









Article

An Adaptive Rank Aggregation-Based Ensemble Multi-Filter Feature Selection Method in Software Defect Prediction

Abdullateef O. Balogun ^{1,2,*} , Shuib Basri ¹ , Luiz Fernando Capretz ³ , Saipunidzam Mahamad ¹ ,
Abdullahi A. Imam ¹ , Malek A. Almomani ⁴ , Victor E. Adeyemo ⁵  and Ganesh Kumar ¹ 

¹ Department of Computer and Information Science, Universiti Teknologi PETRONAS, Bandar Seri Iskandar 32610, Malaysia; shuib_basri@utp.edu.my (S.B.); saipunidzam_mahamad@utp.edu.my (S.M.); imam.abubakar@utp.edu.my (A.A.I.); ganesh_17005106@utp.edu.my (G.K.)

² Department of Computer Science, University of Ilorin, Ilorin 1515, Nigeria

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, Leeds Beckett University, Headingley Campus, Leeds LS6 3QS, UK; v.adeyemo@leedsbeckett.ac.uk

* Correspondence: abdullateef_16005851@utp.edu.my or balogun.ao1@unilorin.edu.ng



Citation: 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. <https://doi.org/10.3390/e23101274>

Academic Editors: Nan Chen, Honghu Liu and Evelyn Lunasin

Received: 15 August 2021

Accepted: 26 September 2021

Published: 29 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: Feature selection is known to be an applicable solution to address the problem of high dimensionality in software defect prediction (SDP). However, choosing an appropriate filter feature selection (FFS) method that will generate and guarantee optimal features in SDP is an open research issue, known as the filter rank selection problem. As a solution, the combination of multiple filter methods can alleviate the filter rank selection problem. In this study, a novel adaptive rank aggregation-based ensemble multi-filter feature selection (AREMFFS) method is proposed to resolve high dimensionality and filter rank selection problems in SDP. Specifically, the proposed AREMFFS method is based on assessing and combining the strengths of individual FFS methods by aggregating multiple rank lists in the generation and subsequent selection of top-ranked features to be used in the SDP process. The efficacy of the proposed AREMFFS method is evaluated with decision tree (DT) and naïve Bayes (NB) models on defect datasets from different repositories with diverse defect granularities. Findings from the experimental results indicated the superiority of AREMFFS over other baseline FFS methods that were evaluated, existing rank aggregation based multi-filter FS methods, and variants of AREMFFS as developed in this study. That is, the proposed AREMFFS method not only had a superior effect on prediction performances of SDP models but also outperformed baseline FS methods and existing rank aggregation based multi-filter FS methods. Therefore, this study recommends the combination of multiple FFS methods to utilize the strength of respective FFS methods and take advantage of filter–filter relationships in selecting optimal features for SDP processes.

Keywords: feature selection; high dimensionality; rank aggregation; software defect prediction

1. Introduction

Software development lifecycle (SDLC) is a structured mechanism explicitly designed and developed for the creation or development of top-quality software systems. To maintain a timely and efficient software system, the incremental measures contained in SDLC, such as requirement elicitation, software system analysis, software system design, and maintenance, should be closely followed and implemented [1,2]. However, since the stepwise processes in SDLC are performed by human experts, mistakes or errors are unavoidable. Currently, due to the large size and dependencies in modules or components of software systems, these mistakes are prominent and re-occurring. Consequently, if not

corrected promptly, these mistakes lead to unstable software systems and finally to software failure. That is, the presence of errors in modules or components of software systems will result in defective and low-quality software systems. In addition, glitches in software systems create frustration for end-users and stakeholders when the failed software system does not function as proposed even after exhausting limited resources (time and effort) [3,4]. Therefore, it is crucial to identify and predict software defects before product release or during the software development phase. Early prediction and identification of faulty modules or components in a software system will allow such modules or components to be spontaneously corrected and the available resources to be used judiciously [5,6].

Software defect prediction (SDP) involves the implementation of machine learning (ML) methods to assess the defectivity of modules or components in a software system. In particular, SDP deploys ML methods on software features that are defined by software metrics to contain defects in software modules or components [7–9]. Several studies have proposed and implemented both supervised and unsupervised forms of ML methods for SDP [10–15]. Nevertheless, the predictive performance of SDP models is flatly dependent on the quality and inherent nature of the software datasets used for developing such SDP models. Software metrics used to characterize the quality and efficiency of software are directly related to its magnitude and complexities. In other words, broad and scalable software systems will require a variety of software metric mechanisms to produce features that better represent the quality of such software systems [16,17]. Generally, software systems with a significant number of features, due to the accumulation of software metrics, are composed mainly of redundant and irrelevant features which can be defined as a problem of high dimensionality.

Several research findings have shown that the high dimensionality of software metrics adversely impacts the predictive efficiency of SDP models [18,19]. The feature selection (FS) method is recognized by many researchers as an effective way of resolving high dimensionality problems. Principally, these FS methods simply extract useful and critical software features from the original software defect dataset for every SDP operation [20–23]. The implementation of FS methods will then 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. In other words, 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 [24,25].

FS methods can be categorized into two groups: filter feature selection (FFS) and wrapper feature selection (WFS). FFS methods evaluate features of a dataset using inherent numerical or statistical properties from the dataset. Subsequently, top-ranked features are chosen based on the predefined threshold value. Contrary to FSS, WFS methods evaluate features of a dataset based on its effectiveness in enhancing the performance of underlining classifiers. That is, WFS selects features based on classifier performance. This characteristic renders WFS computationally costly and difficult to implement as it often leads to biased and over-fitting models [26–28]. Based on this attribute of WFS, researchers usually prefer FSS methods in SDP [18,29]. Nonetheless, selecting a fitting FFS method for SDP is a problem. This is based on findings from existing studies on the impact of FSS in SDP, which concluded that there is no one best FSS method and that their respective performances depend on selected datasets and classifiers [15,19,21,22]. This observation can be due to incomplete and disjointed feature ranking of FFS methods in SDP. Also, the complex and different implicit operational behaviours of FFS methods are another factor that affects the selection of FSS methods in SDP [18,30]. This problem can be described as the filter rank selection problem in SDP, and a viable solution is to hybridize FFS methods by combining and aggregating the rank lists from each FFS method into a single robust and non-disjoint rank list [31,32].

Therefore, this study proposes a novel adaptive rank aggregation-based ensemble multi-filter FS method with a backtracking function (AREMFFS) for SDP. Specifically, the AREMFFS method is based on evaluating and combining the strengths of individual FFS

methods by aggregating multiple rank lists in the generation and subsequent selection of top-ranked features to be used in the SDP process.

The main contributions of this study are as follows:

1. To develop a novel adaptive rank aggregation-based ensemble multi-filter FS method with a backtracking function (AREMFFS).
2. To empirically evaluate and validate the performance of AREMFFS against baseline FFS methods and existing rank aggregation-based FS methods.
3. To validate the application of AREMFFS as a solution for the filter rank selection problem and high dimensionality in SDP.

The remainder of this paper is structured as follows: reviews on existing related works are presented in Section 2; details on proposed AREMFFS and experimental methods are described in Section 3. Experimental results are analysed and discussed in Section 4; and the research is concluded with highlights of future works in Section 5.

2. Related Works

Software testing is a vital stage in SDLC as it involves the detection of defects or bugs in the codebase of software systems. The challenge in this stage is to accurately detect the defective source code. Developing an SDP solution is a difficult issue, and numerous approaches have been suggested in the literature.

Early SDP solutions are primarily based on assessing the software requirement specification (SRS) documents for possible flaws. For instance, Smidts, Stutzke [33] developed a software reliability model based on SRS and failure modes. Data from the failure mode(s) are passed into the suggested model as input data. Also, Cortellessa, Singh [34] combined the unified modelling language (UML) of software architecture and a Bayesian framework for SDP. These approaches, unfortunately, do not provide or allow code re-use, which indirectly implies that component failures are independent of one another. Similarly, Gaffney and Davis [35,36] suggested a phase-based model for software reliability. The model is mostly based on defect data discovered after a study of different software development stages. However, this approach is restrictive as it is usually tailored to a particular organization. In another study, Al-Jamimi [37] investigated the use of fuzzy logic in SDP. Specifically, Takagi–Sugeno fuzzy inference engine was used for SDP. Also, Yadav and Yadav [38] successfully implemented a fuzzy logic method for phase-wise SDP. They aimed to provide a generic set of features for SDP. Nonetheless, the decidability issue is a prominent short-fall of fuzzy logic [39].

In recent times, as a result of huge code bases, many ML techniques have been developed for SDP. For instance, Khan, Naseem [40] investigated the performance of several supervised ML techniques for SDP. In a similar study, Naseem, Khan [41] explored the performance tree-based classifiers for SDP. Also, Akimova, Bersenev [42] experimented with the performance of deep learning methods in SDP. Findings from these studies indicated that ML approaches can be utilized for detecting defects in SDP processes. Similarly, evolutionary computation (EC) techniques have been deployed for SDP tasks. In particular, Haouari, Souici-Meslati [43] investigated the performance of artificial immune systems, which are bio-inspired ML techniques based on the mammalian immune paradigms for SDP. Also, Khurma, Alsawalqah [44] proposed an efficient binary variant of moth flame optimization (BMFO) for SDP. Concerning unsupervised ML techniques, Xu, Li [45] investigated the applicability and performance of 40 clustering techniques for SDP. In addition, Marjuni, Adji [46] developed an unsupervised ML technique named signed Laplacian-based spectral classifier for SDP. The unsupervised ML technique is applied when working on unlabelled datasets, unlike supervised ML [14]. However, the performance of an ML technique depends largely on the quality of the datasets used for training such an ML technique [47–49].

It is important to highlight that SDP is not the only method that can be used to find defects in software systems. For instance, model checking [50] and static code analysis (e.g., Coverity [51]) can be deployed to find defects in software systems. The duo of

model checking and static analysis are conventional methods of finding defects in software systems and are based on fault localization. That is, they use the discrepancies between the inputs of failed and successful software tests to identify problems in the source code. These conventional methods often detect flaws in the present code base only (i.e., the codebase being examined), while SDP warns about future defect-prone regions of a software system.

High dimensionality is a data quality problem that affects the predictive performances of SDP models. In other words, the occurrence of redundant and noisy software features due to the amount and increase in the software metrics deployed for determining the quality of a software system has a negative effect on prediction models in SDP. Findings from existing studies have indicated that FS methods can be used to resolve the high dimensionality problem. As such, several studies have proposed diverse FS methods and investigated the effects of these on the predictive performance of SDP models.

Cynthia, Rasul [52] investigated the impact of FS methods on prediction models in SDP. Specifically, the impact of five FS methods was investigated on selected classifiers. Based on their experimental results, they concluded that FS methods have a significant effect (positive) on the prediction performance of the selected classifiers. However, their study was limited in terms of the scope (number of selected FS methods and datasets evaluated) of the study. Also, Akintola, Balogun [1] conducted a comparative study on filter-based FS methods on heterogeneous prediction models. Principal component analysis (PCA), correlation-based feature selection (CFS), and filtered subset evaluation (FSE) were applied on selected classifiers. They also observed that applying FS methods in SDP is beneficial as it improves the prediction performances of selected classifiers. In another study by Balogun, Basri [20], the authors studied the effect of FS methods on SDP models in terms of applied search methods. They argued that the search method used by an FS method could affect its performance. Eighteen FS methods with four classifiers were considered for experimentation. Observations from their results support the use of FS in SDP. They also posited that the performance of FS methods depends on the datasets and classifiers used in the experiments. That is, there is no one best FS method, and by extension, selecting a suitable FS method to be used in SDP becomes a problem. This observation can be denoted as the filter rank selection problem in SDP. Balogun, Basri [21] extended their initial study [20] by conducting a large-scale empirical FS study on SDP. The empirical study was anchored on findings accentuated by Ghotra, McIntosh [53] and Xu, Liu [54] in their respective studies. Findings from their study suggested that the performance of FS methods relies on the choice of datasets and classifiers. Hence, there are no best FS methods. This observation further alludes to the filter rank selection problem of FS methods in SDP.

As a viable solution to the filter rank selection problem, Jia [55], in their study, recommended a hybrid FS method that combines three filter FS methods (chi-squared, information gain, and correlation filter). The proposed hybrid FS method selects features from each rank list based on a pre-determined (TopK) value. It was observed that SDP models based on the proposed hybrid FS method were superior to models with the experimental baseline FS methods. However, it should be noted that the overall ranking of features can be influenced by the distorted ranks of each feature [56]. Moreover, the choice of predetermined TopK features is not always the right approach, as important and relevant features may be overlooked during the feature selection [52]. To address the filter rank selection problem, Wang, Khoshgoftaar [57] studied the ensemble of FS techniques in SDP. Using 18 distinct filter FS techniques, 17 ensemble methods were developed. The ensemble techniques were based on averaging feature rankings from separate rank lists. They reported the advantages of the ensemble methods based on their experimental findings. However, as Jia [55] pointed out, average rank lists of characteristics may be influenced by the skewed rankings of each feature.

