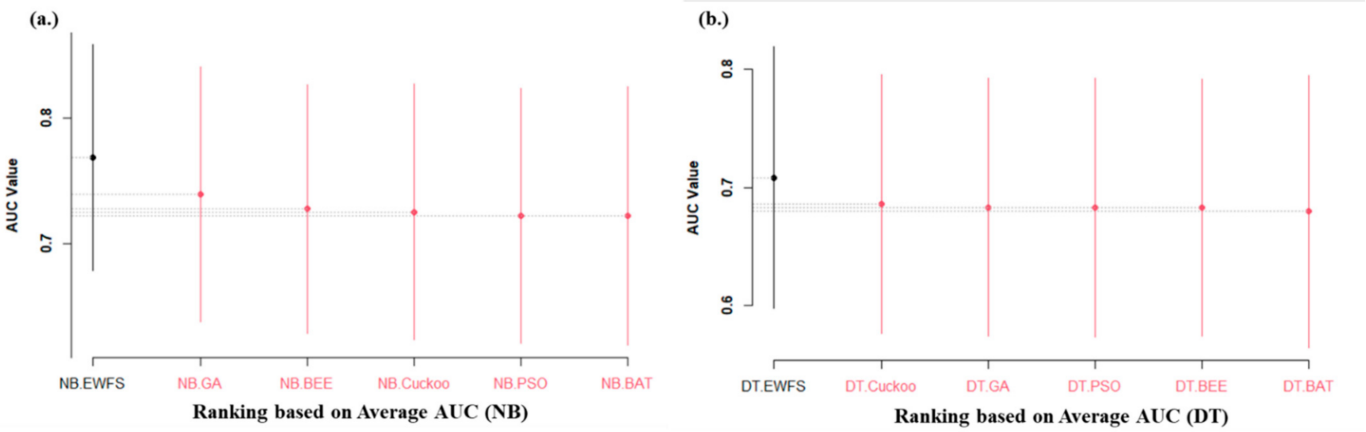

Figure 6. Scott–Knott ESD statistical rank test representations of the performance (AUC) of EWFS and metaheuristic search-based WFS methods on NB and DT classifiers. (**a**) Average AUC values of NB (**b**) Average AUC values of DT.

From Table 6, it can be deduced that NB and DT models based on EWFS are superior and rank best when compared against models based on metaheuristic search-based WFS methods. This indicates that there are significant statistical differences in the average performance of models based on EWFS when compared with existing metaheuristic search-based WFS methods, in favor of EWFS methods, based on average accuracy, average AUC, and average f-measure values. This further confirms and supports the superiority of EFWS over existing metaheuristic search-based WFS methods such as GA, PSO, BEE, BAT, and

Cuckoo searches for the feature selection process in SDP. However, for generalizability, the performance of EWFS is compared against other forms of WFS which are the sequential search-based WFS in the succeeding subsection.

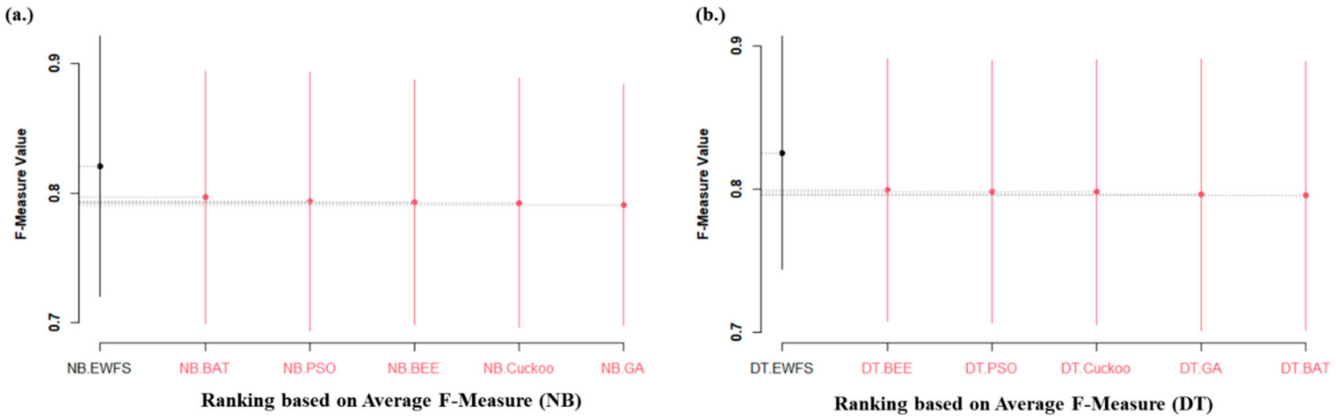

Figure 7. Scott–Knott ESD statistical rank test representations of the performance (f-measure) of EWFS and metaheuristic search-based WFS methods on NB and DT classifiers. (**a**) Average F-measure values of NB (**b**) Average F-measure values of DT.

Table 6. Analysis of Scott–Knott ESD statistical rank test results of EWFS and metaheuristic search-based WFS methods.

Statistical Ranking Based on Average Accuracy				Statistical Ranking Based on Average AUC				Statistical Ranking Based on Average F-Measure			
NB		DT		NB		DT		NB		DT	
Rank	**FS Methods**	**Rank**	**FS Methods**	**Rank**	**FS Methods**	**Rank**	**FS Methods**	**Rank**	**FS Methods**	**Rank**	**FS Methods**
1	EWFS, BEE, PSO, Cuckoo, BAT, GA	1	EWFS, BEE, PSO, Cuckoo, BAT, GA	1	EWFS	1	EWFS	1	EWFS, BAT, PSO, BEE, Cuckoo, GA	1	EWFS
				2	GA, BEE, Cuckoo, PSO, BAT	2	Cuckoo, GA, POS, BEE, BAT			2	BEE, PSO, Cuckoo, GA, BAT

4.2. Comparison of Proposed EWFS Method against Sequential Search-Based WFS Methods

In this sub-section, the performance of the proposed EWFS is compared with sequential search-based WFS methods. Specifically, the prediction performance of NB and DT models based on the proposed EWFS and WFS are compared using Best First Search (BFS), Greedy Step-wise Search (GDS), Linear Forward Search (LFS), and Sequential Forward Search (SFS).

