Journal of Engineering Science and Technology
Vol. 18, No. 1 (2023) 187 - 209
© School of Engineering, Taylor's University

EMPIRICAL ANALYSIS OF THRESHOLD VALUES FOR RANK-BASED FILTER FEATURE SELECTION METHODS IN SOFTWARE DEFECT PREDICTION

MALEK ALMOMANI[1], ABDULLATEEF o. BALOGUN[2,3,*],
SHUIB BASRI[2], ABDULLAHI A. IMAM[4], AMMAR K. ALAZZAWI[5],
VICTOR E. ADEYEMO6, GANESH KUMAR[2]

[1]Department of Software Engineering and Information Systems,
The World Islamic Sciences and Education University, Amman, Jordan
[2]Department of Computer and Information Sciences, Universiti Teknologi PETRONAS,
Bandar Seri Iskandar, 32610, Perak, Malaysia
[3]Department of Computer Science, University of Ilorin, 240003, Ilorin, Nigeria
[4]School of Digital Science, Jalan Tungku Link, Gadong, Universiti Brunei Darussalam,
Brunei Darussalam, BE1410
[5]Computer Techniques Engineering, Al-Mustaqbal University College, Babylon, Iraq
[6]School of Built Environment, Engineering and Computing, Leeds Beckett University,
Headingley Campus, Leeds LS6 3QS, United Kingdom
*Corresponding Author: abdullateef_16005851@utp.edu.my

Abstract

Many studies have been conducted to explore the influence of feature selection (FS) techniques on software defect prediction (SDP) models, with conflicting empirical results and research outcomes. These reported contradictions may be due to relative research limitations, such as types of FS techniques or the size of defect datasets. In the instance of FS methods, it was discovered that selecting a suitable threshold value for picking top-ranked features in FS methods might be a cause of discrepancies in reported findings on SDP. Investigating and assessing the impacts of threshold values for the rank-based filter (RBF) FS techniques, as done in this work, becomes critical. 4 RBF (Chi-square, Correlation, Information Gain, and Relief) methods with 5 thresholds (No FS, log2N, Top20%, Top 30%, and Top 50%) values were investigated with 2 prediction models (Naïve Bayes (NB) and Decision Tree (DT)) on 25 software defects datasets. The experimented RBF techniques were selected based on distinct computational features to assure heterogeneity, as well as their performance in the current SDP research. Developed SDP models were evaluated using accuracy and area under the curve (AUC) values while the Scott-KnottESD rank statistical test technique was employed to rank experimented RBF methods with applied threshold values. According to the experimental results, selecting the Top20% of top-ranked features in RBF methods had a greater (positive) impact on the prediction performances of SDP models than other applied threshold values. Furthermore, the outcomes of this study corroborate previous research on the capacity of FS techniques to improve the prediction efficacies of SDP models. Consequently, we urge that FS methods be utilized in SDP tasks. In the case of RBF methods, the Top20% threshold value should be used since it outperforms de-factor log2N and other threshold values. Moreover, findings from this study can be a guide to subsequent SDP studies and further strengthen the tenacity of experimental findings and conclusions in SDP studies.

Keywords: Feature selection, Rank-based filter, Software defect prediction.

1. Introduction

Software defect prediction (SDP) is the usage of machine learning (ML) techniques for the identification of defective modules or components in a software system before software release. SDP takes place before the software release and is based on the elemental characteristics of software systems. These fundamental properties of a software system are measurable features of software systems that can be investigated and evaluated to determine the degree of software systems' reliability and quality [1-3]. In other words, these features such as software coupling, cohesion, and complexity are processed by ML techniques for knowledge derivation. The derived knowledge can be used for software process improvements. The goal of software engineers is to build high-quality and reliable software using available resources [4-7]. Hence, SDP is pivotal in the software development life cycle (SDLC).

Nowadays, developed software systems are inherently voluminous and convoluted with multiple modules or components that are interrelated. Moreover, these software systems regularly undergo scheduled updates and upgrades for additional features as specified in software requirements or end-users' requests. Hence, ascertaining the fundamental properties of such software systems often engage numerous software metrics with distinct assessments [8-10]. Consequently, this result in a high-dimensionality problem as the amount of generated software features is customarily enormous and somewhat redundant [7, 11, 12]. Some studies in SDP have investigated and reported that the unsatisfactory prediction performances of SDP models can be attributed to the presence of redundant and irrelevant software metrics. Besides, high-dimensional data often incur high computational complexity and memory wastage. Hence, the selection of the relevant set of software metrics is of utmost importance [13-16].

Feature selection (FS) is a data pre-processing task usually deployed to resolve the high dimensionality problem in ML tasks. FS targets and selects a subset of features from a dataset that most depict the full features from the dataset without compromising the essence of the dataset [17-19]. FS augments the efficacy of SDP models by improving the quality of the software metric datasets. Based on these reasons, several research studies have proposed and assessed the efficacy and impact of various FS methods on the prediction performances of SDP models [11, 13-15, 20-25].

Therefore, it is imperative to regularly study and analyse the effectiveness of these FS methods. Some current literature has examined the effect of FS techniques on the prediction performances of SDP models with differing findings. For instance, some studies stated that some specific FS methods such as rank-based filter FS methods are better than other FS methods like wrapper FS methods while some studies remain resolute that there is no significant difference in the efficiency of FS methods in SDP [13-15, 21, 24-27]. We observed that one of such factors that may lead to these contradictions can be the threshold value for rank-based filter FS methods. As gathered in existing SDP studies, $\log_2 N$ (N = number of features) has been used as the de-factor threshold values for rank-based filter FS methods [13-15, 25, 28]. However, in FS-based studies from other research domains such as bioinformatics, $\log_2 N$ isn't the only threshold value used in selecting features in rank-based filter FS methods. Notably, some of the existing studies have recorded better efficacy of rank-based filter FS methods with other threshold values such as Full features, Top20%, Top30% and Top50% [29, 30]. Therefore, it is imperative to investigate the effect of varying threshold values in rank-based filter FS methods. Findings from this study will guide researchers on the selection of threshold values for rank-based filter FS methods in SDP. Also, the outcome of this study will further support the scientific grit of experimental findings and observations in existing FS-based empirical studies.

Consequently, an empirical study of 4 rank-based filter FS methods with 4 distinct threshold values was conducted with 2 classifiers to examine the effect of diverse threshold values on rank-based filter FS methods on the prediction performances of SDP models. The deployed rank-based filter FS techniques are chosen based on consistency in efficacy and popularity as used in current SDP literature.