Xia, Yan [58] used ReliefF and correlation analysis for feature selection in metric-based SDP. The suggested technique (ReliefF-Lc) simultaneously evaluates correlation and redundancy between modules. According to their research findings, ReliefF-Lc surpasses other studied techniques (IG and REF). Malik, Yining [59] have carried out an empirical

comparison study on the usage of an attribute rank approach. The usefulness of principal component analysis (PCA) with the ranker search technique as a filter FS method was specifically explored. They concluded that using PCA with a ranker search technique in the SDP process may enhance the efficacy of classifiers in SDP. Despite the fact that their results cannot be generalised owing to the restricted scope of their research, they do agree with previous SDP studies on the use of FS techniques in SDP.

Iqbal and Aftab [29] created an SDP framework that makes use of multi-filter FS and multi-layer perceptron (MLP). In addition, to solve the intrinsic class imbalance issue, a random over-sampling (ROS) method was included. The suggested multi-filter was created by combining four distinct search methods with correlation feature selection (CFS). Based on their findings, they determined that the multi-filter approach with ROS outperformed the other methods tested.

Similarly, Balogun, Basri [31] addressed the filter rank selection problem in SDP by proposing a rank aggregation-based multi-filter FS method, and in another study, Balogun, Basri [32] conducted an empirical comparative analysis of rank aggregation-based FS methods in SDP. Findings from both studies showed that using rank aggregation methods to aggregate multiple rank lists produced by individual FS methods can address the filter rank selection problem in SDP.

Based on the preceding reviews and findings, this study, therefore, presents an adaptive rank aggregation-based ensemble multi-filter FS method for the SDP process.

3. Methodology

Details on the choice of classifiers, FS methods, proposed AREMFFS, experimental procedure, datasets, and evaluation measures are provided in this section.

3.1. Classification Algorithms

In this study, decision tree (DT) and naïve Bayes (NB) algorithms are selected and deployed as prediction models. The choice of the duo is due to their respective high prediction performance and ability to work on imbalanced datasets [20,60]. Also, DT and NB are typically not affected by parameter tuning. In addition, DT and NB have been used repeatedly in existing SDP studies. As shown in Table 1, the *subTreeRaising* parameter of DT is set to *True* to allow the pruning of trees by pushing nodes upwards towards the tree's root, replacing other nodes along the way. Also, the *ConfidenceFactor* is at a 0.25 threshold value and the minimum number of objects on a leaf (*MinObj*) is set to 2. The essence of these parameters for the DT classifier is to ensure a simpler model with a higher number of nodes (samples) [61]. Concerning the NB classifier, a kernel estimator is used for numeric attributes and the number of decimal places to be used for the output of numbers in the model is set to 2 [62].

Table 1. Classification algorithms.

Classification Algorithms	Parameter Settings
Decision Tree (DT)	ConfidenceFactor = 0.25; MinObj = 2; subTreeRaising = True
Naïve Bayes (NB)	NumDecimalPlaces = 2; UseEstimator = True

3.2. Feature Selection Method

Concerning the baseline FS methods, three filter FS methods with diverse computational characteristics were chosen in this study. Specifically, chi-square (CS), Relief (REF), and information gain (IG) were deployed as baseline FS methods. CS is a statistics-based FS method that assesses the degree of independence of an attribute from the class label. REF, an instance-based FS method, samples features of a given dataset and then compares each sampled feature in its respective neighbourhood and thereafter assigns a relevance score to each feature. Finally, the IG method selects features using an entropy mechanism that is

based on selecting relevant features by minimizing uncertainties associated with identifying the class label when the value of the feature is unknown. The specific selection of these FS methods (CS, REF, IG) is based on findings from existing studies as they have been regarded to have a high positive effect on prediction performances of SDP models [20,21]. Moreover, these FS methods (CS, REF, IG) are selected to introduce heterogeneity as each of the FS methods have different and unique computational mechanisms. More details on the selected FS methods can be found in [31,32,63–66]

3.3. Adaptive Rank Aggregation-Based Ensemble Multi-Filter Feature Selection (AREMFFS) Method

The proposed AREMFFS method aims to take the computational capabilities of multiple independent filter FS methods into account and incorporate their strengths. The objective of this proposed method is to address filter method selection problems by selecting features and generating a robust rank list from multiple filter FS methods in SDP tasks. AREMFFS can be divided into three stepwise phases: the multi-filter FS phase, the ensemble rank aggregation phase, and the backtracking function phase.

3.3.1. Multi-Filter FS Phase

Individual rank lists from the CS, REF, and IG filter FS methods are constructed from the given datasets as shown in Algorithm 1. The multiple rank lists generated by the independent filter FS methods are mutually exclusive as each filter FS method under consideration has distinct computational features. This is undertaken to guarantee that varied representations of features are chosen. Following that, the generated multiple rank lists are aggregated using rank aggregation functions shown in Table 2. Each rank aggregation function blends the multiple rank lists into a single aggregated rank list by using the significance score assigned to each feature on the individual rank lists.

Table 2. Rank aggregation methods.

Aggregators	Formula	Description
Min ()	$\min\{R_1(a_{1\dots n}), R_2(a_{1\dots n}), \dots, R_m(a_{1\dots n})\}$	Selects the <i>minimum</i> of the relevance scores produced by the aggregated rank list
Max ()	$\max\{R_1(a_{1\dots n}), R_2(a_{1\dots n}), \dots, R_m(a_{1\dots n})\}$	Selects the <i>maximum</i> of the relevance scores produced by the aggregated rank list
Mean ()	$\text{mean}(\sum_{i=1}^m R_i(a_{1\dots n})) \times \frac{1}{m}$	Selects the <i>mean</i> of the relevance scores produced by the aggregated rank list

Specifically, the *Minimum (Min)* and *Maximum (Max)* rank aggregation functions choose features based on the minimum and maximum significance score provided by the aggregated rank list, respectively. The *Arithmetic Mean (Mean)* rank aggregation function aggregates the multiple rank lists into a single aggregated rank list by calculating the arithmetic mean of the significance scores assigned to each element on the individual rank lists. This is done to ensure that each feature from each rank list receives equal representation and consideration. Features with poor significance ratings on the aggregated rank list indicate that they should be ranked low on each rank list and, as such, may be dropped. To select appropriate features, a novel dynamic and automated threshold value dependent on the geometric mean function is added to the aggregated rank list.

Algorithm 1. Pseudocode of Proposed AREMFFS.**Input:**

n: Total number of features in the dataset

N: Total Number of Filter Rank Method = |CS, REF, IG|

 T_1 : Threshold value for optimal features selections = $(\prod_{i=1}^n X_i)^{\frac{1}{n}} = \sqrt[n]{X_1 X_2 X_3 \dots X_n}$ T_2 : $\log_2 n$ A[]: Aggregators $A = \{ \min \{ R_1(a_{1\dots n}), R_2(a_{1\dots n}), \dots R_m(a_{1\dots n}) \}, \max \{ R_1(a_{1\dots n}), R_2(a_{1\dots n}), \dots R_m(a_{1\dots n}) \}, \text{mean} \{ (\sum_{i=1}^m R_i(a_{1\dots n})) \times \frac{1}{m} \}, \}$

P[]: Aggregated Features

 P_t^* []: Optimal Features Selected from Aggregated Rank List based on T**Output:** M_t^* [] – Single rank list based on AREMFFS

// Multi-Filter Feature Selection Phase

```

1. for i = 1 to N { do
2.     Generate Rank list  $R_n$  for each filter rank method  $i$ 
3. }
4. Generate Aggregated Rank list using Aggregator functions:
for i=1 to len(A[]) { do
5.      $P_t^*[i]$  // Initialise variable to hold optimal features
6.      $P_i = A_i$ 
7.     for i=1 to  $P_i[N_i]$  { do
8.         if ( $P_i[i] \leq T_1$ )
9.              $P_t^*[i] \leftarrow P_i[i]$  based on T
10.        }
11.    }
//Ensemble Rank Aggregation Phase
12. for each feature  $f$  in  $P_t^*[i]$  s.t.  $i = 1, \dots, A_n$  { do // compute the frequency of each feature in the
        aggregated lists
13.         if feature  $f_j \in P_t^*[i]$ 
14.              $j = \text{count}(f_j)$ 
15.         if ( $j \geq A_{n-1}$ )
16.              $M[] \leftarrow \text{feature } f_j$  //append feature
} //select most occurring feature  $f$  in the aggregated list
17. for i = 1 to N { do
18.     Generate Rank list  $R_n$  for each filter rank method  $i$ 
19.     for i = 1 to  $R_n$  { do
20.          $R_i'[i] \leftarrow \text{TopK features of } R_n \text{ based on } T_2$ 
21.     }
22. }
//Backtracking function Phase
23. for i = 1 to len(M[]) { do
24.     for j = 1 to  $R_i'[N]$  { do
25.         if (feature  $f_i \in R_i'[j]$ )
26.              $g = \text{count}(f_i)$ 
27.         }
28.         if ( $g \geq R_{n-1}$ )
29.              $M_t^*[] \leftarrow \text{feature } f_i$ 
30.     }
31. return  $M_t^*[]$ 

```

The geometric mean of the aggregated significance score is calculated and features with aggregated significance scores less than or equal to the calculated threshold values are chosen. In its calculation, the geometric mean functions take into account feature dependence and the compounding effect. At the end of this phase, aggregated rank lists are generated, based on the *Min*, *Max*, and *Mean* rank aggregation functions. The preference for using *Min*, *Max*, and *Mean* rank aggregation functions is primarily based on the experimental findings reported in [32]. Balogun, Basri [32] deduced that the selected

rank aggregation functions (*Min*, *Max*, and *Mean*) can generate a more stable and complete subset of features that best represent studied datasets.

In the next phase of the AREMFFS method, the resulting aggregated rank lists produced by the respective rank aggregation function are ensembled based on the majority voting mechanism to create a single rank list.

3.3.2. Ensemble Rank Aggregation Phase

In this phase, the resulting rank lists generated by the respective rank aggregation functions (*Min*, *Max*, and *Mean*) are ensembled based on a majority voting mechanism to produce a single rank list. As observed, the generated rank lists from each of the rank aggregators are mutually exclusive subsets of features that are considered relevant and important based on each rank aggregator method. Specifically, the ensemble rank aggregation phase of the proposed AREMFFS method selects features that have a frequency value greater than or equal to $n - 1$, where n is the number of aggregated rank lists. In this study, the number of ensembled rank aggregators is 3 ($n = 3$); hence, for a feature to be selected in the ensemble rank aggregation phase, the feature must be a subset of the $n - 1$ rank list generated by the *Min*, *Max*, and *Mean* rank aggregator methods. This approach aims to combine rank lists generated by the respective rank aggregators into a single robust list that best represents each of the combined rank lists. Each rank aggregator has its advantages and disadvantages, and any rank aggregator method deployed should guarantee diversity while increasing the regularity of the FS process, to take advantage of the rank aggregators to boost the prediction performance of SDP models. In addition, the resulting rank list from the ensemble rank aggregation phase is expected to both produce more stable results and minimize the risk of selecting an unstable subset of features. However, there is a need to reduce to some extent the size of the resulting feature set, if large, while maintaining or improving the prediction performance of SDP models. A backtracking function, which is the third phase of the proposed AREMFFS method, is presented in the following sub-section.

3.3.3. Backtracking Function Phase

The last phase of the proposed AREMFFS method is a backtracking function that is applied to the resulting rank list from the ensemble rank aggregation phase. The backtracking function is based on checking the relevance of each feature in the resulting rank list against the initial rank list produced by the individual FFS method. That is, the rank or score of each feature in the resulting rank list is compared with the rank or score features in each of the individual rank lists produced by CS, IG, and REF FS methods. In particular, the backtracking function phase of the proposed AREMFFS method selects features that have a frequency value greater than or equal to $n - 1$, where n is the number of FFS methods used in the experiment. Iteratively, the number of FFS methods experimented on is 3 ($n = 3$); hence, for a feature to be selected in the backtracking phase into the optimal subset of features, the feature must be a $\log_2 N$ top-ranked feature of at least one $n - 1$ rank list generated by the CS, IG, and REF FS methods. That is, only features from the resulting rank list that are ranked important by at least two of the experimented FFS methods will be selected. This approach aims to confirm the relevancy of features in the optimal resulting rank list and appropriately reduce the size of the resulting feature set while maintaining or improving the prediction performance of SDP models.