Figure 8 shows the box-plot representation of average accuracy values of NB and DT models with EWFS and sequential search-based WFS methods. Like the metaheuristic search-based WFS, concerning average accuracy values, the proposed EWFS with NB and DT classifier had superior average accuracy values when compared with NB and DT models based on the sequential search-based WFS method. EWFS with NB and DT recorded an average accuracy value of 82.57% and 83.07%, respectively, compared with metaheuristic search-based WFS methods based on BFS (NB: 81.9%, DT: 81.81%), GDS (NB: 81.93%, DT: 81.88%), LFS (NB: 81.91%, DT: 81.69%), and SFS (NB: 81.88%, DT: 81.8). Specifically, NB and DT models based on EWFS outperformed models based on BFS by +0.8% and +1.54%, GDS by +0.78% and +1.45%, LFS by +0.8% and +1.69%, and SFS by +0.84% and +1.55%, respectively, based on average accuracy values. Based on these analyses, the performance and superiority of EWFS over sequential search-based WFS (BFS, GDS, LFS, and SFS) methods based on average accuracy values can be observed.

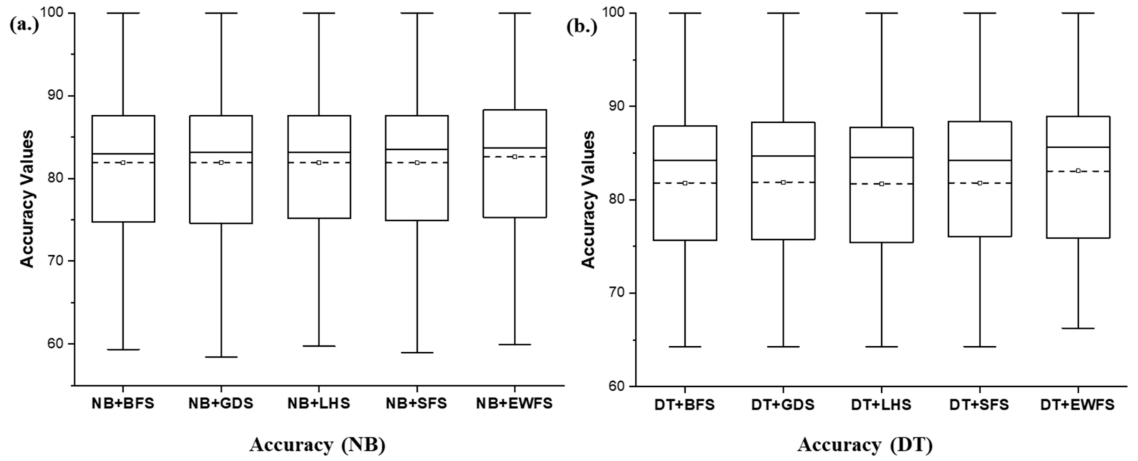

Figure 8. Box-plot representations of the performance (accuracy) of EWFS and sequential search-based WFS methods on NB and DT classifiers. (**a**) Average accuracy values of NB (**b**) Average accuracy values of DT.

Additionally, Figure 9 showcases box-plot representations based on average AUC values of NB and DT models with EWFS and sequential search-based WFS methods. NB and DT classifiers with the EWFS method recorded higher average AUC values when compared against models based on sequential search-based WFS methods. EWFS with NB and DT recorded an average AUC value of 0.768 and 0.708, respectively, compared with metaheuristic search-based WFS methods based on BFS (NB: 0.732, DT: 0.679), GDS (NB: 0.725, DT: 0.668), LFS (NB: 0.731, DT: 0.683), and SFS (NB: 0.733, DT: 0.676). That is, NB and DT models based on EWFS outperformed models based on BFS by +4.92% and +4.66%, GDS by +5.93% and +5.98%, LFS by +5.06% and +3.66%, and SFS by +4.77% and +4.73%, respectively, based on average AUC values.

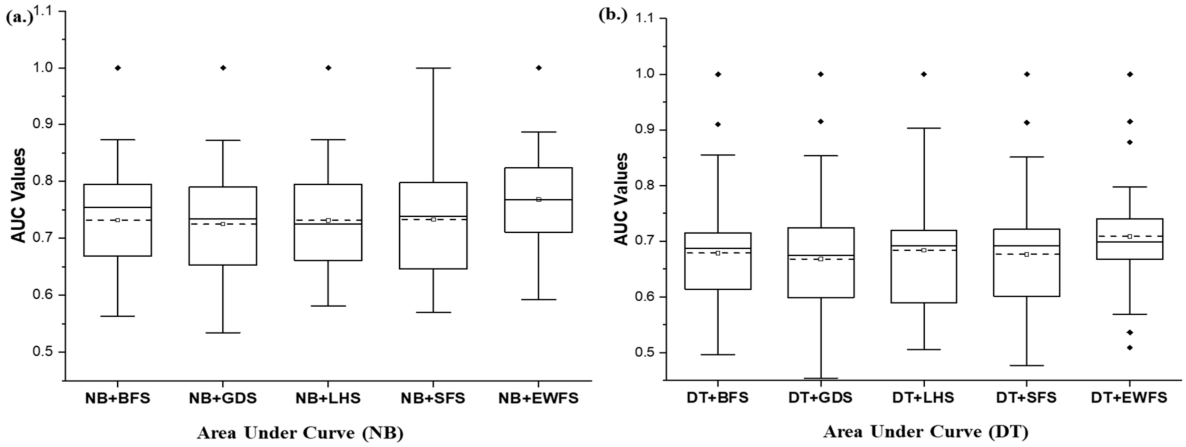

Figure 9. Box-plot representations of the performance (AUC) of EWFS and sequential search-based WFS methods on NB and DT classifiers. (**a**) Average AUC values of NB (**b**) Average AUC values of DT.

Furthermore, Figure 10 presents the box-plot representation based on average f-measure values of NB and DT models with EWFS and sequential search-based WFS methods. Prediction models (NB and DT) with EWFS methods recorded average f-measure values of 0.821 and 0.826, respectively, which are superior to average f-measure values of models based on sequential search-based WFS methods such as BFS (NB: 0.804, DT: 0.808), GDS (NB: 0.801, DT: 0.81), LFS (NB: 0.801, DT: 0.8), and SFS (NB: 0.8, DT: 0.806). Specifically, NB and DT models based on EWFS outperformed models based on LFS by +2.11% and +2.23%, GDS by +2.5% and +1.97%, LFS by +2.5% and +3.25%, and SFS by +2.63% and +2.48%, respectively, based on average f-measure values. Consequently, NB and DT models with EWFS recorded superior f-measure values when compared against models with sequential search-based WFS methods.