Summarily, the following are the major contributions of this study:

- An empirical analysis of the impact and effectiveness of various threshold values on 2 classifiers over 25 datasets in SDP on selected rank-based filter FS methods. The strength of this research is an objective study of this magnitude.
- This study empirically validates the suitability and efficiency of 4 distinct threshold values for rank-based filter FS methods.

This research paper is structured and divided into six sections. Section 2 reviews related current SDP literature. Section 3 discusses applied FS methods, selected classifiers, defect datasets, and performance evaluation metrics. The developed experimental framework and procedures used in conducting experiments are presented in Section 4 while results from experiments are presented and discussed in Section 5. Conclusions and future research works are given in Section 6.

2.Literature Review

From existing studies on SDP, it has been reported that high dimensionality, a data quality problem, dampens the efficacy and efficiency of SDP models' predictive performance. Feature selection techniques are implemented to pick only the necessary and irredundant software features for the SDP phase as a practical solution to the dimensionality drawback in SDP. Several refereed scientific research studies have identified the effectiveness of FS techniques on SDP models. Contrasting views and research findings have been gathered from these studies. This may be due to the choice and scope of FS methods investigated by these studies. Notably, some studies have investigated the problem of inconsistencies in the effectiveness of FS methods in SDP.

For instance, Xu et al. [15] performed study on performance impact analysis of 32 FS techniques in SDP. Their study aimed at addressing the limitations in existing studies SDP studies but was limited to the noise level and type of defect dataset used in existing studies. 14 rank-based filter FS methods were investigated. However, the threshold value of $\log_2 N$ was used for their RBF methods.

In another study, Balogun et al. [13] investigated and analysed the effectiveness of 18 FS methods on 5 defect datasets to overcome certain inconsistencies in current literature. Similarly, the relatively limited scope of their study focused only on search methods absent in current literature. $\log_2 N$ was utilized as the threshold value in selecting optimal features for respective RBF methods. From the results of both studies, it was observed that there is no concrete difference (statistically) in the effectiveness of the studied FS methods. Besides, Balogun et al. [13] claimed that FFR methods produced more stable performance accuracy values.

Shivaji et al. [31] conducted an empirical study on 6 FS methods on 11 SDP datasets with support vector machine (SVM) and NB as base classifiers. 4 rank-based filter methods were used in their study. Afzal and Torkar [11] also compared 8 FS methods on 5 defect datasets. 2 rank-based filter methods were investigated in their study. From their experimental results, both studies concluded that FS techniques are integral and useful for SDP as it enhances the prediction efficacies of examined classifiers. However, we noted different threshold values for RBF methods that were used. Afzal and Torkar [11] used 1/3(N), 1/5(N) and 1/2 while Shivaji et al. [31] utilized $\log_2 N$ as their threshold values.

Also, Rathore and Gupta [25] investigated the effectiveness of 15 FS methods with different underlining computational properties. Based on their experimental findings, they observed that both principal component analysis (PCA) and information gain (IG) outperform other examined FS techniques. Similarly, Ghotra et al. [14] performed an extensive empirical analysis which consists of 11 RBF methods and $\log_2 N$ was their threshold value. They concluded that the RBF method outperforms other applied methods. However, this finding may be different if another threshold value for RBF methods were used as suggested by Afzal and Torkar [11].

Based on the aforementioned reviews, it is observed and evident that prediction performances of SDP models can be improved by applying FS techniques. As such, many FS method has been proposed and deployed to improve the prediction performances of SDP models since the effect of mispredictions of software defects can be detrimental [4, 32]. Nonetheless, in some of these studies, as presented in Table 1, conflicting experimental findings and test results have been found. It was found that these inconsistencies typically come from the study scope of these studies, which are relative in most instances.

It was observed that the threshold values used for RBF methods vary amongst existing studies. This variation could be the main reason for the inconsistencies in existing studies as the effect of RBF methods depends on the threshold value used for selecting optimal features for the SDP process.

Another observation is the variation in size and granularity of software defect datasets used in existing studies. For instance, Ghotra et al. [14] and Xu et al. [15] conducted their experiments on different granularity and number of software defects datasets (18 and 16 respectively). Less amount of software defect datasets was experimented with [13, 33, 34]. These defect datasets are from diverse repositories and have disparate properties. Studies like Gao et al.[35] experimented on private datasets. Moreover, research outcomes from experimental procedures with a limited number of datasets may not be generalizable since defect datasets have disparate characteristics and granularity [15]. This can also be seen as one of the sources of the discrepancies and contradictions described in current studies.

Therefore, to empirically validate the understanding of FS techniques in SDP, it is imperative to discuss the constraints of current studies generally. This research carries out an empirical research study on the effects of FS techniques on SDP. Specifically, this study investigates the effect of threshold values on rank-based filter FS techniques on software defect datasets with disparate characteristics and granularity. Findings from this study will further substantiate the impact of the FS method on SDP models and validate the applicable threshold values for rank-based filter FS techniques. To the best of our knowledge, no study had considered addressing these limitations in SDP studies.

Table 1. Analysis of current literature on the effect of rank-based filter methods on SDP models.

Current Literature	Feature Selection Methods	Filter Threshold Value	Datasets
	Rank-Based Filter Methods		
Ghotra et al. [14]	11 filter Rank Methods	log2N	18 datasets (NASA and PROMISE)
Xu et al. [15]	14 filter Rank Methods	log2N	16 Datasets (NASA and AEEEM)
Shivaji et al. [31]	4 filter Rank Methods	1/2(N)	11 software projects
Muthukumaran et al. [24]	7 filter Rank Methods	1/3(N), 1/2(N),1/5(N)	16 Datasets (NASA and AEEEM)
Gao et al. [35]	7 filter Rank Methods	log2N	Private Dataset
Wang et al. [27]	6 filter Rank Methods	log2N	3 datasets
Khoshgoftaar et al. [36]	7 filter Rank Methods	log2N	16 datasets
Rathore and Gupta [25]	7 filter Rank Methods	log2N	14 datasets (PROMISE)
Afzal and Torkar [11]	2 Filter Rank Method	1/3(N), 1/2(N),1/5(N)	5 datasets
Balogun et al. [13]	4 filter Rank Methods	log2N	5 datasets (NASA)

3. Methodology

This sub-section highlights FS methods, classification techniques, software defect datasets, and the evaluation metrics deployed in this research. FS methods were chosen from current and similar research for a broad empirical analysis [13-15, 25].