3.4. Software Defect Datasets

Defect datasets from four publicly available repositories were used for the experiments in this study. Specifically, 25 datasets with varying granularities were selected from PROMISE, NASA, AEEEM, and ReLink repositories. For the NASA repository, the Shepperd, Song [67] version of defect datasets was used. The datasets consist of software features produced by static code metrics. Static code metrics are derived from the source code size and complexity [19,22]. The PROMISE repository contains defect datasets de-

rived from object-oriented metrics and additional information from software modules. This additional information is derived from Apache software [18,22,68]. Concerning the ReLink repository, datasets from this repository are derived from source code information from version control. These datasets were created by Wu, Zhang [69] as linkage data, and have been widely used in existing studies in SDP [62,70,71]. Lastly, the AEEEM datasets contain software features from source code metrics based on change metrics, entropy, and churn of source code metrics [19,22,68,72]. The description of these datasets is presented in Table 3.

Table 3. Description of software defect datasets.

Datasets	Number of Features	Number of Modules
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

3.5. Experimental Procedure

This section presents and analyses the experimental procedure followed in this study as shown in Figure 1.

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

- **Scenario A:** In this case, the performance of the proposed AREMFFS method was tested and compared with the baseline FS methods used in this study. The essence of this scenario was to evaluate and validate the performance of the AREMFFS against NoFS, CS, IG, and REF FS methods.

- **Scenario B:** In this case as well, the performance of the proposed AREMFFS method was tested and compared with existing rank aggregation-based multi-filter FS methods as proposed in [31,32]. Findings from this scenario were used to validate the effectiveness of the proposed AREMFFS against Min, Max, Mean, Range, GMean, and HMean rank aggregation-based FS methods.
- **Scenario C:** For this case, the performance of AREMFFS was tested and compared with its variant (REMFFS) as proposed in this study. This allowed a fair comparison and empirically validated the effectiveness of the proposed AREMFF method.

Experimental results and findings based on the aforementioned scenarios were used to answer the following research questions:

- RQ1: How effective is the proposed AREMFFS method compared to baseline FFS methods?
- RQ2: How effective is the proposed AREMFFS method compared to existing rank aggregation-based multi-filter FS methods?

SDP models generated based on the above-listed scenarios were trained and tested using the 10-fold cross-validation (CV) technique. The CV technique mitigates data variability issues that may occur in defect datasets. In addition, the CV technique has been known to produce models with low bias and variance [73–77]. The prediction performances of generated SDP models were assessed using performance evaluation metrics such as accuracy, AUC, and f-measure. The Scott–Knott ESD statistical rank test was used to ascertain the significant differences in the prediction performances of the models used in the experiment. The Weka machine learning library, R lang, and Origin Plot were used for the experimentation [78].

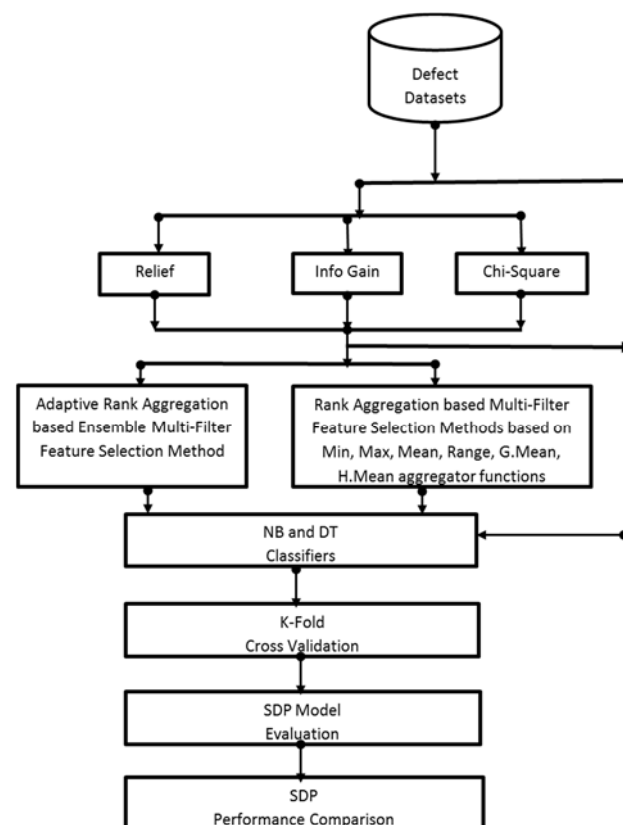


Figure 1. Experimental procedure.

3.6. Performance Evaluation Metrics

In terms of performance evaluation, SDP models based on the proposed and other methods were analysed 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 [6,79].

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

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \times 100\% \quad (1)$$

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:

$$\text{F - Measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2)$$

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:

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

$$\text{Precision} = \left(\frac{\text{TP}}{\text{TP} + \text{FP}} \right), \quad (4)$$

where TP is true positive (representing accurate prediction), FP is false positive (representing inaccurate prediction), TN is true negative (representing accurate mis-prediction), and FN is false negative (representing inaccurate mis-prediction).

4. Results and Discussion

This section presents and discusses experimental results based on the experimental process and procedure described in Section 3.5. Box plots are used to represent the prediction performances (accuracy, AUC, and f-measure values) of NB and DT models based on the various FS methods used. Also, Scott–KnottESD statistical rank tests are used to show significant differences in the prediction performance of developed models.

4.1. Experimental Results on Scenario A

In this section, experimental results based on Scenario A as defined in Section 3.5 are presented and discussed. Scenario A is based on assessing and comparing the prediction performances of NB and DT models based on proposed AREMFFS and baseline FS (CS, IG, REF, and NoFS) methods.

Figures 2–4 display box-plot representations of the accuracy, AUC, and f-measure values of NB and DT classifiers with the NoFS method, the baseline FFS (IG, REF, CS) method, and the proposed AREMFFS method. Specifically, the accuracy values of NB and DT classifiers compared with the FS (IG, CS, REF, and AREMFFS) and NoFS methods are presented in Figure 2. The results indicate that NB and DT had good accuracy values on the software defect dataset. Nonetheless, the increased deployment of baseline FFS methods (IG, CS, and REF) further enhanced the accuracy values of NB and DT classifiers. This can be seen in their respective average accuracy values as depicted in Figure 2. Particularly, NB and DT classifiers with the NoFS method had average accuracy values of 76% and 80.89%, respectively. Concerning baseline FFS methods, CS with NB and DT classifiers recorded average accuracy values of 78.93% and 81.97%, which indicated increments of +3.85% and +1.34%, respectively. Identical occurrences were realized in models with IG (NB: 78.51%, DT: 81.9%) and REF (NB: 78.99%, DT: 81.17%) FS methods, with increments of average accuracy values (IG: (+3.32%, +1.25%) and DT: (+3.93%, +0.3%)), respectively. However, models based on AREMFFS with NB and DT classifiers had superior average accuracy values over the NB and DT models with baseline FFS (NoFS, CS, IG, REF) methods. As presented in Table 4, concerning models based on the NB classifier, AREMFFS had increments of +7.46%, +3.47%, +4.02%, and +3.39% in the average accuracy values over models based

on the NoFS, CS, IG, and REF methods, respectively. Also, concerning models based on the DT classifier as shown in Table 5, AREMFFS had increments of +3%, +1.63%, +1.72%, and +2.64% in average accuracy values over models based on the NoFS, CS, IG, and REF methods, respectively. These results showed that, concerning accuracy values, models based on AREMFFS outperformed models based on baseline FSS (CS, IG, REF) methods. That is, AREMFFS had a superior positive impact on the prediction accuracy values of NB and DT models over the CS, IG, and REF FS methods.

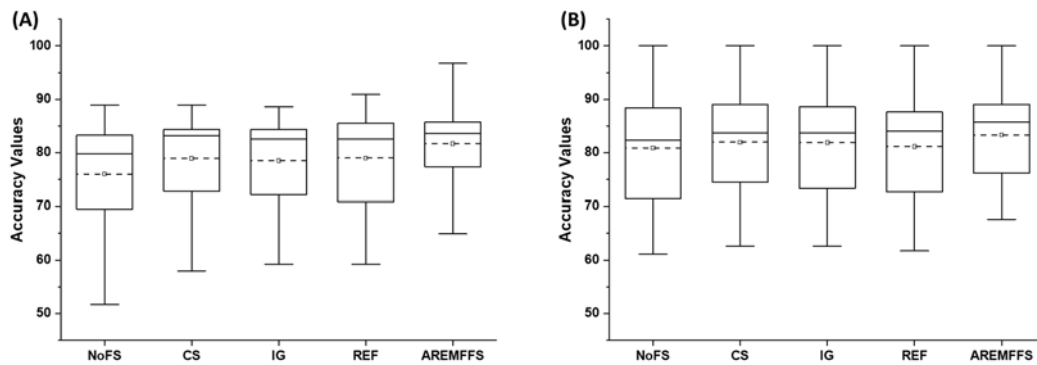


Figure 2. Box-plot representations (accuracy values) of models based on AREMFFS and baseline FFS methods: (A) average accuracy values of NB and (B) average accuracy values of DT.

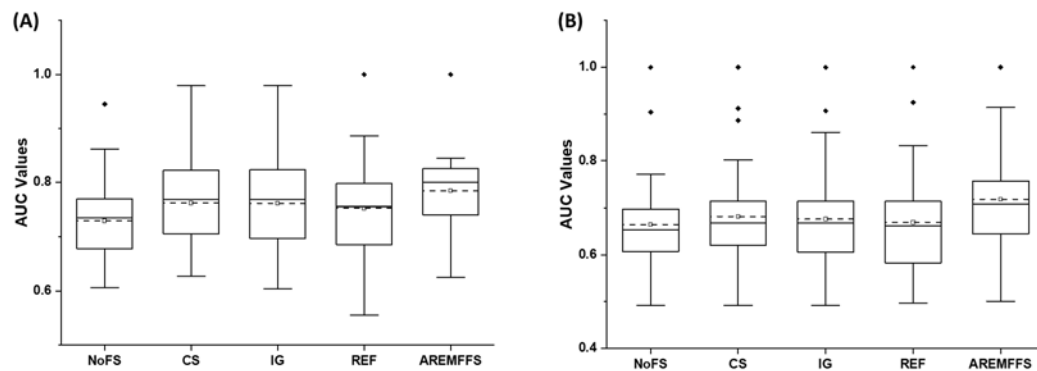


Figure 3. Box-plot representations (AUC values) of models based on AREMFFS and baseline FFS methods: (A) average accuracy values of NB and (B) average accuracy values of DT.

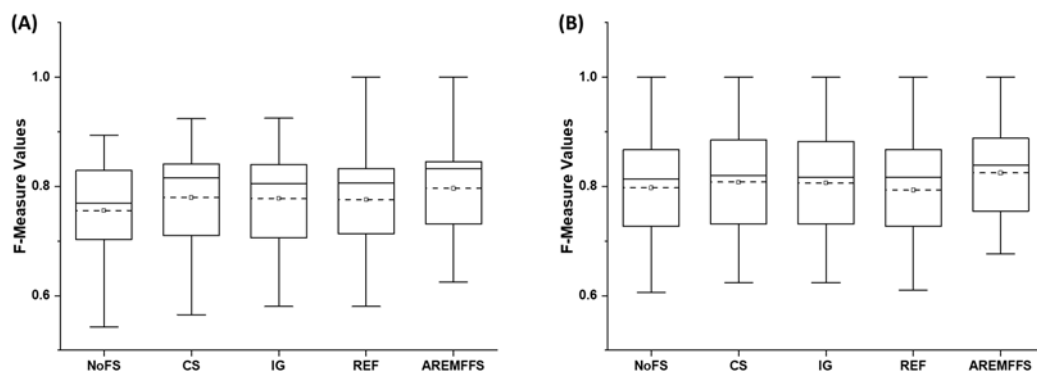


Figure 4. Box-plot representations (F-Measure values) of models based on AREMFFS and baseline FFS methods: (A) average accuracy values of NB and (B) average accuracy values of DT.

Table 4. Models based on the NB classifier with proposed AREMFFS and baseline FFS methods.