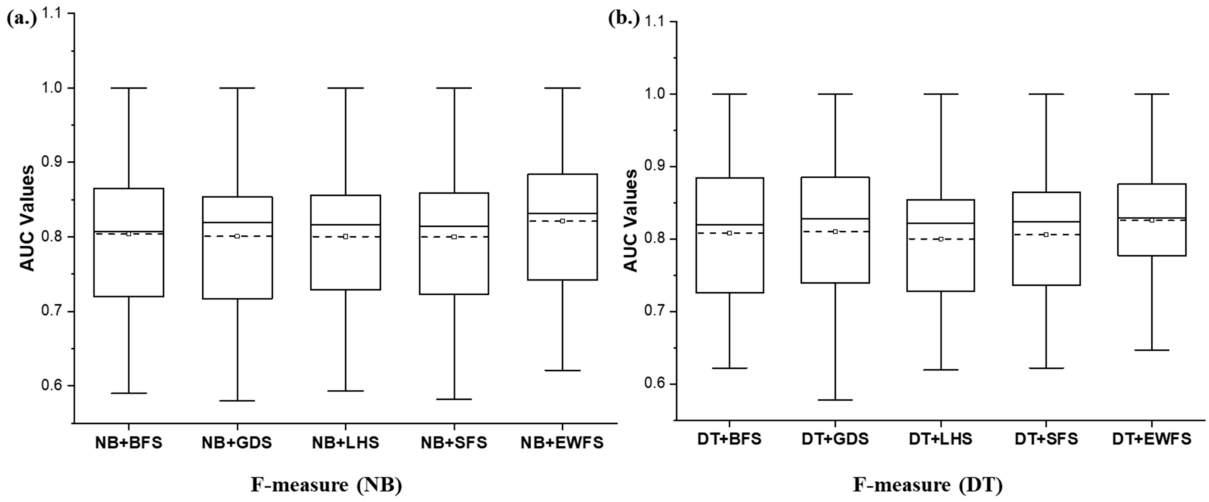

Figure 10. Box-plot representations of the performance (f-measure) of EWFS and sequential search-based WFS methods on NB and DT classifiers. (**a**) Average F-measure values of NB (**b**) Average F-measure values of DT.

In terms of the number of selected features, Table 7 presents and compares the number of selected features by the proposed EWFS and sequential search-based WFS methods. It can be observed that the proposed EWFS selected, on average, a relatively fewer number of features compared with the average number of features selected by the sequential search-based WFS methods studied. In some cases, the sequential search-based WFS (GDS and SFS) selected fewer features than the proposed EWFS. However, EWFS still outperformed these sequential search-based WFS methods (GDS and SFS). Perhaps, the features selected by these methods have low prediction performance, unlike the EWFS, which dynamically selects features based on their relevance while maintaining or enhancing its performance.

Table 7. Comparison of average selected features by the EWFS and sequential search-based WFS methods.

Dataset	NB+GA	NB+PSO	NB+BAT	NB+BEE	NB+Cuckoo	NB+EWFS	DT+GA	DT+PSO	DT+BAT	DT+BEE	DT+Cuckoo	DT+EWFS
EQ	10	8	10	5	6	8	7	8	4	2	10	8
JDT	15	4	8	5	6	12	3	7	1	5	15	4
ML	2	2	2	1	6	11	4	12	1	2	2	2
PDE	4	4	4	5	7	3	3	3	6	4	4	4
CM1	5	1	4	1	4	1	1	1	1	7	5	1
KC1	4	2	4	3	2	3	3	3	2	4	4	2
KC2	3	3	3	1	5	4	2	4	2	3	3	3
KC3	3	3	3	2	3	2	2	2	1	3	3	3
MC2	8	3	8	3	5	1	1	4	1	1	8	3
MW1	5	1	3	3	3	7	1	5	2	3	5	1
PC1	7	1	1	1	5	8	5	5	1	6	7	1
PC3	1	1	1	1	3	5	1	1	1	5	1	1
PC4	6	6	6	6	6	5	4	5	2	3	6	6
PC5	6	3	6	4	3	9	4	9	2	6	6	3
Safe	3	3	3	1	3	7	1	7	1	4	3	3
Zxing	2	2	2	1	2	2	1	9	10	3	2	2
Apache	11	2	11	4	4	4	2	2	1	4	11	2
Eclipse	1	1	1	1	1	1	1	1	1	1	1	1
SWT	2	2	2	2	3	12	8	12	9	6	2	2
ANT	2	2	2	1	4	1	1	1	1	4	2	2
JEDIT	3	3	3	2	4	5	5	5	2	3	3	3
REDKITOR	1	1	1	1	5	1	1	1	4	3	1	1
TOMCAT	1	1	1	1	6	1	1	1	1	4	1	1
VELOCITY	3	3	3	2	4	4	4	3	3	2	3	3
XALAN	8	6	8	4	3	7	7	7	14	4	8	6
Average	4.64	2.72	4.00	2.44	4.12	4.96	2.92	4.72	2.96	3.68	4.64	2.72

In addition, as presented in Table 8, the average computational time (in seconds) taken by the proposed EWFS in selecting features is relatively less than the computational time of the sequential search-based WFS methods studied. On NB models, EWFS had a relatively low computational time, although other sequential search-based WFS methods recorded

a shorter time. However, in the case of DT models, EWFS had low computational time, which was better than that of the sequential search-based WFS methods, which had high computational times. The high computational time of the sequential search-based WFS methods we tested can be attributed to their respective stagnation and trapping in local optima during the subset evaluation phase. Specifically, EWFS took an average computational time of 3.32 and 4.8 s for selecting an average of five and four features (approximately) for NB and DT models, respectively, as compared with BFS (NB: (4.64 features, 3.24 s), DT: (4.96 features, 21.13 s)), GDS (NB: (2.72 features, 0.7 s), DT: (2.92 features, 2.8 s)), LFS (NB: (4 features, 2.17 s), DT: (4.72 features, 13.76 s)), and SFS (NB: (2.44 features, 1.39 s), DT: (2.96 features, 9.13 s)). The preceding findings demonstrate that the proposed EWFS can select better and more important features in a reasonable computational time. The relatively low computational time for EFWS may be attributed to the re-ranking strategy deployed at the subset evaluation stage for dynamically and iteratively selecting relevant features, thereby reducing the wrapper evaluation cycle, and subsequently computational time.