3.1. Rank-based filter (RBF) FS method

Rank-based filter (RBF) FS Methods consider and use characteristics derived from datasets to assess and rank features of datasets. Rank and weight scores are produced based on the underlining computational functionalities of RBF methods. The performance of RBF methods is independent of the influence of classifier(s) to be used for categorizing a dataset. Features are chosen based on their rank and weight scores [13, 15]. In this study, 4 RBF methods (See Table 2) with distinct computational characteristics and the Ranker search method for subset selection are used. Specifically, Chi-Square Filter (CSF), Correlation Filter (COF), Information Gain Filter (IGF) and Relief Filter (REF) were selected in this study. The preference for the selected RBF methods is due to their reported recurrent usage and performance in existing FS-related SDP studies [13-15, 25, 35, 37].

3.1.1. Chi-square filter (CSF)

The Chi-square Filter (CSF) technique is feature selection proposed based on a statistical basis technique and it is used to indicate the level of independence score by checking feature independence in the class label. Feature with a high obtained score, have a higher dependency relationship between the feature and the class label. Scientifically is represented as:

$$\mathrm{X}^2(\mathrm{c_i}) = \sum \frac{(O_i - E_i)^2}{E_i} \quad (1)$$

where: *C* = degree of freedom, *O* = observed value(s), *E* = Expected value(s)

3.1.2. Correlation filter (COF)

The correlation filter (COF) method is another statistics-based FS method that selected features based on similarity measures between features. That is, COF uses the association between the continuous features and the class feature for its FS process. Specifically, if any given two features are linearly independent, then the correlation co-efficient score *(r)* is ±1 or 0 otherwise. The value of *r* is generated as:

$$r = \frac{\sum(X_i - \bar{X}_i)(Y_i - \bar{Y}_i)}{\sqrt{\sum(X_i - \bar{X}_i)^2}\sqrt{\sum(Y_i - \bar{Y}_i)^2}} \quad (2)$$

3.1.3. Information gain filter (IGF)

The information Gain filter (IGF) technique considers one of the most popular filter techniques used to select the appropriate features in the case of unknown features, by lowering the uncertainties attributed to detecting the class label with reference to the mechanism of information theory. The information theory support and assist to select top features before initiating the educating process [38, 39]. The concept of IG is based on entropy while aiming at decreasing the level of entropy, starting from the root node to the leaf nodes. The entropy of an instance X can be determined as:

$$H(X) = \sum_i -P_{x_i} \; log_2(P_{x_i}) \quad (3)$$

where P_{x_i} represents the prior probabilities of *X*.

The entropy of X indicated a new instance Y is represented as:

$$H(X|Y) = -\sum_i P_{y_j} \; \sum_i P_{(x_i|y_j)} \, log_2 \, P_{(x_i|y_j)} \quad (4)$$

The entropy is indicated as the level by which the entropy of X decreases to present further information regarding X as given by Y, and is described as:

$$IG(X|Y) = H(X) - H(X|Y) \quad (5)$$

where H = Entropy, Y = dependent variable, and X = independent variable

3.1.4. Relief filter (REF)

ReliefF filter (REF) technique deploys sampling procedure on a particular dataset and then detects the closest neighbors from the same and alternative classes. The features of the sampled instances are compared with those of their vicinity, and then a score of relevance is assigned to each feature. REF is an instance-based FS technique could be utilized to noisy and unfinish datasets. It shows dependencies between features with minimal bias[38, 39].

Table 2. Rank-based filter FS methods.

Rank-based Filter FS Methods	Computational Characteristics	References
Chi-Square Filter (CSF)	Statistic-based Filter method	[13-15, 25, 35]
Correlation Filter (COF)	Statistic-based Filter method	[13-15, 25, 35]
Information Gain Filter (IGF)	Probability-based Filter method	[13-15, 25, 35]
Relief Filter (REF)	Instance-based Filter method	[14, 15, 24]

3.2. Classification techniques

In this study, the duo of Naïve Bayes (NB) and Decision Tree (DT) classifiers are deployed to evaluate the effectiveness and influence of the studied FS techniques. NB and DT classifiers are independent of the FS techniques and have been extensively used in SDP experiments and studies. Also, NB and DT have been reported to have good prediction performance, adequately handle the class imbalance problem, and produce stable classification models [40-43]. Table 3 presents NB and DT classifiers with respective parameter settings as used in this study.

Table 3. Classification techniques.

Classification Techniques	Computational Characteristics	Parameter Setting

NB	A probability-based classifier.	NumDecimalPlaces = 2;
		UseKernelEstimator = True

DT	An information entropy-based classifier.	Confidence factor = 0.25;
		MinNumObj = 2

3.3. Software defect datasets

The datasets utilized in this research were collected from publicly accessible software repositories such as NASA, PROMISE, ReLink and AEEEM. The choice and selection of defect datasets from these repositories are to ensure the usage of defect datasets with diverse defect granularity. In this research, Shepperd et al. [44] version of the NASA repository defect datasets is used. The NASA datasets are from static code metrics culled at the function level [14, 15]. PROMISE repository defect datasets are generated from object-oriented metrics at the class level. Specifically, PROMISE defect datasets are derived from java-based Apache software [14, 23, 25]. Developed by Wu et al. [45], ReLink dataset uses details from a program such as version control and it is commonly used in SDP experiments [46-48]. The AEEEM datasets are quite different as it comprises software features gotten from source code metrics [14, 15, 23, 24]. A detailed overview of studied defect datasets is provided in Table 4.

Table 4. Software defect datasets.

Datasets	Repository	Granularity	References
EQ JDT ML PDE	AEEEM	Class Level	[14, 15, 23, 24, 49]
CM1 KC1 KC2 KC3 MW1 PC1 PC3 PC4 PC5	NASA	Function Level	[13-15, 34, 44]
ANT CAMEL JEDIT REDKITOR TOMCAT VELOCITY XALAN	PROMISE	Class Level	[14, 34, 44]
SAFE ZXING APACHE ECLIPSE SWT	ReLink	File Level	[45-48]

3.4. Performance evaluation metrics

Accuracy and the Area Under the Curve (AUC) were adopted to measure the prediction efficiency of SDP models concerning the impact of the proposed RBF methods as performance assessment metrics. These metrics are commonly used and chosen from current SDP literature [13, 14, 25].

i. Accuracy is the rate of instances correctly predicted by the total number of instances.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \qquad (6)$$

ii. The Area under Curve (AUC) illustrates the trade-off between TP and FP. Its value varies between 0 and 1 and it provides an aggregate output metric across all possible thresholds for classification. High AUC values show superior predictive performance.

where *TP* = Correct Prediction, *FP* = Incorrect Prediction, *TN* = Correct Misprediction, and *FN* = Incorrect Misprediction.