Models	Average Accuracy (%)	Average AUC	Average F-Measure
NoFS	76.00	0.730	0.756
CS	78.93	0.762	0.779
IG	78.51	0.761	0.778
REF	78.99	0.753	0.776
AREMFFS	81.67	0.784	0.797

Table 5. Models based on the DT classifier with proposed AREMFFS and baseline FFS methods.

Models	Average Accuracy (%)	Average AUC	Average F-Measure
NoFS	80.89	0.665	0.797
CS	81.97	0.682	0.808
IG	81.90	0.676	0.806
REF	81.17	0.669	0.793
AREMFFS	83.31	0.723	0.825

In terms of AUC values, Figure 3 displays box-plot representations of models based on NB and DT classifiers with baseline FFS and proposed AREMFFS methods. Similar to observations on accuracy values, NB and DT models based on baseline FFS (CS, IG, and REF) had superior AUC values when compared with NB and DT models with the NoFS method. Specifically, CS had increments of +4.4% and +2.55% in AUC values for models based on NB (0.762) and DT (0.682) over the NoFS (NB: 0.73, DT: 0.665) method. Correspondingly, NB and DT models based on IG had increments of +4.25% and +1.65% in average AUC values. Also, models based on REF recorded increments of +3.15% and +0.6% in average AUC values for NB and DT classifiers, respectively, when compared with models with the NoFS method. Nonetheless, similar to the observations on accuracy values, models based on AREMFFS had superior average AUC values over models with baseline FFS (NoFS, CS, IG, REF) methods. As shown in Table 4, concerning models based on the NB classifier, AREMFFS had increments of +7.4%, +2.88%, +3.02%, and +4.12% in average AUC values over models based on the NoFS, CS, IG, and REF methods, respectively. A similar case is observed with models based on the DT classifier as shown in Table 5. AREMFFS had increments of +8.72%, +6.01%, +6.95%, and +8.07% in average AUC values over models based on the NoFS, CS, IG, and REF methods, respectively.

Also, Figure 4 presents the f-measure values for NB and DT models based on the baseline FFS and proposed AREMFFS methods. Models based on the NB classifier with the CS (0.779), IG (0.778), and REF (0.776) methods recorded average f-measure value increases of +3.04%, +2.91%, and +2.64%, respectively, over the NB model with the NoFS method. As for the DT models, IG and CS recorded increments of +1.38% and +1.13% in average f-measure values, but DT models with the REF method performed poorly with a −0.5% decrease of the f-measure value. Summarily, it can be observed that models based on FS methods had better prediction performance than models with the NoFS method. However, models based on AREMFFS with NB and DT classifiers recorded better average f-measure values over NB and DT models with baseline FFS (CS, IG, REF) methods. From Table 4, the NB model based on AREMFFS had increments of +5.42%, +2.31%, +2.44%, and +2.71% in average f-measure values over models based on the NoFS, CS, IG, and REF methods, respectively. Also, the DT model with AREMFFS had increments of +3.51%, +2.1%, +2.36%, and +4.04% in average f-measure values over models based on the NoFS, CS, IG, and REF methods, respectively. These results further indicate the superiority of models (NB and DT) based on AREMFFS over models based on baseline FFS (NoFS, CS, IG, REF) methods.

Summarily, experimental results, as displayed in Figures 2–4, showed that the deployment of FS methods in SDP further enhances the prediction performances of SDP models. This finding is supported by observations in existing studies where FS methods are applied in SDP [19–22]. Nonetheless, it was also observed that the effect of FS methods varies and depends on the classifiers selected in this study. Also, there are no clear-cut differences in the performances of each of the FFS (CS, IG, and REF) methods, even though the selected FFS methods have different underlying computational characteristics. Thus, the selection of an appropriate FFS method to be used in SDP processes becomes a problem that can be termed a filter rank selection problem. This observation from the experimental results strengthens the aim of our study, which proposes a rank aggregation-based multi-filter FS method for SDP. As shown in Figures 2–4, the proposed AREMFFS method not only had a superior positive impact on NB and DT models, but also had a more positive impact than the individual CS, IG, and REF FFS methods. Particularly, Tables 4 and 5 present the prediction performances (average accuracy, average AUC, and average f-measure) of NB and DT models with the proposed AREMFFS methods and the experimented baseline FFS (NoFS, CS, IG, REF) methods, respectively.

Figures 5–7 present statistical rank tests of the models (NB and DT) tested based on accuracy, AUC, and f-measure values, respectively. Specifically, the Scott–KnottESD statistical rank test, a mean comparison approach that uses hierarchical clustering to separate mean values into statistically distinct clusters with non-negligible mean differences, was conducted [62,80] to show significant statistical differences in the mean values of methods and results used. As depicted in Figures 5–7, models with different colours show that there are statistically significant differences amongst their values; hence, they are grouped into a different category. Similarly, models with the same colour indicate that there are no statistically significant differences in their values.

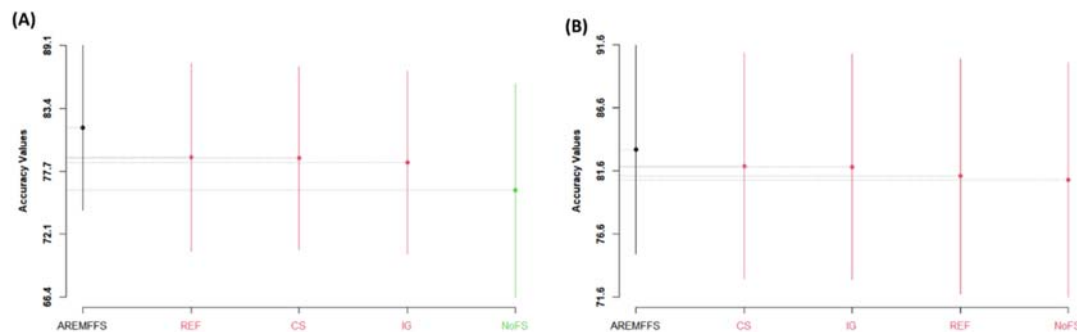


Figure 5. Scott–KnottESD rank test result of models based on AREMFFS and baseline FFS methods: (A) average accuracy values of NB and (B) average accuracy values of DT.

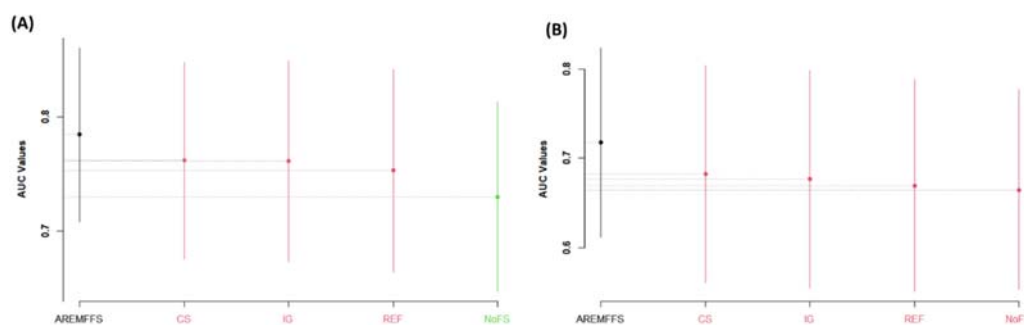


Figure 6. Scott–KnottESD rank test result of models based on AREMFFS and baseline FFS methods: (A) average AUC values of NB and (B) average AUC values of DT.

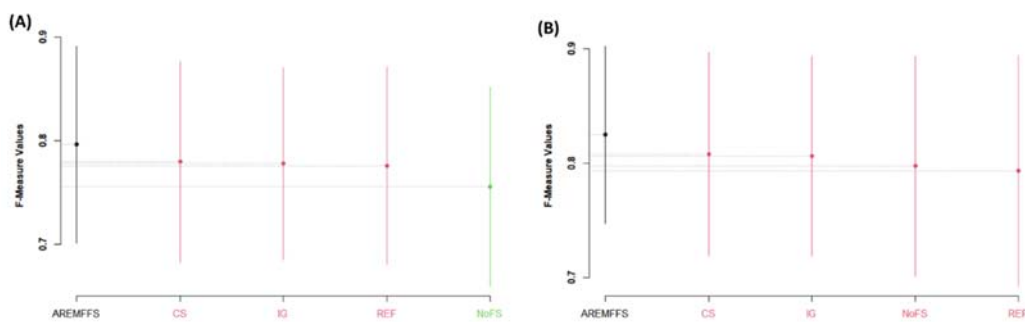


Figure 7. Scott–KnottESD rank test result of models based on AREMFFS and baseline FFS methods: (A) average F-Measure values of NB (B) average f-measure values of DT.

As presented in Figure 5, there are statistically significant differences in the average accuracy values of NB and DT models with the proposed AREMFFS method when compared with other FS methods. In particular, for NB models, AREMFFS ranks highest (first), followed by REF, CS, and IG, which are in the same category, while the NoFS method ranks last. In the case of DT models, AREMFFS still ranks highest followed by other FS methods (CS, IG, REF, and NoFS). It should be noted that the arrangements of models from the statistical rank test are vital as models that appear first (from left to right) are superior to the other models, irrespective of their category. This observation indicates that models based on AREMFFS have superior accuracy values over models based on CS, IG, REF, and NoFS methods. Also, similar observations were recorded from statistical rank tests based on AUC values. Figure 6 presents the Scott–KnottESD statistical rank tests based on AUC values, and there too models based on AREMFFS rank highest. In terms of NB models, AREMFFS ranks highest followed by CS, IG, and REF, which are in the same category, while the NoFS method ranks last. As for DT models, AREMFFS still ranks highest, followed by CS, IG, REF, and NoFS FS methods. Lastly, statistical rank tests based on f-measure values, as shown in Figure 7, followed the same pattern as that of accuracy and AUC values, with models based on AREMFFS being statistically superior to the other experimented baseline FS methods. A summary of the Scott–KnottESD statistical rank tests of the proposed AREMFFS and baseline FS methods with NB and DT classifiers is presented in Table 6.

In summary, from the experimental and statistical test results, the proposed AREMFFS method recorded a superior positive impact on the prediction performances of SDP models (NB and DT) in comparison with individual FSS (CS, IG, REF, and NoFS) methods on the defect datasets that were studied.

Table 6. Summary of the Scott–Knott rank test of proposed AREMFFS and baseline FS methods.

Statistical Rank	Average Accuracy		Average AUC		Average F-Measure	
	NB	DT	NB	DT	NB	DT
1	AREMFFS	AREMFFS	AREMFFS	AREMFFS	AREMFFS	AREMFFS
2	REF, CS, IG	CS, IG, REF, NoFS	CS, IG, REF	CS, IG, REF, NoFS	CS, IG, REF	CS, IG, REF, NoFS
3	NoFS	-	NoFS	-	NoFS	-

4.2. Experimental Results on Scenario B

This section presents and discusses experimental results based on Scenario B (see Section 3.5). Scenario B is defined by evaluating and comparing the prediction performances of NB and DT models based on the proposed AREMFFS method and the existing (Min, Max, Mean, Range, GMean, HMean) rank aggregation-based multi-filter FS methods.

Figures 8–10 show box-plot representations of the accuracy, AUC, and f-measure values of NB and DT classifiers with proposed AREMFFS and existing rank aggregation-

based multi-filter FS methods. In particular, Figure 8 presents the accuracy values of NB and DT models with AREMFFS and existing rank aggregation-based multi-filter FS methods. It can be observed that models based on AREMFFS (NB: 81.67%, DT: 83.31%) had superior average accuracy values compared to existing Min (NB: 79.72%, DT: 82.60%), Max (NB: 79.88%, DT: 82.34%), Mean (NB: 79.36%, DT: 82.53%), Range (NB: 77.99%, DT: 81.87%), GMean (NB: 79.48%, DT: 82.70%), and HMean (NB: 79.66%, DT: 82.61%) rank aggregation-based multi-filter FS methods. Specifically, based on NB models, AREMFFS had increments of +2.91%, +2.45%, +2.24%, +4.72%, +2.76%, and +2.52% in average accuracy value over the existing Mean, Min, Max, Range, GMean, and HMean rank aggregation-based multi-filter FS methods. Likewise for DT models, AREMFFS had increments of +0.95%, +0.86%, +1.18%, +1.76%, +0.74%, and +0.85% in average accuracy value over the existing Min, Max, Mean, Range, GMean, and HMean rank aggregation-based multi-filter FS methods. As observed, the experimental results indicate that models based on AREMFFS outperformed models based on existing rank aggregation-based multi-filter FS methods on accuracy values. In other words, AREMFFS had a superior positive impact on the prediction accuracy values of NB and DT models over the Min, Max, Mean, Range, GMean, and HMean rank aggregation-based multi-filter FS methods.