Table 8. Comparison of average computational time (in seconds) by the EWFS and sequential search-based WFS methods.

Dataset	NB+GA	NB+PSO	NB+BAT	NB+BEE	NB+Cuckoo	NB+EWFS	DT+GA	DT+PSO	DT+BAT	DT+BEE	DT+Cuckoo	DT+EWFS
EQ	6.96	2.81	5.43	2.63	4.45	46.97	6.27	11.86	5.9	1.28	6.96	2.81
JDT	29.07	2.07	8.74	4.46	5.55	133.64	4.06	32.47	14.76	7.73	29.07	2.07
ML	5.38	1.63	4.76	3.05	4.83	141.98	11.51	115.55	41.16	7.39	5.38	1.63
PDE	6.77	3.07	5.96	6.5	4.94	14.55	4.92	11.85	46.09	6.41	6.77	3.07
CM1	2.26	0.1	2.19	0.68	1.65	1.01	0.1	0.69	1.46	7.19	2.26	0.1
KC1	1.63	0.32	1.19	1.27	3.09	6.33	2	6.35	6.12	3.99	1.63	0.32
KC2	0.84	0.34	0.93	0.31	4.14	2.86	0.55	3	1.82	1.66	0.84	0.34
KC3	1.02	0.43	1.11	0.8	2.42	1.36	0.43	1.39	1.41	1.71	1.02	0.43
MC2	1.54	0.3	1.54	0.67	6.74	0.91	0.25	2	1.19	1.3	1.54	0.3
MW1	1.44	0.08	1.16	0.85	3.31	5.44	0.1	2.6	1	1.58	1.44	0.08
PC1	5.19	0.32	1.47	0.68	4.35	15.65	3.41	8.49	2.51	4.75	5.19	0.32
PC3	0.95	0.16	1.16	0.47	2.77	13.92	0.18	1.67	0.57	21.12	0.95	0.16
PC4	4.94	2.61	5.08	4.29	3.18	34.44	6.3	29.35	12.57	2.58	4.94	2.61
PC5	4.85	1.02	5.05	3.33	3.21	63.38	9.98	62.79	45.29	12.51	4.85	1.02
Safe	0.33	0.14	0.35	0.24	1.7	1.36	0.14	1.53	0.52	4.61	0.33	0.14
Zxing	0.51	0.22	0.58	0.42	3.22	1.91	0.1	9.57	4.56	1.76	0.51	0.22
Apache	1.51	0.18	1.55	0.72	2.09	1.9	0.37	1.1	0.85	2.4	1.51	0.18
Eclipse	0.92	0.21	0.95	0.3	1.86	0.53	0.21	0.67	0.47	1.73	0.92	0.21
SWT	0.96	0.35	1.04	0.73	2.05	22.92	10.91	23.46	15.65	8.03	0.96	0.35
ANT	0.52	0.2	0.56	0.32	2.5	0.89	0.21	0.93	0.53	4.79	0.52	0.2
JEDIT	0.58	0.26	0.63	0.43	2.86	2.23	1.31	2.32	1.56	2.44	0.58	0.26
REDKITOR	0.29	0.07	0.32	0.18	3.24	0.26	0.09	0.29	0.55	1.3	0.29	0.07
TOMCAT	0.45	0.08	0.53	0.23	2.69	1.14	0.26	1.26	1.61	4.09	0.45	0.08
VELOCITY	0.46	0.18	0.44	0.38	2.67	1.29	0.6	1.09	0.9	1.67	0.46	0.18
XALAN	1.52	0.72	1.57	0.87	3.44	11.39	5.86	11.65	19.08	6.1	1.52	0.72
Average	3.24	0.71	2.17	1.39	3.32	21.13	2.80	13.76	9.13	4.80	3.24	0.71

Based on the aforementioned experimental results and findings, it is evident that the proposed EWFS method is superior to existing sequential search-based WFS methods. Nonetheless, further statistical analysis was conducted on the performance of EWFS and the experimented sequential search-based WFS methods using the Scott–Knott ESD statistical rank test to determine the statistically significant differences in their respective performances based on average accuracy, average AUC, and average f-measure values.

Figure 11 shows the Scott–Knott ESD statistical rank test results of the proposed EWFS method and sequential search-based WFS methods on NB and DT based on average accuracy values. As shown in Figure 11, like the metaheuristic search-based WFS case, it can be observed that average accuracy performances of NB and DT based on the EWFS method, although superior, showed no statistically significant differences compared to NB and DT models based on sequential search-based WFS methods. That is, although models based on EWFS had superior average accuracy values, there is not much difference between the average accuracy values of EWFS and existing sequential search-based WFS methods.

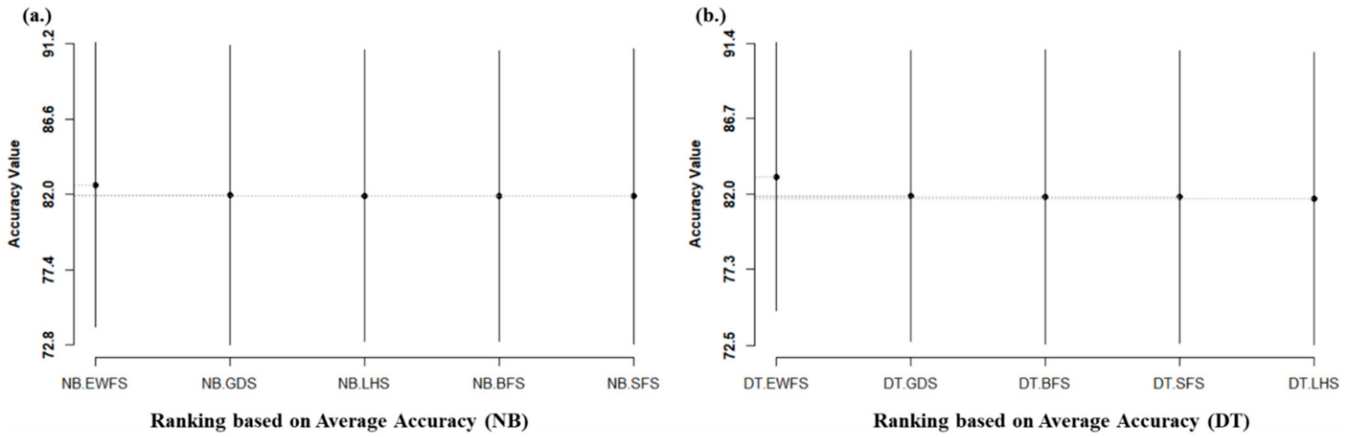

Figure 11. Scott–Knott ESD statistical rank test representations of the performance (accuracy) of EWFS and sequential search-based WFS methods on NB and DT classifiers. (**a**) Average accuracy values of NB (**b**) Average accuracy values of DT.