The Scott-Knott Effect Size Difference (Scott-KnottESD) (α=0.05) and the Double Scott-KnottESD rank statistical test are also utilized to further determine the importance of the effect of the RBF FS threshold methods on the baseline classifiers (NB and DT). The Scott-KnottESD statistical rank test is a method of mean comparison that uses a hierarchical method of clustering to divide mean values into statistically meaningful partitions of non-negligible differences. [48, 50]. That is, Scott-KnottESD ranks, and partitions mean values such that mean values in the same partition have no significant differences (based on Cohen's *d* effect size) while mean values in different partitions have significant differences. As such, the results from Scott-KnottESD are statistically significant partitions without any overlap. The Double Scott-KnottESD rank statistical test is the double application of the Scott-KnottESD on mean values. Double Scott-KnottESD re-ranks the Scott-KnottESD results on a general scale [14, 15]. The essence of Double Scott-KnottESD is to ensure that the statistical rankings are not dependent on the evaluation metric values (accuracy and AUC).

4. Experimental Framework

As shown in Fig. 1, this section explains the experimental context of this research. To investigate the impact of varying threshold values on RBF methods SDP, 4 RBF methods (See Table 2) with varying threshold values (log2N, Top 20%, Top 30%, and Top 50%) are experimented with 2 classification techniques (NB and DT) on 25 software defect datasets (See Table 4). The Cross-validation (CV) technique was utilized as the experimental assessment tool to mitigate data variability and the possible overfitting problem [51]. The choice of the CV technique is to minimize the bias and variance of experimented models [52]. Specifically, in the k-fold CV technique (in this study, k=10), each dataset is randomly split into k folds of approximately equal proportions. Each fold is utilized iteratively on the remaining k-1 folds [51-54]. The Waikato Environment for Knowledge Analysis (WEKA) machine learning library [55] and R programming language [56] are utilized for the experimentation on an Intel(R) Core™ machine equipped with i7-6700, running at a speed of 3.4 GHz CPU with 16 GB RAM.

The experimental framework is sub-divided into 2 distinct levels:

i. Rank-Based Filter FS Method Level:

Each of the RBF techniques, as presented in Table 2, is deployed on the training dataset for software defect datasets as it can be seen in Table 6. That is, CSF, COF, IGF and RFE with Ranker Search Method are used to assess and rank features of each dataset. Relevant and top-ranked features generated by each of the RBF methods will be selected based on varying threshold values (log2N, Top 20%, Top 30%, and Top 50%) (where N is the number of features in each dataset). The essence of this is to investigate and determine which threshold value selects the most important features (features with best predictive performance values) irrespective of the characteristics of the defect datasets and RBF methods in SDP. In the end, datasets with reduced features will be generated. The original software defect datasets are pre-processed at the end of this stage.

ii. SDP Model and Performance Evaluation Level:

At this level, SDP models based on NB and DT classification techniques are built based on the 10-folds CV technique. The main objective of this level is to measure the efficacy and importance of lowered software metrics in SDP. Besides, the 10-fold CV will help address biases and overfitting of the ensuing prediction patterns. Most especially class inequality problem is a lurking data quality problems [40, 41, 57]. The predictive performances of the resulting SDP approaches are assessed according to accuracy and AUC values. To guard against chance divisions of the data, the 10-fold CV was repeated 10 times (10X10 execution for each model development) and the average values of the performance metrics were obtained [58-60]. Table 5 shows labels for each RBF method with varying threshold values.

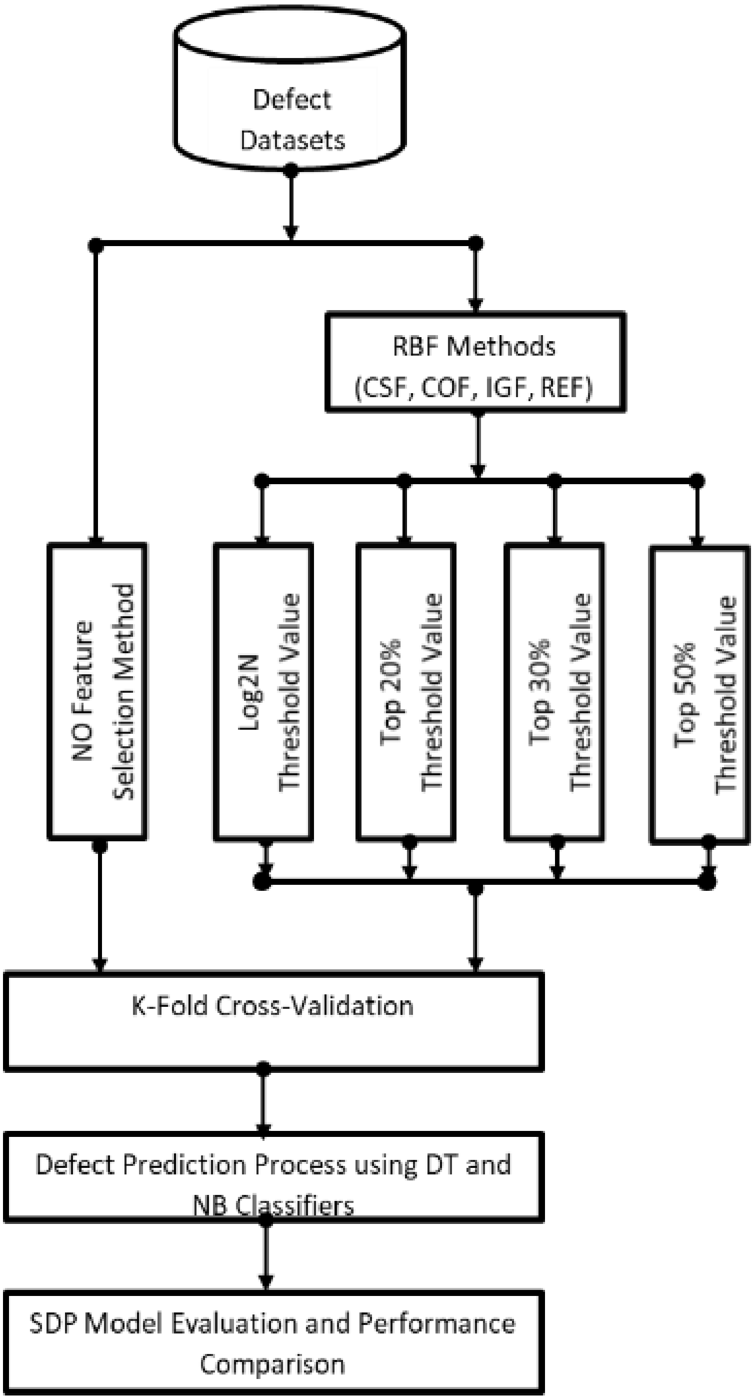
Defect Datasets
RBF Methods (CSF, COF, IGF, REF)
NO Feature Selection Method
Log2N Threshold Value
Top 20% Threshold Value
Top 30% Threshold Value
Top 50% Threshold Value
K-Fold Cross-Validation
Defect Prediction Process using DT and NB Classifiers
SDP Model Evaluation and Performance Comparison

Fig. 1. Experimental framework.