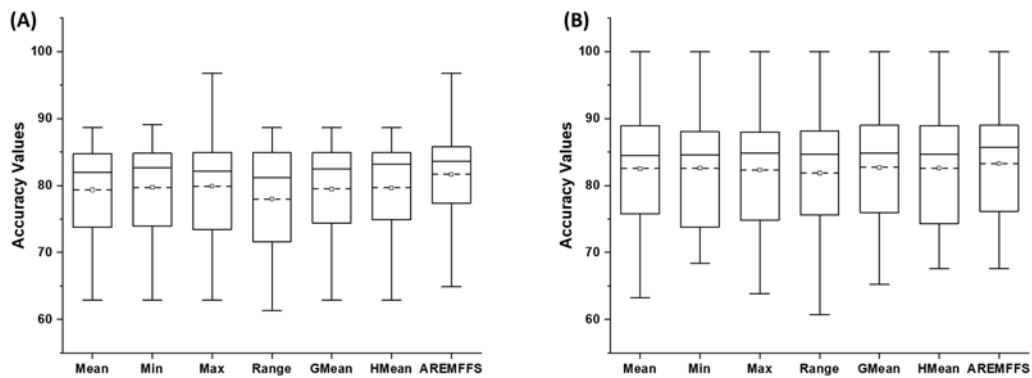


Figure 8. Box-plot representations (accuracy values) of models based on AREMFFS and existing rank aggregation-based multi-filter FS methods: (A) average accuracy values of NB and (B) average accuracy values of DT.

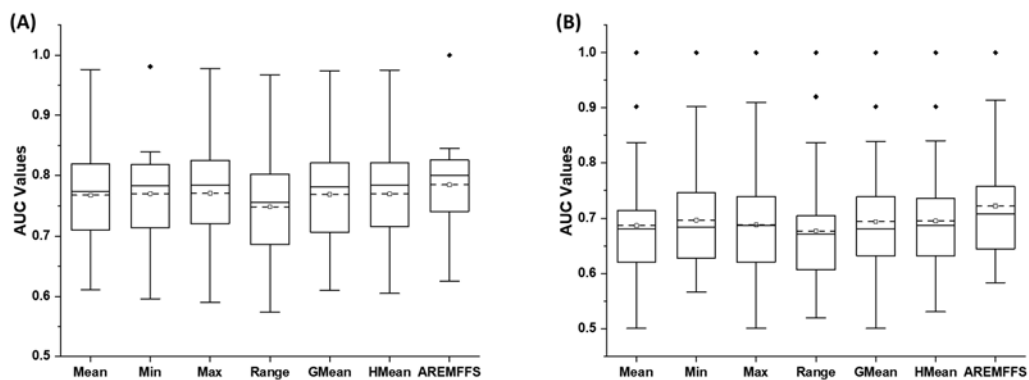


Figure 9. Box-plot representations (AUC values) of models based on AREMFFS and existing rank aggregation-based multi-filter FS methods: (A) average accuracy values of NB and (B) average accuracy values of DT.

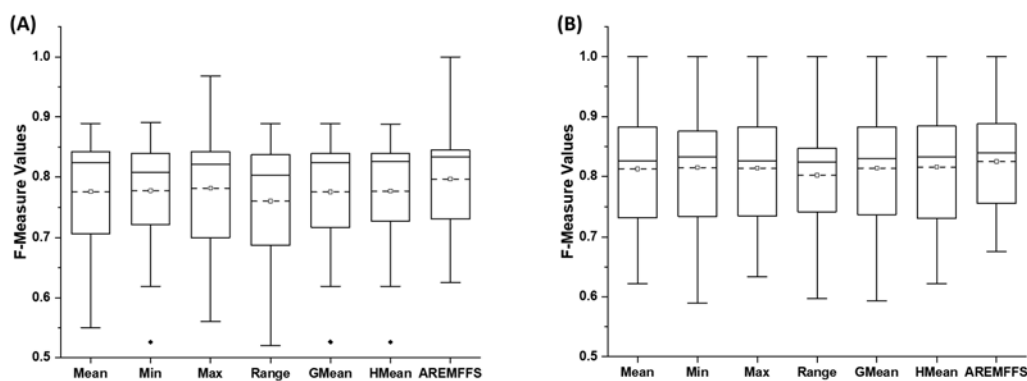


Figure 10. Box-plot representations (f-measure values) of models based on AREMFFS and existing rank aggregation-based multi-filter FS methods: (A) average accuracy values of NB and (B) average accuracy values of DT.

Concerning AUC values, Figure 9 presents box-plot representations of models based on NB and DT classifiers with the proposed AREMFFS and existing rank aggregation based multi-filter FS methods. Similar to observations on accuracy values, models based on AREMFFS (NB: 0.784, DT: 0.723) had superior average AUC values over models with existing Min (NB: 0.769 DT: 0.697), Max (NB: 0.770, DT: 0.688), Mean (NB: 0.767, DT: 0.687), Range (NB: 0.748, DT: 0.677), GMean (NB: 0.768, DT: 0.694), and HMean (NB: 0.769, DT: 0.696) rank aggregation-based multi-filter FS methods. As presented in Table 7, concerning models based on NB classifier, AREMFFS had increments of +2.22%, +1.95%, +1.82%, +4.81, +2.08%, and +1.95% in average AUC values over models based on Mean, Min, Max, Range, GMean, and HMean rank aggregation-based multi-filter FS methods, respectively. A similar case is observed with models based on the DT classifier as depicted in Table 8. AREMFFS had increments of +5.24%, +3.73%, +5.09%, +6.79%, +4.18%, and +3.88% in average AUC values over models based on Mean, Min, Max, Range, GMean, and HMean rank aggregation-based multi-filter FS methods, respectively. As observed, the experimental results indicated that models based on AREMFFS outperformed models based on existing rank aggregation-based multi-filter FS methods on accuracy values. In other words, AREMFFS had a superior positive impact on the prediction accuracy values of NB and DT models over Min, Max, Mean, Range, GMean, and HMean rank aggregation-based multi-filter FS methods.

Table 7. Models based on the NB classifier with proposed AREMFFS and existing rank aggregation-based multi-filter FS methods.

Models	Average Accuracy (%)	Average AUC	Average F-Measure
Mean	79.36	0.767	0.775
Min	79.72	0.769	0.777
Max	79.88	0.770	0.781
Range	77.99	0.748	0.759
GMean	79.48	0.768	0.775
HMean	79.66	0.769	0.776
AREMFFS	81.67	0.784	0.797

Table 8. Models based on the DT classifier with proposed AREMFFS and existing rank aggregation-based multi-filter FS methods.

Models	Average Accuracy (%)	Average AUC	Average F-Measure
Mean	82.53	0.687	0.813
Min	82.60	0.697	0.815
Max	82.34	0.688	0.814
Range	81.87	0.677	0.802
GMean	82.70	0.694	0.814
HMean	82.61	0.696	0.815
AREMFFS	83.31	0.723	0.825

Furthermore, concerning f-measure values, Figure 10 presents box-plot representations of models with NB and DT classifiers with proposed AREMFFS and existing rank aggregation based multi-filter FS methods. Models based on AREMFFS (NB: 0.797, DT: 0.825) had superior average f-measure values over models with existing Min (NB: 0.777, DT: 0.815), Max (NB: 0.781, DT: 0.814), Mean (NB: 0.775, DT: 0.813), Range (NB: 0.789, DT: 0.802), GMean (NB: 0.775, DT: 0.814), and HMean (NB: 0.776, DT: 0.815) rank aggregation-based multi-filter FS methods. Specifically, based on the NB classifier, AREMFFS had increments of +2.84%, +2.57%, +2.05%, +5.00, +2.84%, and +2.71% in average AUC values over models based on Mean, Min, Max, Range, GMean, and HMean rank aggregation-based multi-filter FS methods, respectively. Also, based on the DT classifier, AREMFFS had increments of +1.48%, +1.23%, +1.35%, +2.87%, +1.35%, and +1.23% in average AUC values over models based on Mean, Min, Max, Range, GMean, and HMean rank aggregation-based multi-filter FS methods, respectively. As observed, the experimental results indicated that models based on AREMFFS outperformed models based on existing rank aggregation-based multi-filter FS methods on f-measure values. In other words, AREMFFS had a superior positive impact on the f-measure values of NB and DT models over Min, Max, Mean, Range, GMean, and HMean rank aggregation-based multi-filter FS methods.

In summary, the findings from the experimental results, as shown in Figures 8–10, indicate the superiority of the proposed AREMFFS over existing rank aggregation-based multi-filter FS methods. That is, NB and DT models based on AREMFFS outperformed NB and DT models based on existing Mean, Min, Max, Range, GMean, and HMean rank aggregation-based multi-filter FS methods. The superior performance of the proposed AREMFFS can be attributed to a combination of the robust strategy it deploys for aggregating multiple rank lists based on majority voting, and its backtracking ability which further removes irrelevant features from the generated optimal feature list.

For further analyses, the performance of the proposed AREMFFS and the existing experimented rank aggregation-based multi-filter FS methods were subjected to Scott–KnottESD statistical rank tests to determine the statistically significant differences in their respective performances. Figures 11–13 present statistical rank tests of the proposed AREMFFS and existing rank aggregation-based multi-filter FS methods on NB and DT classifiers based on accuracy, AUC, and f-measure values, respectively. As depicted in Figure 11A, it can be observed that there are statistically significant differences in the average accuracy values of NB models using the proposed AREMFFS when compared with NB models based on existing rank aggregation-based multi-filter FS methods.

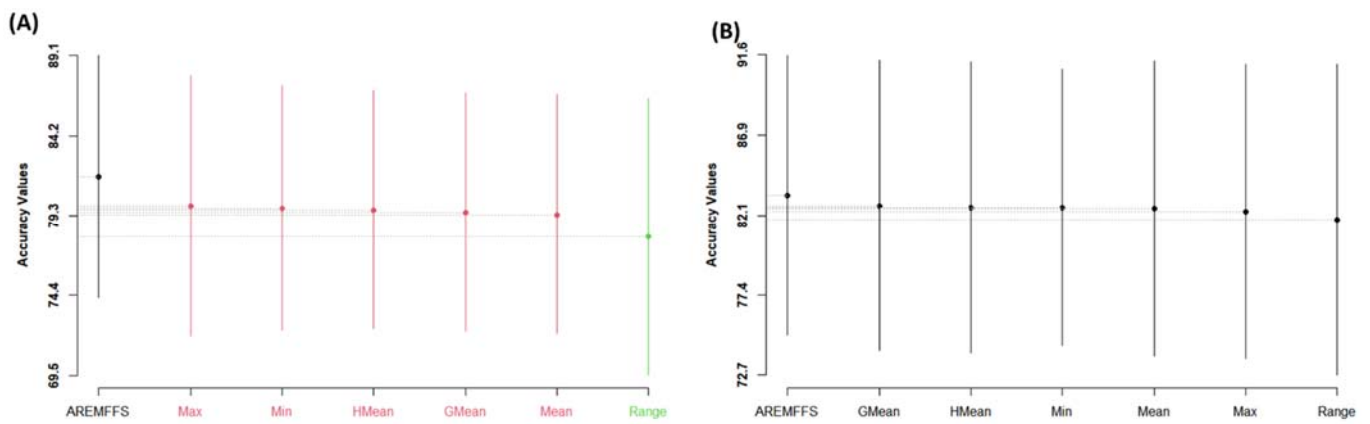


Figure 11. Scott-KnottESD rank test result of models based on AREMFFS and existing rank aggregation-based multi-filter FS methods: (A) average accuracy values of NB and (B) average accuracy values of DT.