Figure 12 illustrates the Scott–Knott ESD statistical rank test results of the proposed EWFS method and sequential search-based WFS methods based on average AUC values. There existed significant statistical differences in the average AUC values of NB and DT models based on EWFS methods when compared to sequential search-based WFS. Specifically, NB and DT models based on EWFS outranked and outperformed the sequential search-based WFS methods we tested. These findings suggest that the high accuracy and low AUC values of models based on sequential search-based WFS might be due to overfitting, which is one of the prominent problems in sequential search-based WFS.

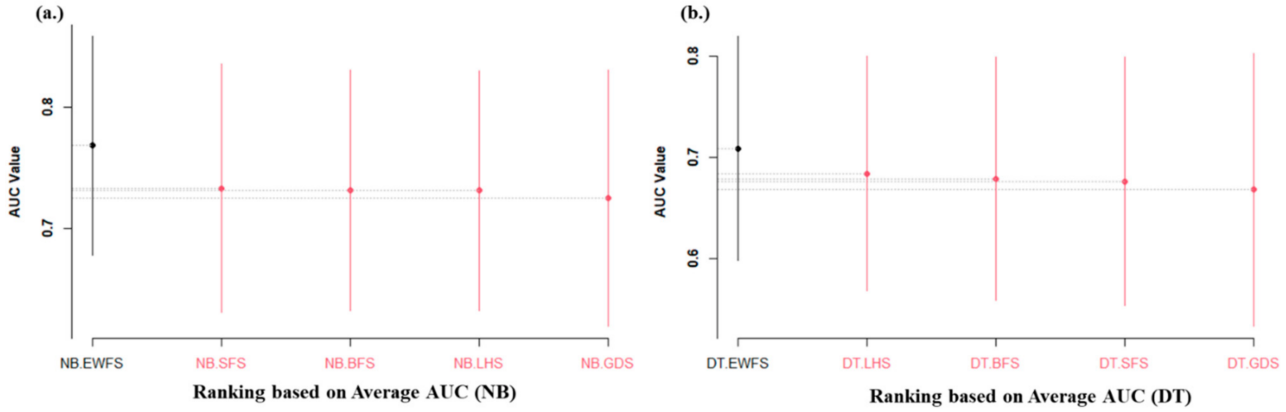

Figure 12. Scott–Knott ESD statistical rank test representations of the performance (AUC) of EWFS and sequential search-based WFS methods on NB and DT classifiers. (**a**) Average AUC values of NB (**b**) Average AUC values of DT.

Furthermore, similar findings were observed in the case of the average f-measure value, as presented in Figure 13. NB and DT models based on the EWFS method outranked and outperformed sequential search-based WFS, as evidenced by significant statistical differences in the average f-measure values in favor of NB and DT models based on EWFS. In summary, Table 9 tabulates the Scott–Knott ESD statistical rank test results of EWFS and sequential search-based WFS (BFS, GDS, LFS, and SFS) methods.

As depicted in Table 9, it can be deduced that NB and DT models based on EWFS are superior and rank best when compared against models based on sequential search-based WFS methods. These findings indicate that there are significant statistical differences in the average performance of NB and DT models based on EWFS when compared with existing sequential search-based WFS methods based on average accuracy, average AUC, and average f-measure values. This further confirms and supports the superiority of the proposed EWFS over existing sequential search-based WFS methods such as BFS, GDS, LFS, and SFS in terms of selecting relevant features at a reasonable computational time in SDP processes.

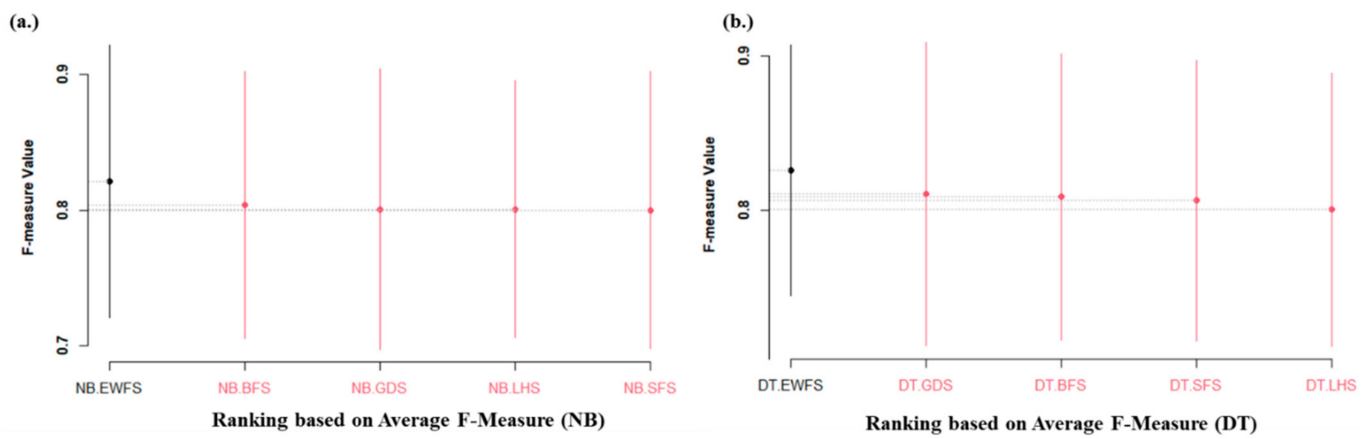

Figure 13. Scott–Knott ESD statistical rank test representations of the performance (f-measure) of EWFS and sequential search-based WFS methods on NB and DT classifiers. (**a**) Average F-measure values of NB (**b**) Average F-measure values of DT.