Table 5. Labelled RBF methods with different threshold values.

Rank-based Filter Methods	Threshold Values	Labels
Chi-Square Filter (CSF)	Full Features	RBF1
	Log2N	RBF2
	Top20%	RBF3
	Top30%	RBF4
	Top50%	RBF5
Correlation Filter (COF)	Full Features	RBF6
	Log2N	RBF7
	Top20%	RBF8
	Top30%	RBF9
	Top50%	RBF10
Information Gain Filter (IGF)	Full Features	RBF11
	Log2N	RBF12
	Top20%	RBF13
	Top30%	RBF14
	Top50%	RBF15
Relief Filter (REF)	Full Features	RBF16
	Log2N	RBF17
	Top20%	RBF18
	Top30%	RBF19
	Top50%	RBF20

5. Results and Discussion

The experimental findings based on the experimental context are discussed in this section (See Fig. 1). Accuracy and AUC are used to evaluate the efficacy of the ensuing prediction models from the impact of threshold values for RBF methods. WEKA machine learning tool was used to build all prediction models, R-language was used for statistical analysis and OriginLab was deployed for the graph analysis. The experimental results were analysed based on each of the studies of RBF methods. The essence of this is to investigate and validate the effect of change in threshold values for selecting top-ranked features in studied RBF FS methods.

Figure 2 shows the box-plot interpretations of the prediction performance of CSF on NB and DT classifiers with varying threshold values. Based on the average accuracy of the NB classifier, RBF3 (See Table 5) had the highest average accuracy value of 81.72%, followed by RBF2 with 80.47% and then RBF4 (80.1%) and RBF5 (77.61%). Except for RBF5, other labels (RBF2, RBF3, and RBF4) were superior to RBF1 (when all full features are used). Based on NB average AUC values, RBF3 (0.779) was superior to other methods (RBF1 (0.745), RBF2 (0.766), RBF4 (0.767), RBF5 (0.759)). In the case of the DT classifier, according to accuracy, the results were similar to that of NB. RBF3 recorded the highest average accuracy of 84.29 followed by RBF4 (83.34%), RBF2 (83.33%), and RBF5 (82.94%). Concerning average AUC values, RBF3 had an average AUC value of 0.706 which was superior to the average AUC values of RBF1 (0.671), RBF2 (0.69), RBF3 (0.686), and RBF5 (0.686). From these, in the case of CSF, RBF3 (CSF with Top20%) is superior to other methods (RBF1, RBF2, RBF4, and RBF5) on accuracy and AUC values with NB and DT classifiers.

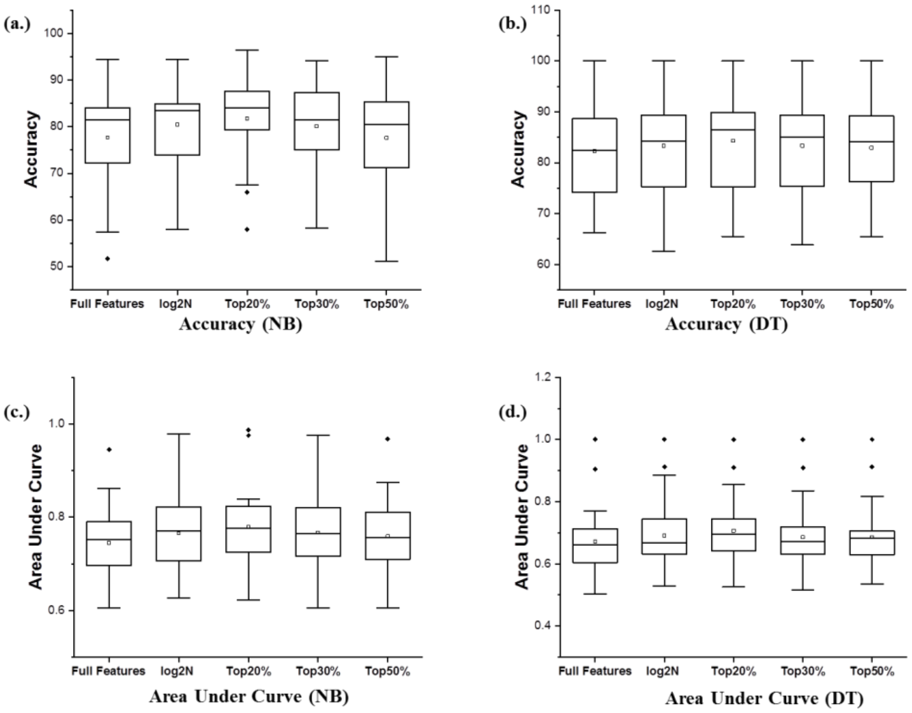

Fig. 2. Box-Plot representations of the predictive performances of NB and DT models based on CSF with varied thresholds.

Just as in the case of CSF, Fig. 3 demonstrates the box-plot interpretations of the predictive performance of COF on NB and DT classifiers with varying threshold values. Based on the average accuracy of the NB classifier, RBF8 (See Table 5) recorded the best average accuracy value of 80.87%, followed by RBF7 with 80.36% and then RBF9 (79.46%) and RBF10 (78.75%). It was also observed that RBF7, RBF8, RBF9 and RBF10 were superior to RBF6 (when all full features are used). With respect to average AUC values, RBF8 (0.775) was superior to other methods (RBF6 (0.745), RBF7 (0.77), RBF9 (0.764), RBF10 (0.765)). For the DT classifier, based on accuracy, RBF8 had the highest average accuracy of 84.1% followed by RBF7 (83.67%), RBF9 (83.05%), and RBF10 (83.04%). As for average AUC values, RBF8 outperforms other methods (RBF6 (0.675), RBF7 (0.69), RBF9 (0.685), and RBF10 (0.686)) with an average AUC value of 0.71. Similar to CSF, RBF8 (COF with Top20%) is superior to other methods (RBF6, RBF7, RBF9, and RBF10) in accuracy and AUC values with NB and DT classifiers.