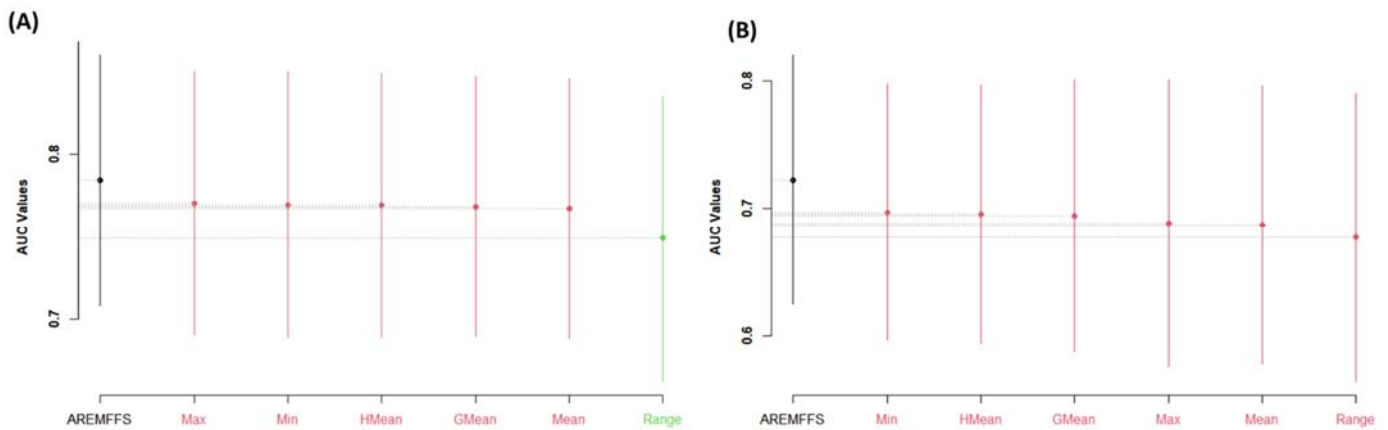


Figure 12. Scott-KnottESD rank test result of models based on AREMFFS and existing rank aggregation-based multi-filter FS methods: (A) average AUC values of NB and (B) average AUC values of DT.

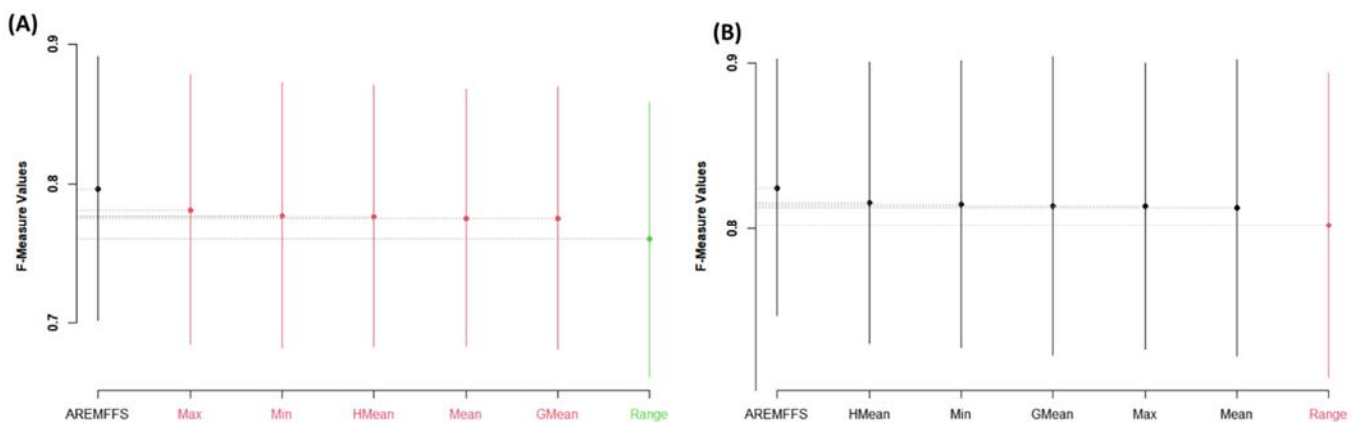


Figure 13. Scott-KnottESD Rank test result of models based on AREMFFS and existing rank aggregation-based multi-filter FS methods. (A) average f-measure values of NB and (B) average f-measure values of DT.

Specifically, AREMFFS ranks highest (first), followed by Max, Min, HMean, GMean, and Mean, which are in the same category, while the Range aggregation method ranks last. In the case of DT models as presented in Figure 11B, although there is no significant statistical difference in the prediction accuracy values, AREMFFS still ranks highest, followed by the GMean, HMean, Min, Mean, Max, and Range aggregation methods. In this case, the order of superiority/arrangements (left to right) of the models from the statistical

test was considered and AREMFFS appeared first. These findings indicate that models based on AREMFFS have superior accuracy values over models based on Min, Max, Mean, Range, GMean, and HMean rank aggregation-based multi-filter FS methods. Also, similar observations were recorded from statistical rank tests based on AUC values. Figure 12 presents the Scott–KnottESD statistical rank tests based on AUC values, and models based on AREMFFS rank highest. In terms of NB models, AREMFFS ranks highest followed by Max, Min, HMean, GMean, and Mean, which are in the same category, while the Range rank aggregation method ranks last. A similar situation can be observed in the case of DT models as AREMFFS ranks highest followed by the Min, HMean, GMean, Max, Mean, and Range rank aggregation methods. In addition, statistical rank tests based on f-measure values, as presented in Figure 13, indicated similar findings to those of the accuracy and AUC values with Min, Max, Mean, Range, GMean, and HMean rank aggregation-based multi-filter FS methods. Table 9 summarizes and presents the Scott–KnottESD statistical rank tests of proposed AREMFFS and existing rank aggregation-based multi-filter FS methods with NB and DT classifiers.

In summary, based on the experimental and statistical test results, the proposed AREMFFS method recorded a superior positive impact on the prediction performances of SDP models (NB and DT) over existing rank aggregation-based multi-filter FS (Min, Max, Mean, Range, GMean, HMean) methods on the defect datasets studied.

Table 9. Summary of Scott–KnottESD statistical rank tests of proposed AREMFFS and existing rank aggregation-based multi-filter FS methods.

Statistical Rank	Average Accuracy		Average AUC		Average F-Measure	
	NB	DT	NB	DT	NB	DT
1	AREMFFS	AREMFFS, GMean, HMean, Min, Mean, Max, Range	AREMFFS	AREMFFS	AREMFFS	AREMFFS, HMean, Min, GMean, Max, Mean,
2	Max, Min, HMean, GMean, Mean	-	Max, Min, HMean, GMean, Mean	Min, HMean, GMean, Max, Mean, Range	Max, Min, HMean, Mean, GMean	Range
3	Range	-	Range	-	Range	-

4.3. Experimental Results on Scenario C

In this section, experimental results based on Scenario C (See Section 3.5) are presented and discussed. Scenario C is based on assessing and comparing the prediction performances of NB and DT models based on the proposed AREMFFS and its variant (REMFFS: rank aggregation-based ensemble multi-filter feature selection) as proposed in this study. The REMFFS method is based on the same working principle as AREMFFS but without the backtracking function included. The results of this analysis will allow for a fair comparison between the two and empirically validate the effectiveness of the proposed AREMFFS method.

Figures 14–16 present box-plot representations of the accuracy, AUC, and f-measure values of NB and DT classifiers with the proposed AREMFFS and REMFFS methods. Correspondingly, Figure 14 presents the accuracy values of NB and DT models with the AREMFFS and REMFFS methods. It can be observed that models based on AREMFFS (NB: 81.67%, DT: 83.31%) had superior average accuracy values when compared with the REMFFS (NB: 80.62%, DT: 82.75%) methods. In particular, based on NB and DT models, AREMFFS had increments of +1.3% and +0.67% in average accuracy values, respectively, over the REMFFS method. As observed, the experimental results indicated that models based on AREMFFS outperformed models based on REMFFS on accuracy values. That is, AREMFFS had a superior positive impact on the prediction accuracy values of NB and DT models over the REMFFS method.

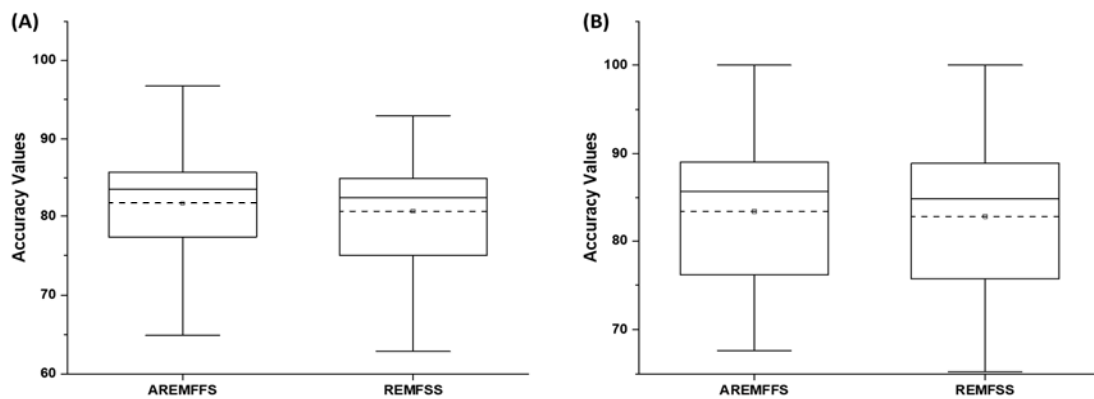


Figure 14. Box-plot representations (accuracy values) of models based on AREMFFS and REMFFS: (A) average accuracy values of NB and (B) average accuracy values of DT.

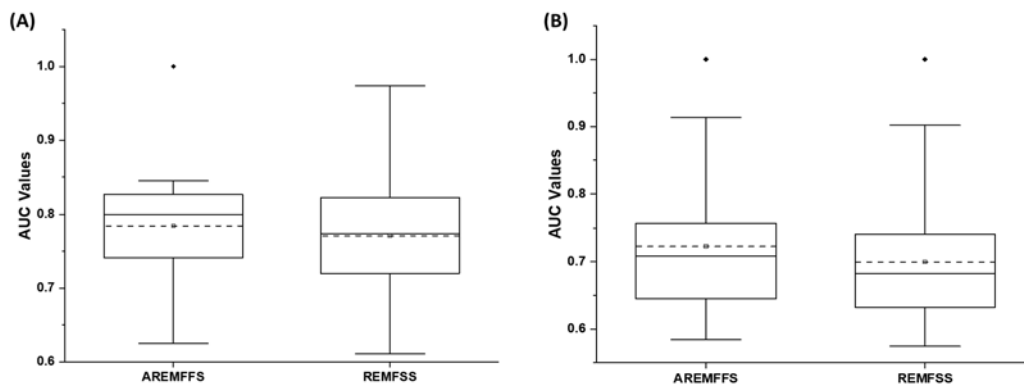


Figure 15. Box-plot representations (AUC values) of models based on AREMFFS and REMFFS: (A) average accuracy values of NB and (B) average accuracy values of DT.

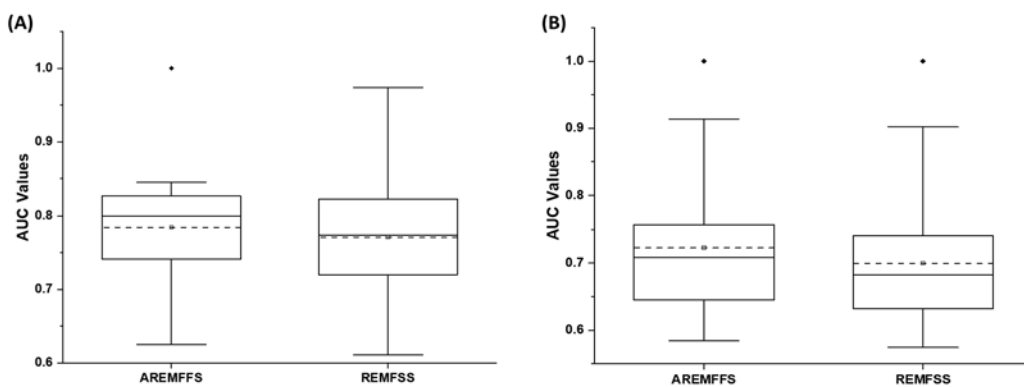


Figure 16. Box-plot representations (f-measure values) of models based on AREMFFS and REMFFS: (A) average accuracy values of NB and (B) average accuracy values of DT.

Regarding AUC values, Figure 15 shows box-plot representations of models based on NB and DT classifiers with the proposed AREMFFS and REMFFS methods. As presented in Tables 10 and 11, models based on AREMFFS (NB: 0.784, DT: 0.723) had superior average AUC values over models with REMFFS (NB: 0.771, DT: 0.699). Specifically, NB and DT models with AREMFFS had increments of +1.69% and +3.43% in average AUC values, respectively, over models based on the REMFFS method. As observed, the experimental results indicated that models based on AREMFFS outperformed models based on REMFFS. In other words, AREMFFS had a superior positive impact on the AUC values of NB and DT models over the REMFFS method. Also, in terms of f-measure values, Figure 16 presents

box-plot representations of models with NB and DT classifiers with proposed AREMFFS and REMFFS methods. Models based on AREMFFS (NB: 0.797, DT: 0.825) had superior average f-measure values over models with the REMFFS (NB: 0.778, DT: 0.813) method. In particular, NB and DT models with AREMFFS had increments of +2.44% and +1.48% in average f-measure values, respectively, over models based on the REMFFS method. Similarly, the experimental results showed the superiority of models based on the proposed AREMFFS as it outperformed models based on REMFFS on f-measure values. That is, AREMFFS had a superior positive impact on the f-measure values of NB and DT models over the REMFFS method.