Table 9. Analysis of Scott–Knott ESD statistical rank test results of EWFS and sequential search-based WFS methods.

Statistical Ranking Based on Average Accuracy				Statistical Ranking Based on Average AUC				Statistical Ranking Based on Average F-Measure			
NB		DT		NB		DT		NB		DT	
Rank	**FS Methods**	**Rank**	**FS Methods**	**Rank**	**FS Methods**	**Rank**	**FS Methods**	**Rank**	**FS Methods**	**Rank**	**FS Methods**
1	EWFS, GDS, LFS, BFS, SFS	1	EWFS, GDS, BFS, SFS, LFS	1	EWFS	1	EWFS	1	EWFS,	1	EWFS
				2	SFS, BFS, LFS, GDS	2	LFS, BFS, SFS, BFS		BFS, GDS, LFS, SFS	2	GDS, BFS, SFS, LFS

In summary, the proposed EWFS approach outperformed current metaheuristic and sequential search-based WFS approaches on the analyzed defect datasets in terms of positive effects on the predictive performance of SDP models (NB and DT). These findings, therefore, answer RQ1 and RQ2 (see Section 3.4), as presented in Table 10. As a result, we recommend enhancing the subset evaluation process of WFS methods by deploying a dynamic re-ranking strategy that iteratively selects germane features with a low subset evaluation cycle, while not compromising the prediction performance of the ensuing model.

Table 10. Answers to research questions.

Research Questions	Answers
RQ1. How effective is the EWFS method in comparison to existing metaheuristic search-based WFS methods?	The proposed EWFS method outperforms metaheuristics search-based WFS with significant differences.
RQ2. How effective is the EWFS method in comparison to existing sequential search-based WFS methods?	The proposed EWFS outperforms sequential search-based WFS with significant differences.

5. Conclusions and Future Work

This study focuses on resolving the high dimensionality problem in software defect prediction. Specifically, this study addresses the local minima stagnation and computational cost problems of the WFS method by proposing a novel EWFS method. The proposed EWFS method dynamically and iteratively selects optimal features with good prediction performance values for SDP. The proposed EWFS, in particular, was able to produce a more robust and complete subset of features that best represented the datasets we studied.

Thus, these results justify the use of a dynamic re-ranking mechanism in the WFS process to improve the subset evaluation time and prediction efficacies of SDP models.

In a broader context, the observations and conclusions of this study can be used by experts and researchers in SDP and other relevant research domains who utilize WFS methods to solve the high dimensionality problem.

As a drawback of this research, there is a need to explore and broaden the application of this research later on by analyzing other WFS system re-ranking configurations with more prediction models. Furthermore, the effects of using ensemble-based WFS as opposed to a single classifier in WFS is worth exploring.

Author Contributions: Conceptualization, A.O.B. (Abdullateef Oluwagbemiga Balogun), S.B., L.F.C. and V.E.A.; Data curation, A.O.B. (Abdullateef Oluwagbemiga Balogun), A.A.I., A.K.A., A.O.B. (Amos Orenyi Bajeh) and G.K.; Formal analysis, A.O.B. (Abdullateef Oluwagbemiga Balogun), A.A.I., M.A.A. and A.K.A.; Funding acquisition, S.M.; Investigation, A.O.B. (Abdullateef Oluwagbemiga Balogun), A.K.A. and G.K.; Methodology, A.O.B. (Abdullateef Oluwagbemiga Balogun) and V.E.A.; Project administration, S.B., L.F.C. and G.K.; Resources, A.O.B. (Abdullateef Oluwagbemiga Balogun), S.M., M.A.A., A.O.B. (Amos Orenyi Bajeh) and G.K.; Software, A.O.B. (Abdullateef Oluwagbemiga Balogun), S.M. and A.O.B. (Amos Orenyi Bajeh); Supervision, S.B., L.F.C., S.M. and A.O.B. (Amos Orenyi Bajeh); Validation, L.F.C., A.A.I., M.A.A. and V.E.A.; Visualization, A.A.I. and M.A.A.; Writing—original draft, A.O.B. (Abdullateef Oluwagbemiga Balogun); Writing—review & editing, Abdullateef Oluwagbemiga Balogun, L.F.C. and V.E.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Yayasan Universiti Teknologi PETRONAS (YUTP) Research Grant Scheme under grant numbers (YUTP-FRG/015LC0240) and (YUTP-FRG/015LC0297).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Available on request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Afzal, W.; Torkar, R. Towards benchmarking feature subset selection methods for software fault prediction. In *Computational Intelligence and Quantitative Software Engineering*; Springer: New York, NY, USA, 2016; pp. 33–58.
2. Akintola, A.G.; Balogun, A.; Lafenwa-Balogun, F.B.; Mojeed, H.A. Comparative Analysis of Selected Heterogeneous Classifiers for Software Defects Prediction Using Filter-Based Feature Selection Methods. *FUOYE J. Eng. Technol.* **2018**, *3*, 134–137. [CrossRef]
3. Alazzawi, A.K.; Rais, H.M.; Basri, S.; Alsariera, Y.A.; Capretz, L.F.; Balogun, A.O.; Imam, A.A. HABCSm: A Hamming Based t-way Strategy based on Hybrid. Artificial Bee Colony for Variable Strength Test. Sets Generation. *Int. J. Comput. Commun. Control.* **2021**, *16*, 1–18. [CrossRef]
4. Bajeh, A.O.; Oluwatosin, O.J.; Basri, S.H.U.I.B.; Akintola, A.G.; Balogun, A.O. Object-oriented measures as testability indicators: An empirical study. *J. Eng. Sci. Technol.* **2020**, *15*, 1092–1108.
5. Balogun, A.; Bajeh, A.; Mojeed, H.; Akintola, A. Software defect prediction: A multi-criteria decision-making approach. *Niger. J. Technol. Res.* **2020**, *15*, 35–42. [CrossRef]
6. Ameen, A.O.; Mojeed, H.A.; Bolariwa, A.T.; Balogun, A.O.; Mabayoje, M.A.; Usman-Hamzah, F.E.; Abdulraheem, M. Application of shuffled frog-leaping algorithm for optimal software project scheduling and staffing. In *International Conference of Reliable Information and Communication Technology*; Springer: New York, NY, USA, 2020.
7. Balogun, A.O.; Lafenwa-Balogun, F.B.; Mojeed, H.A.; Usman-Hamza, F.E.; Bajeh, A.O.; Adeyemo, V.E.; Adewole, K.S.; Jimoh, R.G. Data sampling-based feature selection framework for software defect prediction. In *The International Conference on Emerging Applications and Technologies for Industry 4.0*; Springer: Cham, Switzerland, 2020.
8. Chauhan, A.; Kumar, R. Bug severity classification using semantic feature with convolution neural network. In *Computing in Engineering and Technology*; Springer: Cham, Switzerland, 2020; pp. 327–335.
9. Jimoh, R.G.; Balogun, A.O.; Bajeh, A.O.; Ajayi, S. A PROMETHEE based evaluation of software defect predictors. *J. Comput. Sci. Its Appl.* **2018**, *25*, 106–119.
10. Gotlieb, A. Exploiting symmetries to test programs. In Proceedings of the 14th International Symposium on Software Reliability Engineering, Denver, CO, USA, 17–21 November 2003.
11. Alazzawi, A.K.; Rais, H.M.; Basri, S.; Alsariera, Y.A.; Balogun, A.O.; Imam, A.A. A hybrid artificial bee colony strategy for t-way test set generation with constraints support. *J. Phys. Conf. Ser.* **2020**, *1529*, 042068. [CrossRef]