Figure 4 shows the box-plot interpretations of the predictive performance of IGF on NB and DT classifiers with different threshold values. Based on average accuracy of NB classifier, RBF13 (80.38%) was superior to RBF11 (77.56%), RBF12 (79.77%), RBF14 (78.76%), and RBF15 (77.58%). With average AUC values, RBF13 (0.768) and RBF14 (0.768) recorded same performance but were superior to other methods (RBF11 (0.745), RBF12 (0.764), and RBF15 (0.762)). Regarding the DT classifier, based on accuracy, RBF13 had the highest average accuracy of 83.57% followed by RBF12 (83.49%), RBF14 (83.47%), and RBF15

(83.16%). Although, there exist no significant differences in the average accuracy values of these methods. Concerning average AUC values, RBF13 had an average AUC value of 0.706 which was superior to the average AUC values of RBF11 (0.68), RBF12 (0.692), RBF14 (0.694), and RBF5 (0.689. Conclusively, RBF13 (IGF with Top20%) is superior to RBF11, RBF12, RBF14, and RBF5) on accuracy and AUC values with NB and DT classifiers.

Figure 5 shows the box-plot interpretations of the prediction performance of REF on NB and DT classifiers with varying threshold values. Based on the average accuracy of the NB classifier, RBF18 had the highest average accuracy value of 82.06%, followed by RBF17 with 81.42% and then RBF19 (80.9%) and RBF20 (79.73%). Based on NB average AUC values, RBF19 (0.772) was superior to other methods (RBF16 (0.737), RBF17 (0.748), RBF18 (0.763), RBF20 (0.769)). In the case of the DT classifier, with reference to accuracy, RBF18 recorded the highest average accuracy of 83.69% followed by RBF19 (82.8%), RBF16 (82.68%), RBF17 (82.58%) and RBF20 (82.48%). Concerning average AUC values, RBF18 had an average AUC value of 0.694 which was superior to the average AUC values of RBF16 (0.667), RBF17 (0.648), RBF19 (0.687), and RBF20 (0.685). Nonetheless, in the case of REF, RBF18 (REF with Top20%) is superior to RBF16, RBF17, RBF19, and RBF20 on accuracy and AUC values with DT and NB classifiers except in the case of AUC values for NB where RBF19 had the highest value.

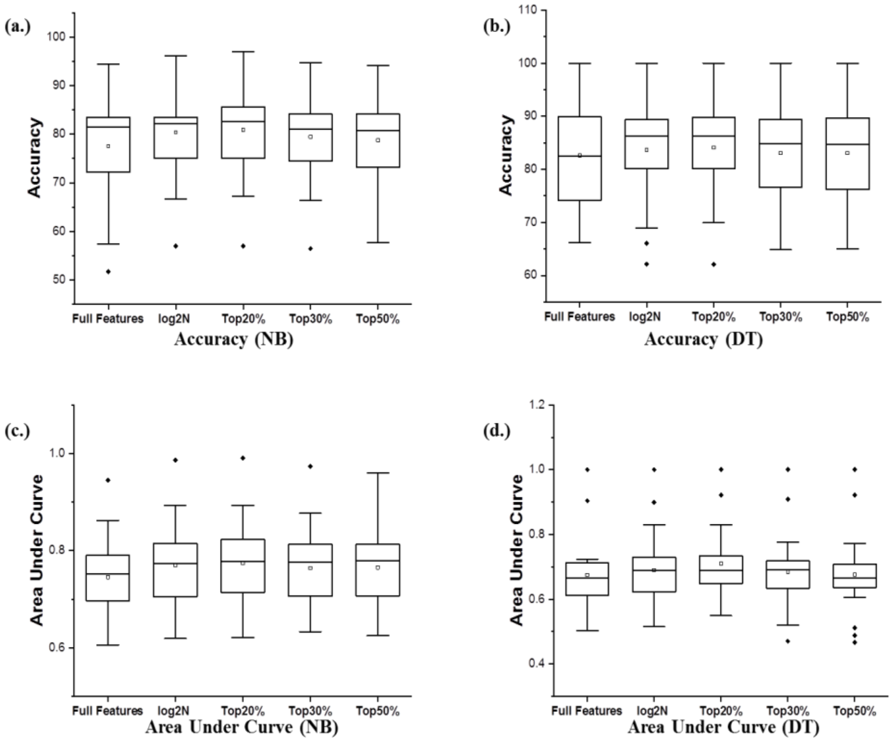

Fig. 3. Box-Plot representations of the predictive performances of NB and DT models based on COF with varied thresholds.

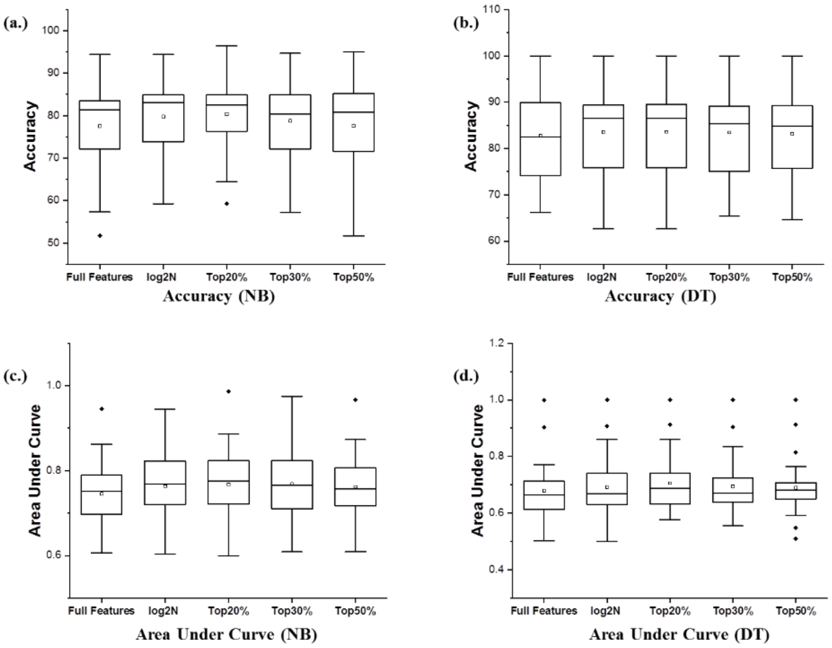

Fig.4. Box-Plot representations of the predictive performances of NB and DT models based on IGF with varied thresholds.