Table 10. Models based on the NB classifier with the proposed AREMFFS and REMFFS methods.

Models	Average Accuracy (%)	Average AUC	Average F-Measure
REMFFS	80.62	0.771	0.778
AREMFFS	81.67	0.784	0.797

Table 11. Models based on the DT classifier with the proposed AREMFFS and REMFFS methods.

Models	Average Accuracy (%)	Average AUC	Average F-Measure
REMFFS	82.75	0.699	0.813
AREMFFS	83.31	0.723	0.825

Based on the preceding experimental results as presented in Figures 14–16, the superiority of the proposed AREMFFS over REMFFS can be observed. Correspondingly, the observed superior performance of the proposed AREMFFS over REMFFS can be attributed to its backtracking ability to further remove irrelevant features from the generated optimal feature list. As established, the removal of irrelevant features will further improve the performance of the proposed AREMFFS method. However, it should be noted that REMFFS, which is a variant of AREMFFS, generated a good and competitive prediction performance. Figures 17–19 further analysed the performance of AREMFFS and REMFFS methods statistically. In other words, the Scott–KnottESD statistical rank test was used to determine the statistically significant differences in their respective performances based on accuracy, AUC, and f-measure values, respectively.

As presented in Figure 17, it can be observed that there are no statistically significant differences in the average accuracy values of NB and DT models with the proposed AREMFFS and REMFFS methods. Nonetheless, AREMFFS still ranks higher than the REMFFS method when the order of superiority/arrangements (left to right) of the models from the statistical test were considered; that is, AREMFFS appeared first.

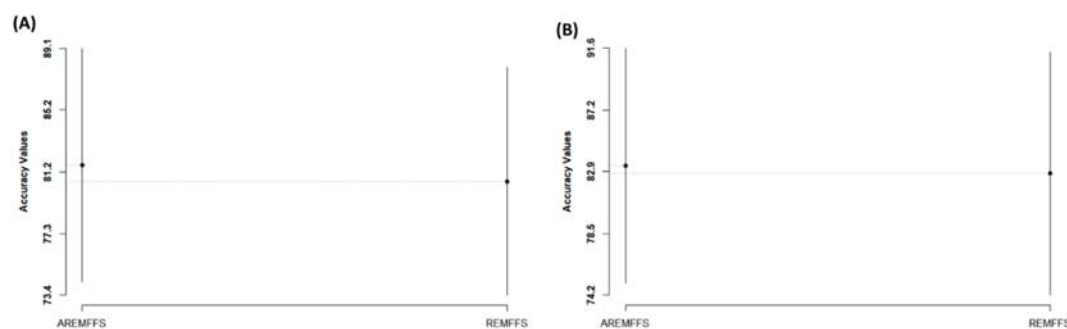


Figure 17. Scott–KnottESD rank test result of models based on AREMFFS and REMFFS: (A) average accuracy values of NB and (B) average accuracy values of DT.

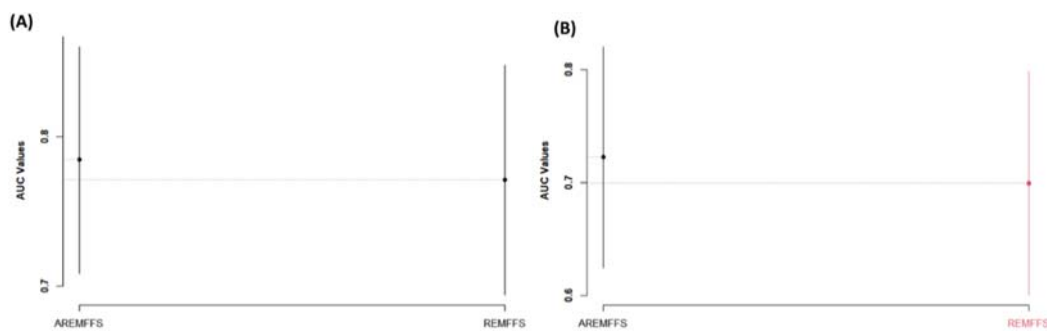


Figure 18. Scott–KnottESD rank test result of models based on AREMFFS and REMFFS: (A) average AUC values of NB and (B) average AUC values of DT.

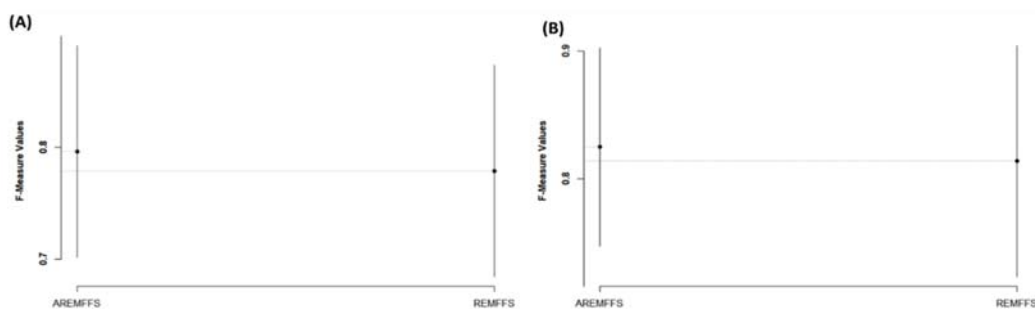


Figure 19. Scott–KnottESD rank test result of models based on AREMFFS and REMFFS: (A) average f-measure values of NB and (B) average f-measure values of DT.

From Figure 19, similar observations were recorded from statistical rank tests based on f-measure values, as there is no statistically significant difference in the average f-measure values of models based on the two methods (AREMFFS and REMFFS), although AREMFFS is better. The case is slightly different for Scott–KnottESD statistical rank tests based on AUC values as shown in Figure 18. In terms of NB models (Figure 18A), AREMFFS and REMFFS fall into the same grouping. That is, the difference between their respective AUC values is statistically insignificant. However, for DT models (Figure 18B), AREMFFS is statistically superior to REMFFS, as there is a significant difference in their AUC values. In addition, a summary of the statistical test analyses on the performance of AREMFFS and REMFFS methods is presented in Table 12. These findings indicate that models based on AREMFFS have superior performance over models based on the REMFFS method.

Based on the experimental and statistical test results, the proposed AREMFFS method recorded a superior positive impact on the prediction performances of SDP models (NB and DT) over its variant (REMFFS method) for the defect datasets that were studied.

Table 12. Summary of the Scott–Knott rank test of the experimented FS methods on NB and DT models.

Statistical Rank	Average Accuracy		Average AUC		Average F-Measure	
	NB	DT	NB	DT	NB	DT
1	AREMFFS, REMFFS	AREMFFS, REMFFS	AREMFFS, REMFFS	AREMFFS	AREMFFS, REMFFS	AREMFFS, REMFFS
2	-	-	-	REMFFS	-	-

In summary, from the experimental results and statistical test analyses, the proposed AREMFFS method had a superior positive effect on the prediction performances of SDP models (NB and DT) compared to the individual filter FS methods (CS, IG, REF, and NoFS), existing rank aggregation based multi-filter FS methods (Min, Max, Mean, Range, GMean, and HMean), and its variant (REMFFS) on the defect datasets studied. These findings,

therefore, answer RQ1 and RQ2 (see Section 3.6) as presented in Table 13. Furthermore, the efficacy of AREMFFS solves the filter rank selection problem in SDP by integrating the power of individual filter FS methods. As a result, combining filter (multi-filter) methods is suggested as a viable choice for harnessing the power of the respective FFS and the strengths of filter–filter relationships in selecting germane features for FS methods as conducted in this report

Table 13. Answers to research questions.

Research Questions	Answers
RQ1. How effective is the proposed AREMFFS method compared to baseline FFS methods?	The proposed AREMFFS outperformed individual FFS methods with statistically significant differences.
RQ2. How effective is the proposed AREMFFS method compared to existing rank aggregation-based multi-filter FS methods?	The proposed AREMFFS outperformed existing rank aggregation-based multi-filter FS methods with statistically significant differences.

5. Conclusions

This study focuses on resolving high dimensionality and filter rank selection problems in software defect prediction by proposing a novel AREMFFS method. Selecting an applicable and pertinent filter FS method to be used in SDP is often a problem as the efficacy of these filter FS methods varies. As such, AREMFFS is proposed and designed to combine multiple rank lists generated by different filter FS methods into a single robust rank list. Moreover, a geometric mean function and a backtracking function are used to automatically select top-ranked features and reduce the number of features on the aggregated list. For performance evaluation and validation, NB and DT models were developed using features generated by AREMFFS, selected baseline filter FS methods (IG, CS, REF, and NoFS methods), existing rank aggregation based multi-filter FS methods (Min, Max, Mean, Range, GMean, and HMean), and variants of AREMFFS (REMFFS) for defect datasets with different granularity. Findings from the experimental results indicated the efficacy and superiority of the proposed AREMFFS method as it recorded a better positive impact on the prediction performances of NB and DT classifiers than the other experimented FS methods in most cases.

In particular, the proposed ARMFFS was able to generate a more stable and complete subset of features that best represented the datasets studied. These findings, therefore, support the combination and aggregation of rank lists from multiple filter FS methods as a feasible solution to primarily high dimensionality and filter rank selection problems in SDP. In a wider sense, the results and findings from this research can be used by experts and researchers in SDP and other applicable research domains that require FS methods to address high dimensionality and filter rank selection problems.

Furthermore, in this study, there is a trade-off of the computational time for prediction performance. That is, the computational time is not used as an assessment metric. This is because in most cases, ensemble methods have been reported to record more computational time than single methods [81]. Moreover, the overhead cost of a software defect misprediction could have dire consequences.

As it is a limitation of this study, we plan to investigate and broaden the scope of this study in the future by analysing other ensemble configurations of the FS method with more prediction models. Application of EC techniques for selection of features and reduction in computational time of ensemble methods will be explored. Furthermore, the effect of threshold values on FFS efficacies is worth examining, as the appropriate threshold value is dependent on the dataset used.

Author Contributions: Conceptualization, A.O.B. and S.B.; methodology, A.O.B. and L.F.C.; software, A.O.B., G.K. and V.E.A.; validation, A.O.B., S.B., M.A.A. and A.A.I.; formal analysis, A.O.B., S.M., M.A.A., V.E.A. and A.A.I.; investigation, A.O.B., S.B. and L.F.C.; resources, S.B., S.M., M.A.A. and A.O.B.; data curation, A.A.I. and A.O.B.; writing—original draft preparation, A.O.B.; writing—review and editing, S.B., L.F.C., V.E.A., A.A.I. and A.O.B.; visualization, S.M., G.K., V.E.A. and A.O.B.; supervision, S.B. and L.F.C.; project administration, S.B., S.M. and L.F.C.; funding acquisition, S.B. and S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research paper was fully supported by Universiti Teknologi PETRONAS, under the Yayasan Universiti Teknologi PETRONAS (YUTP) Fundamental Research Grant Scheme (YUTP-FRG/015LC0240) and (YUTP-FRG/015LC0297).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Akintola, A.G.; Balogun, A.O.; Lafenwa-Balogun, F.; 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](#)]
2. Balogun, A.O.; Lafenwa-Balogun, F.B.; Mojeed, H.A.; Adeyemo, V.E.; Akande, O.N.; Akintola, A.G.; Bajeh, A.O.; Usman-Hamza, F.E. SMOTE-Based Homogeneous Ensemble Methods for Software Defect Prediction. In Proceedings of the International Conference on Computational Science and Its Applications, online, 1–4 July 2020; Springer: Heidelberg, Germany, 2020; pp. 615–630.
3. Bajeh, A.O.; Oluwatosin, O.J.; Basri, S.; Akintola, A.G.; Balogun, A.O. Object-oriented measures as testability indicators: An empirical study. *J. Eng. Sci. Technol.* **2020**, *15*, 1092–1108.
4. 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](#)]
5. Chauhan, A.; Kumar, R. Bug severity classification using semantic feature with convolution neural network. In *Computing in Engineering and Technology*; Springer: Heidelberg, Germany, 2020; pp. 327–335.
6. Jimoh, R.; Balogun, A.; Bajeh, A.; Ajayi, S. A PROMETHEE based evaluation of software defect predictors. *J. Comput. Sci. Its Appl.* **2018**, *25*, 106–119.
7. 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](#)]
8. 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; IEEE: Piscataway, NJ, USA, 2011; pp. 207–214.
9. 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.
10. 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](#)]
11. Li, N.; Shepperd, M.; Guo, Y. A systematic review of unsupervised learning techniques for software defect prediction. *Inf. Softw. Technol.* **2020**, *122*, 106287. [[CrossRef](#)]
12. Okutan, A.; Yıldız, O.T. Software defect prediction using Bayesian networks. *Empir. Softw. Eng.* **2014**, *19*, 154–181. [[CrossRef](#)]
13. 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; ACM: New York, NY, USA, 2014.
14. 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. Its Appl.* **2019**, *26*, 73–88.
15. 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.
16. 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; IEEE: Piscataway, NJ, USA, 2007.
17. 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](#)]