12. Catal, C.; Diri, B. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Inf. Sci.* **2009**, *179*, 1040–1058. [CrossRef]
13. Li, L.; Leung, H. Mining static code metrics for a robust prediction of software defect-proneness. In Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement, Banff, AB, Canada, 22–23 September 2011.
14. Mabayoje, M.A.; Balogun, A.O.; Bajeh, A.O.; Musa, B.A. Software defect prediction: Effect of feature selection and ensemble methods. *FUW Trends Sci. Technol. J.* **2018**, *3*, 518–522.
15. Aleem, S.; Capretz, L.F.; Ahmed, F. Comparative performance analysis of machine learning techniques for software bug detection. In Proceedings of the 4th International Conference on Software Engineering and Applications, Vienna, Austria, 19–20 December 2015.
16. Lessmann, S.; Baesens, B.; Mues, C.; Pietsch, S. Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. *IEEE Trans. Softw. Eng.* **2008**, *34*, 485–496. [CrossRef]
17. Li, N.; Shepperd, M.; Guo, Y. A systematic review of unsupervised learning techniques for software defect prediction. *Inf. Softw. Technol.* **2020**, *122*, 106287. [CrossRef]
18. Okutan, A.; Yıldız, O.T. Software defect prediction using Bayesian networks. *Empir. Softw. Eng.* **2012**, *19*, 154–181. [CrossRef]
19. Rodriguez, D.; Herraiz, I.; Harrison, R.; Dolado, J.; Riquelme, J.C. Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, UK, 13–14 May 2014.
20. Usman-Hamza, F.; Atte, A.; Balogun, A.; Mojeed, H.; Bajeh, A.; Adeyemo, V. Impact of feature selection on classification via clustering techniques in software defect prediction. *J. Comput. Sci. Appl.* **2020**, *26*, 73–88. [CrossRef]
21. Balogun, A.; Oladele, R.; Mojeed, H.; Amin-Balogun, B.; Adeyemo, V.E.; Aro, T.O. Performance analysis of selected clustering techniques for software defects prediction. *Afr. J. Comput. ICT* **2019**, *12*, 30–42.
22. Rodriguez, D.; Ruiz, R.; Cuadrado-Gallego, J.; Aguilar-Ruiz, J.; Garre, M. Attribute selection in software engineering datasets for detecting fault modules. In Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007), Lubeck, Germany, 28–31 August 2007.
23. Wang, H.; Khoshgoftaar, T.M.; van Hulse, J.; Gao, K. Metric selection for software defect prediction. *Int. J. Softw. Eng. Knowl. Eng.* **2011**, *21*, 237–257. [CrossRef]
24. Rathore, S.S.; Gupta, A. A comparative study of feature-ranking and feature-subset selection techniques for improved fault prediction. In Proceedings of the 7th India Software Engineering Conference, Chennai, India, 19–21 February 2014.
25. Xu, Z.; Liu, J.; Yang, Z.; An, G.; Jia, X. The impact of feature selection on defect prediction performance: An empirical comparison. In Proceedings of the IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, ON, Canada, 23–27 October 2016.
26. Balogun, A.O.; Basri, S.; Jadid, S.A.; Mahamad, S.; Al-momani, M.A.; Bajeh, A.O.; Alazzawi, A.K. Search-based wrapper feature selection methods in software defect prediction: An empirical analysis. In *Computer Science On-line Conference*; Springer: Cham, Switzerland, 2020.
27. Ghotra, B.; McIntosh, S.; Hassan, A.E. A large-scale study of the impact of feature selection techniques on defect classification models. In Proceedings of the IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, Argentina, 20–28 May 2017.
28. Balogun, A.O.; Basri, S.; Mahamad, S.; Abdulkadir, S.J.; Almomani, M.A.; Adeyemo, V.E.; Al-Tashi, Q.; Mojeed, H.A.; Imam, A.A.; Bajeh, A.O. Impact of Feature Selection Methods on the Predictive Performance of Software Defect Prediction Models: An Extensive Empirical Study. *Symmetry* **2020**, *12*, 1147. [CrossRef]
29. Balogun, A.O.; Basri, S.; Capretz, L.F.; Mahamad, S.; Imam, A.A.; Almomani, M.A.; Adeyemo, V.E.; Kumar, G. An adaptive rank aggregation-based ensemble multi-filter feature selection method in software defect prediction. *Entropy* **2021**, *23*, 1274. [CrossRef]
30. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Hashim, A.S. Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach. *Appl. Sci.* **2019**, *9*, 2764. [CrossRef]
31. Anbu, M.; Mala, G.S.A. Feature selection using firefly algorithm in software defect prediction. *Clust. Comput.* **2017**, *22*, 10925–10934. [CrossRef]
32. Kakkar, M.; Jain, S. Feature selection in software defect prediction: A comparative study. In Proceedings of the 6th International Conference on Cloud System and Big Data Engineering, Noida, India, 14–15 January 2016.
33. Al-Tashi, Q.; Kadir, S.J.A.; Rais, H.M.; Mirjalili, S.; Alhussian, H. Binary Optimization Using Hybrid Grey Wolf Optimization for Feature Selection. *IEEE Access* **2019**, *7*, 39496–39508. [CrossRef]
34. Al-Tashi, Q.; Rais, H.; Jadid, S. Feature selection method based on grey wolf optimization for coronary artery disease classification. In Proceedings of the 3rd International Conference of Reliable Information and Communication Technology (IRICT), Kuala Lumpur, Malaysia, 23–24 July 2018.
35. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Sobri, A.H. A hybrid multi-filter wrapper feature selection method for software defect predictors. *Int. J. Supply Chain. Manag.* **2019**, *8*, 916–922.
36. Gao, K.; Khoshgoftaar, T.M.; Wang, H.; Seliya, N. Choosing software metrics for defect prediction: An investigation on feature selection techniques. *Software Pr. Exp.* **2011**, *41*, 579–606. [CrossRef]
37. Wahono, R.S.; Suryana, N.; Ahmad, S. Metaheuristic optimization based feature selection for software defect prediction. *J. Softw.* **2014**, *9*, 1324–1333. [CrossRef]