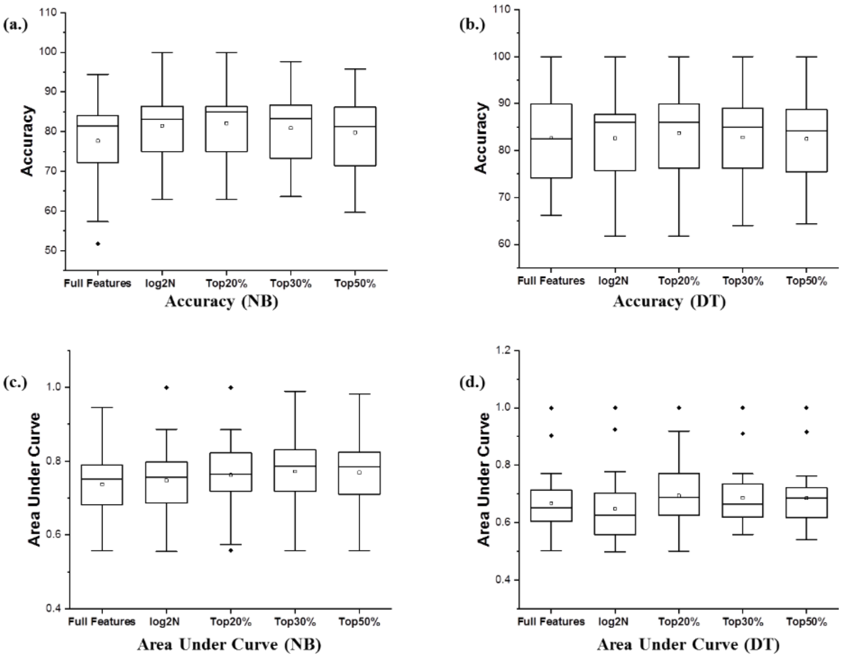

Fig. 5. Box-Plot representations of the predictive performances of NB and DT models based on REF with varied thresholds.

Summarily, Figs. 2-5 present the box-plot interpretations of the predictive performance of CSF, COF, IGF and RFF on NB and DT classifiers respectively with different threshold values (Full features, Log2N, Top20%, Top30%, Top50%) on the studied datasets. From the representations, it was observed that prediction models (NB and DT) based on RBF methods were superior to those when no FS method (Full features) was used. That is, RBF methods improve the predictive performances of NB and DT classifiers. This result coincides with current research studies on the impact of FS techniques on SDP models [11, 13-15, 24, 25, 35]. Also, models with RBF with Top20% ranked features were superior to other SDP models based on other investigated threshold values (Full feature, Log2N, Top30%, and Top50%).

To further strengthen the experimental results, a double Scott-Knott statistical test was performed on the experimental results to determine if there are statistically significant differences in the impact of the threshold values on RBF methods with NB and DT classifiers over the studied datasets. Figure 6 presents the Double Scott-Knott Statistical Test Results according to Accuracy and AUC values of NB and DT Classifiers with varying threshold values. Table 6 summarizes the statistical rank test results as shown in Fig. 6.

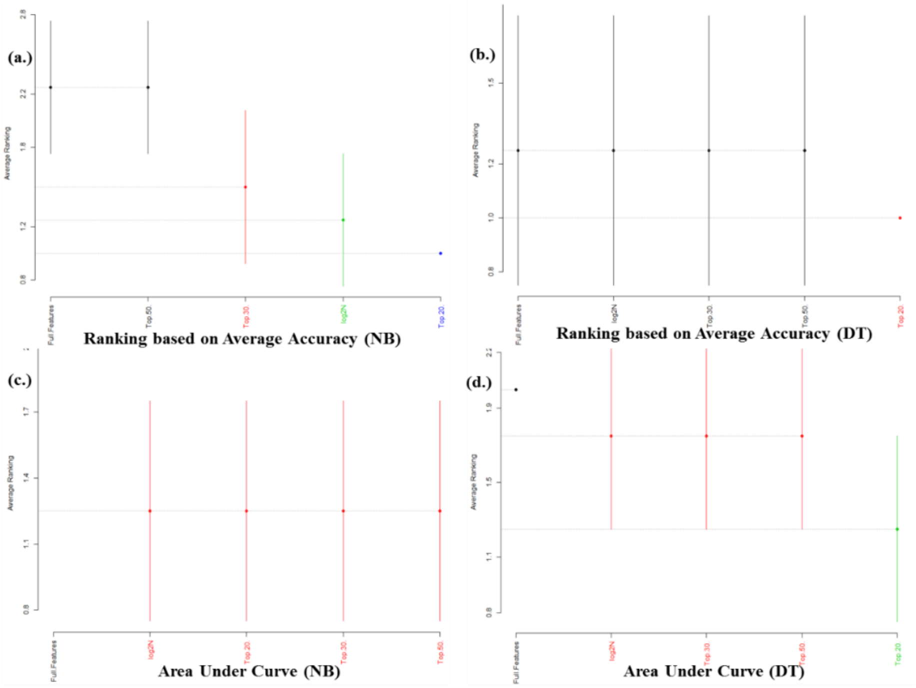

Fig. 6. Double Scott-Knott statistical rank test results based on accuracy and AUC values of RBF-based NB and DT models with varying threshold values.

From Table 6, based on the average accuracy value, it can be observed that the Top20% threshold value ranked first with a significant difference from other threshold values for SDP models with reference to NB and DT classifiers. Log2N threshold value ranked second in the case of NB classifier; however, there is no significant difference in its (log2N) impact on the RBF than Top30 and Top 50% threshold values for DT-based models. Based on the average AUC value, the

Top20% threshold value ranked first, while other threshold values (log2N, Top20%, and Top50%) ranked second while using full feature ranked third for RBF-based DT models. There is no statistically significant difference among the threshold values for RBF-based NB models. That is, Log2N, Top205, Top30% and Top50% threshold values all ranked jointly first while full features ranked last. Scott-Knott rank test of each RBF method can be seen in *Appendix A* (see Figs. A-1 to A-4).

Table 6. Summary of double Scott-Knott rank test of RBF-based NB and DT models with different threshold values.

Statistical Ranking based on Average Accuracy				**Statistical Ranking based on Average AUC**			
NB		**DT**		**NB**		**DT**	
Rank	**Threshold Value**	**Rank**	**Threshold Value**	**Rank**	**Threshold Value**	**Rank**	**Threshold Value**
1	Top20%	1	Top20%	1	Top20%, Log2N, Top30%, Top50%	1	Top20%
2	Log2N	2	Log2N, Top30%, Top50%, Full Features	2	Full Features	2	Log2N, Top30%, Top 50%
3	Top30%					3	Full Features
4	Top50%, Full Features						

It can be deduced that using Top20% as a threshold value for RBF methods proved to be superior to other threshold values (Full features, Log2N, Top30% and Top50%) irrespective of the classification algorithm used. Of utmost concern is the Log2N threshold as most existing studies have been based on it for selecting top-ranked features. From the research findings, we recommend the usage of the Top20% threshold value for the selection of top-ranked features on RBF feature selection techniques in SDP.