18. 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; ACM: New York, USA, 2014.
19. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Hashim, A.S. A hybrid multi-filter wrapper feature selection method for software defect predictors. *Int. J. Supply Chain Manag.* **2019**, *8*, 916–922.
20. 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](#)]
21. 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](#)]
22. 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 Proceedings of the International Conference on Emerging Applications and Technologies for Industry 4.0, Uyo, Akwa Ibom, Nigeria, 21–23 July 2020; Springer: Heidelberg, Germany, 2020; pp. 39–52.
23. 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, Zurich, Switzerland, 2–3 January 2015; AIRCC Press: Chennai, Tamil Nadu, India, 2015; pp. 71–79.
24. Anbu, M.; Mala, G.A. Feature selection using firefly algorithm in software defect prediction. *Clust. Comput.* **2019**, *22*, 10925–10934. [[CrossRef](#)]
25. 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; IEEE: Piscataway, NJ, USA, 2016.
26. Cai, J.; Luo, J.; Wang, S.; Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing* **2018**, *300*, 70–79. [[CrossRef](#)]
27. Li, Y.; Li, T.; Liu, H. Recent advances in feature selection and its applications. *Knowl. Inf. Syst.* **2017**, *53*, 551–577. [[CrossRef](#)]
28. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [[CrossRef](#)]
29. Iqbal, A.; Aftab, S. A Classification Framework for Software Defect Prediction Using Multi-filter Feature Selection Technique and MLP. *Int. J. Mod. Educ. Comput. Sci.* **2020**, *12*. [[CrossRef](#)]
30. Osanaiye, O.; Cai, H.; Choo, K.-K.R.; Dehghantanha, A.; Xu, Z.; Dlodlo, M. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP J. Wirel. Commun. Netw.* **2016**, *2016*, 130. [[CrossRef](#)]
31. 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, 8–9 December 2020; Springer: Heidelberg, Germany, 2020; pp. 371–383.
32. 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](#)]
33. Smidts, C.; Stutzke, M.; Stoddard, R.W. Software reliability modeling: An approach to early reliability prediction. *IEEE Trans. Reliab.* **1998**, *47*, 268–278. [[CrossRef](#)]
34. Cortellessa, V.; Singh, H.; Cukic, B. Early reliability assessment of UML based software models. In Proceedings of the 3rd International Workshop on Software and Performance, Rome, Italy, 24–26 July 2002.
35. Gaffney, J.; Pietrolewicz, J. An automated model for software early error prediction (SWEEP). In Proceedings of the 13th Minnow Brook Workshop on Software Reliability, Blue Mountain Lake, NY, USA, 24–27 July 1990.
36. Gaffney, J.; Davis, C.F. An approach to estimating software errors and availability. In Proceedings of the 11th Minnow Brook workshop on Software Reliability, Blue Mountain Lake, NY, USA, 26–29 July 1988.
37. Al-Jamimi, H.A. Toward comprehensible software defect prediction models using fuzzy logic. In Proceedings of the 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 26–28 August 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 127–130.
38. Yadav, H.B.; Yadav, D.K. A fuzzy logic based approach for phase-wise software defects prediction using software metrics. *Inf. Softw. Technol.* **2015**, *63*, 44–57. [[CrossRef](#)]
39. Borgwardt, S.; Distel, F.; Peñalosa, R. The limits of decidability in fuzzy description logics with general concept inclusions. *Artif. Intell.* **2015**, *218*, 23–55. [[CrossRef](#)]
40. Khan, B.; Naseem, R.; Shah, M.A.; Wakil, K.; Khan, A.; Uddin, M.I.; Mahmoud, M. Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques. *J. Healthc. Eng.* **2021**, *2021*. [[CrossRef](#)] [[PubMed](#)]
41. Naseem, R.; Khan, B.; Ahmad, A.; Almogren, A.; Jabeen, S.; Hayat, B.; Shah, M.A. Investigating tree family machine learning techniques for a predictive system to unveil software defects. *Complexity* **2020**, *2020*, 6688075. [[CrossRef](#)]
42. Akimova, E.N.; Bersenev, A.Y.; Deikov, A.A.; Kobylkin, K.S.; Konygin, A.V.; Mezentsev, I.P.; Misilov, V.E. A Survey on Software Defect Prediction Using Deep Learning. *Mathematics* **2021**, *9*, 1180. [[CrossRef](#)]
43. Haouari, A.T.; Souici-Meslati, L.; Atil, F.; Meslati, D. Empirical comparison and evaluation of Artificial Immune Systems in inter-release software fault prediction. *Appl. Soft. Comput.* **2020**, *96*, 106686. [[CrossRef](#)]

44. Khurma, R.A.; Alsawalqah, H.; Aljarah, I.; Elaziz, M.A.; Damaševičius, R. An Enhanced Evolutionary Software Defect Prediction Method Using Island Moth Flame Optimization. *Mathematics* **2021**, *9*, 1722. [[CrossRef](#)]
45. Xu, Z.; Li, L.; Yan, M.; Liu, J.; Luo, X.; Grundy, J.; Zhang, Y.; Zhang, X. A comprehensive comparative study of clustering-based unsupervised defect prediction models. *J. Syst. Softw.* **2021**, *172*, 110862. [[CrossRef](#)]
46. Marjuni, A.; Adji, T.B.; Ferdiana, R. Unsupervised software defect prediction using signed Laplacian-based spectral classifier. *Soft Comput.* **2019**, *23*, 13679–13690. [[CrossRef](#)]
47. Balogun, A.O.; Akande, N.O.; Usman-Hamza, F.E.; Adeyemo, V.E.; Mabayoje, M.A.; Ameen, A.O. Rotation Forest-Based Logistic Model Tree for Website Phishing Detection. In Proceedings of the International Conference on Computational Science and Its Applications, Cagliari, Italy, 5–8 July 2021; Springer: Heidelberg, Germany, 2021; pp. 154–169.
48. Yao, J.; Shepperd, M. The impact of using biased performance metrics on software defect prediction research. *Inf. Softw. Technol.* **2021**, *139*, 106664. [[CrossRef](#)]
49. Kotte, A.; Qyser, D.; Moiz, A.A. A Survey of different machine learning models for software defect testing. *Eur. J. Mol. Clin. Med.* **2021**, *7*, 3256–3268.
50. Clarke, E.M., Jr.; Grumberg, O.; Kroening, D.; Peled, D.; Veith, H. *Model Checking*; MIT Press: Cambridge, MA, USA, 2018.
51. Imtiaz, N.; Murphy, B.; Williams, L. How do developers act on static analysis alerts? an empirical study of coverage usage. In Proceedings of the 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE), Berlin, Germany, 28–31 October 2019; IEEE: Piscataway, NJ, USA, 2019.
52. Cynthia, S.T.; Rasul, M.G.; Ripon, S. Effect of feature selection in software fault detection. In Proceedings of the International Conference on Multi-disciplinary Trends in Artificial Intelligence, Kuala Lumpur, Malaysia, 17–19 November 2019; Springer: Heidelberg, Germany, 2019; pp. 52–63.
53. 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 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, Argentina, 20–21 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 146–157.
54. 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 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, ON, Canada, 23–27 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 309–320.
55. Jia, L. A hybrid feature selection method for software defect prediction. In Proceedings of the IOP Conference Series: Materials Science and Engineering, Nanjing, China, 7–19 August 2018; IOP Publishing: Temple way, Bristol, UK, 2018.
56. Jacquier, E.; Kane, A.; Marcus, A.J. Geometric or arithmetic mean: A reconsideration. *Financ. Anal. J.* **2003**, *59*, 46–53. [[CrossRef](#)]
57. Wang, H.; Khoshgoftaar, T.M.; Napolitano, A. A comparative study of ensemble feature selection techniques for software defect prediction. In Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications, Washington, DC, USA, 12–14 December 2010; IEEE: Piscataway, NJ, USA, 2010.
58. Xia, Y.; Yan, G.; Jiang, X.; Yang, Y. A new metrics selection method for software defect prediction. In Proceedings of the 2014 IEEE International Conference on Progress in Informatics and Computing, Shanghai, China, 16–18 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 433–436.
59. Malik, M.R.; Yining, L.; Shaikh, S. The Role of Attribute Ranker using classification for Software Defect-Prone Data-sets Model: An Empirical Comparative Study. In Proceedings of the 2020 IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 24–20 September 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–8.
60. 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](#)]
61. Stiglic, G.; Kocbek, S.; Pernek, I.; Kokol, P. Comprehensive decision tree models in bioinformatics. *PLoS ONE* **2012**, *7*, e33812.
62. 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](#)]
63. Azhagusundari, B.; Thanamani, A.S. Feature selection based on information gain. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* **2013**, *2*, 18–21.
64. Bahassine, S.; Madani, A.; Al-Sarem, M.; Kissi, M. Feature selection using an improved Chi-square for Arabic text classification. *J. King Saud Univ.-Comput. Inf. Sci.* **2020**, *32*, 225–231. [[CrossRef](#)]
65. Urbanowicz, R.J.; Meeker, M.; La Cava, W.; Olson, R.S.; Moore, J.H. Relief-based feature selection: Introduction and review. *J. Biomed. Inform.* **2018**, *85*, 189–203. [[CrossRef](#)]
66. Oladepo, A.G.; Bajeh, A.O.; Balogun, A.O.; Mojeed, H.A.; Salman, A.A.; Bako, A.I. Heterogeneous Ensemble with Combined Dimensionality Reduction for Social Spam Detection. *Int. J. Interact. Mob. Technol.* **2021**, *15*, 84–103. [[CrossRef](#)]
67. 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](#)]
68. 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](#)]
69. 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; ACM: New York, USA, 2011; pp. 15–25.

70. Song, Q.; Guo, Y.; Shepperd, M. A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction. *IEEE Trans. Softw. Eng.* **2019**, *14*, 1253–1269. [[CrossRef](#)]
71. Nam, J.; Nam, J.; Fu, W.; Kim, S.; Menzies, T.; Tan, L. Heterogeneous defect prediction. *IEEE Trans. Softw. Eng.* **2017**, *44*, 874–896. [[CrossRef](#)]
72. 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; ACM: Bangalore, India, 2015.
73. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer: Heidelberg, Germany, 2013; Volume 112.
74. Kuhn, M.; Johnson, K. *Applied Predictive Modeling*; Springer: Heidelberg, Germany, 2013; Volume 26.
75. Alsariera, Y.A.; Adeyemo, V.E.; Balogun, A.O.; Alazzawi, A.K. Ai meta-learners and extra-trees algorithm for the detection of phishing websites. *IEEE Access* **2020**, *8*, 142532–142542. [[CrossRef](#)]
76. Alsariera, Y.A.; Elijah, A.V.; Balogun, A.O. Phishing Website Detection: Forest by Penalizing Attributes Algorithm and Its Enhanced Variations. *Arab. J. Sci. Eng.* **2020**, *45*, 10459–10470. [[CrossRef](#)]
77. 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. Improving the phishing website detection using empirical analysis of Function Tree and its variants. *Heliyon* **2021**, *7*, e07437. [[CrossRef](#)]
78. 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](#)]
79. Balogun, A.O.; Bajeh, A.O.; Orié, V.A.; Yusuf-Asaju, W.A. Software defect prediction using ensemble learning: An ANP based evaluation method. *FUOYE J. Eng. Technol.* **2018**, *3*, 50–55. [[CrossRef](#)]
80. Tantithamthavorn, C.; McIntosh, S.; Hassan, A.E.; Matsumoto, K. Comments on “Researcher bias: The use of machine learning in software defect prediction”. *IEEE Trans. Softw. Eng.* **2016**, *42*, 1092–1094. [[CrossRef](#)]
81. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev.* **2018**, *8*, e1249. [[CrossRef](#)]