38. Song, Q.; Jia, Z.; Shepperd, M.; Ying, S.; Liu, J. A General Software Defect-Proneness Prediction Framework. *IEEE Trans. Softw. Eng.* **2010**, *37*, 356–370. [CrossRef]
39. Muthukumaran, K.; Rallapalli, A.; Murthy, N.B. Impact of feature selection techniques on bug prediction models. In Proceedings of the 8th India Software Engineering Conference, Bangalore, India, 18–20 February 2015.
40. Rodríguez, D.; Ruiz, R.; Cuadrado-Gallego, J.; Aguilar-Ruiz, J. Detecting fault modules applying feature selection to classifiers. In Proceedings of the IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, USA, 13–15 August 2007.
41. Cynthia, S.T.; Rasul, M.G.; Ripon, S. Effect of feature selection in software fault detection. In *International Conference on Multi-disciplinary Trends in Artificial Intelligence*; Springer: Cham, Switzerland, 2019.
42. Ekundayo, A. Wrapper feature selection based heterogeneous classifiers for software defect prediction. *Adeleke Univ. J. Eng. Technol.* **2019**, *2*, 1–11.
43. Ding, Z. Isolation forest wrapper approach for feature selection in software defect prediction. In *IOP Conference Series: Materials Science and Engineering*; IOP Publishing: Bristol, UK, 2021.
44. Yu, Q.; Jiang, S.; Zhang, Y. The performance stability of defect prediction models with class imbalance: An empirical study. *IEICE Trans. Inf. Syst.* **2017**, *100*, 265–272. [CrossRef]
45. Bermejo, P.; Gámez, J.A.; Puerta, J.M. Adapting the CMIM algorithm for multilabel feature selection. A comparison with existing methods. *Expert Syst.* **2017**, *35*, e12230. [CrossRef]
46. Yang, X.-S.; Deb, S. Multiobjective cuckoo search for design optimization. *Comput. Oper. Res.* **2013**, *40*, 1616–1624. [CrossRef]
47. Shepperd, M.; Song, Q.; Sun, Z.; Mair, C. Data Quality: Some Comments on the NASA Software Defect Datasets. *IEEE Trans. Softw. Eng.* **2013**, *39*, 1208–1215. [CrossRef]
48. Kondo, M.; Bezemer, C.-P.; Kamei, Y.; Hassan, A.E.; Mizuno, O. The impact of feature reduction techniques on defect prediction models. *Empir. Softw. Eng.* **2019**, *24*, 1925–1963. [CrossRef]
49. Wu, R.; Zhang, H.; Kim, S.; Cheung, S.C. Relink: Recovering links between bugs and changes. In Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, Szeged, Hungary, 5–9 September 2011.
50. Song, Q.; Guo, Y.; Shepperd, M. A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction. *IEEE Trans. Softw. Eng.* **2018**, *45*, 1253–1269. [CrossRef]
51. Nam, J.; Fu, W.; Kim, S.; Menzies, T.; Tan, L. Heterogeneous defect prediction. *IEEE Trans. Softw. Eng.* **2017**, *44*, 874–896. [CrossRef]
52. Tantithamthavorn, C.; McIntosh, S.; Hassan, A.E.; Matsumoto, K. The Impact of Automated Parameter Optimization on Defect Prediction Models. *IEEE Trans. Softw. Eng.* **2018**, *45*, 683–711. [CrossRef]
53. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Mahamad, S.; Al-momamni, M.A.; Imam, A.A.; Kumar, G.M. Rank aggregation based multi-filter feature selection method for software defect prediction. In Proceedings of the International Conference on Advances in Cyber Security, Penang, Malaysia, 30 July–1 August 2020.
54. Balogun, A.O.; Basri, S.; Mahamad, S.; Abdulkadir, S.J.; Capretz, L.F.; Imam, A.A.; Almomani, M.A.; Adeyemo, V.E.; Kumar, G. Empirical analysis of rank aggregation-based multi-filter feature selection methods in software defect prediction. *Electronics* **2021**, *10*, 179. [CrossRef]
55. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer: New York, NY, USA, 2013; Volume 112.
56. Kuhn, M.; Johnson, K. *Applied Predictive Modeling*; Springer: Berlin, Germany, 2013; Volume 26.
57. Balogun, A.O.; Adewole, K.S.; Raheem, M.O.; Akande, O.N.; Usman-Hamza, F.E.; Mabayoje, M.A.; Akintola, A.G.; Asaju-Gbolagade, A.W.; Jimoh, M.K.; Jimoh, R.G.; et al. Improving the phishing website detection using empirical analysis of Function Tree and its variants. *Heliyon* **2021**, *7*, e07437. [CrossRef]
58. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA data mining software: An update. *ACM SIGKDD Explor. Newsl.* **2009**, *11*, 10–18. [CrossRef]
59. Crawley, M.J. *The R Book*; John Wiley & Sons: Hoboken, NJ, USA, 2012.