6. Conclusions

The experimental results showed that selecting the appropriate threshold values for RBF methods is crucial as the impact of threshold values on RBF varies. Accordingly, using a Top20% threshold value for selecting top-ranked features in SDP proved to be superior to other threshold values (Full features, Log2N, Top30% and Top50%). Also, it was observed that RBF methods have a positive effect on the prediction performances of SDP models which coincides with current literature. Hence, this research study recommends the usage of the Top20% threshold value for the selection of top-ranked features for RBF methods in SDP. The above guidelines will serve as a guide to choosing the acceptable threshold value and RBF methods for SDP for software professionals and researchers. Latent data quality concerns for instance class disparity, outliers, data imputation and ultimate values relating to FS techniques in SDP will be investigated in future works.

Abbreviations

AUC	Area Under the Curve
COF	Correlation Filter
CSF	Chi-Squared Filter
CV	Cross Vali
DT	Decision Tree
FFR	Filter Feature Rank
FN	False Negative
FP	False Positive
FS	Feature Selection
IGF	Information Gain Filter
ML	Machine Learning
NASA	National Aeronautics and Space Administration
NB	Naïve Bayes
PCA	Principal Component Analysis
RBF	Rank-based Filter
REF	Relief Filter
SDP	Software Defect Prediction
SDLC	Software Development Life Cycle
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
WEKA	Waikato Environment and Knowledge Analysis

References

1. Bajeh, A.O.; Oluwatosin, O.J.; Basri, S.; Akintola, A.G.; and Balogun, A.O. (2020). Object-oriented measures as testability indicators: An empirical study. *Journal of Engineering Science and Technology* (*JESTEC*), 15(2), 1092-1108.
2. Balogun, A.; Bajeh, A.; Mojeed, H.; and Akintola, A. (2020). Software defect prediction: A multi-criteria decision-making approach. *Nigerian Journal of Technological Research*, 15(1), 35-42.
3. Basri, S.; Almomani, M.A.; Imam, A.A.; Thangiah, M.; Gilal, A.R.; and Balogun, A.O. (2019). The organisational factors of software process improvement in small software industry: Comparative study. *Proceedings of the 4th International Conference of Reliable Information and Communication Technology (IRICT)*, Johor, Malaysia. 1132-1143.
4. Mabayoje, M.A.; Balogun, A.O.; Bello, S.M.; Atoyebi, J.O.; Mojeed, H.A.; and Ekundayo, A.H. (2019). Wrapper feature selection based heterogeneous classifiers for software defect prediction. *Adeleke University Journal of Engineering and Technology* (*JESTEC*), 2(1), 1-11.
5. Mabayoje, M.A.; Balogun, A.O.; Jibril, H.A.; Atoyebi, J.O.; Mojeed, H.A.; and Adeyemo, V.E. (2019). Parameter tuning in KNN for software defect prediction: An empirical analysis. *Jurnal Teknologi dan Sistem Komputer*, 7(4), 121-126.
6. Balogun, A.O.; Lafenwa-Balogun, F.B.; Mojeed, H.A.; Adeyemo, V.E.; Akande, O.N.; Akintola, A.G.; Bajeh, A.O.; and Usman-Hamza, F.E. (2020). SMOTE-based homogeneous ensemble methods for software defect

prediction. *Proceeding of the 20th International Conference on Computational Science and Its Applications (ICCSA)*, Cagliari, Italy, 615-631.

7. Balogun, A.O.; Shuib, B.; Abdulkadir, S.J.; and Sobri, A. (2019). A hybrid multi-filter wrapper feature selection method for software defect predictors. *International Journal of Supply Chain Management*, 8(2), 916-922.
8. Chauhan, A.; and Kumar, R. (2020). Bug severity classification using semantic feature with convolution neural network, *Proceedings of International Computing in Engineering and Technology*, Aurangabad, India, 327-335.
9. Li, N.; Shepperd, M.; and Guo, Y. (2020). A systematic review of unsupervised learning techniques for software defect prediction. *Information and Software Technology*, 122, 106287.
10. Mojeed, H.A.; Bajeh, A.O.; Balogun, A.O.; and Adeleke, H.O. (2019). Memetic approach for multi-objective overtime planning in software engineering projects. *Journal of Engineering Science and Technology* (*JESTEC*), 14(6), 3213-3233.
11. Afzal, W.; and Torkar, R. (2016). Towards benchmarking feature subset selection methods for software fault prediction. *Computational Intelligence and Quantitative Software Engineering*, 33-58.
12. Anbu, M.; and Mala, G.A. (2019). Feature selection using firefly algorithm in software defect prediction. *Cluster Computing*, 22(5), 10925-10934.
13. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; and Hashim, A.S. (2019). Performance analysis of feature selection methods in software defect prediction: A search method approach. *Applied Sciences*, 9(13), 2764.
14. Ghotra, B.; McIntosh, S.; and Hassan, A.E. (2017). A large-scale study of the impact of feature selection techniques on defect classification models. *Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, Piscataway, NJ, USA, 146-157.
15. Xu, Z.; Liu, J.; Yang, Z.; An, G.; and Jia, X. (2016). The impact of feature selection on defect prediction performance: An empirical comparison. *Proceeding of IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, Ottawa, Canada, 1-12.
16. Zemmal, N.; Azizi, N.; Sellami, M.; Zenakhra, D.; Cheriguene, S.; Dey, N.; and Ashour, A.S. (2018). Robust feature selection algorithm based on transductive SVM wrapper and genetic algorithm: application on computer-aided glaucoma classification. *International Journal of Intelligent Systems Technologies and Applications*, 17(3), 310-346.
17. Balogun, A.O.; Basri, S.; Mahamad, S.; Abdulkadir, S.J.; Capretz, L.F.; Imam, A.A.; Almomani, M.A.; Adeyemo, V.E.; and Kumar, G. (2021). Empirical analysis of rank aggregation-based multi-filter feature selection methods in software defect prediction. *Electronics*, 10(2), 179.
18. Balogun, A.O.; Basri, S.; Capretz, L.F.; Mahamad, S.; Imam, A.A.; Almomani, M.A.; Adeyemo, V.E.; and Kumar, G. (2021). An adaptive rank aggregation-based ensemble multi-filter feature selection method in software defect prediction. *Entropy*, 23(10), 1274.
19. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Mahamad, S.; Al-momamni, M.A.; Imam, A.A.; and Kumar, G.M. (2020). Rank aggregation based multi-filter feature selection method for software defect prediction. *Proceedings of the 2nd*

International Conference on Advances in Cyber Security, Penang, Malaysia, 371-383.

20. Hall, M.A.; and Holmes, G. (2003). Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data engineering*, 15(6), 1437-1447.
21. He, P.; Li, B.; Liu, X.; Chen, J.; and Ma, Y. (2015). An empirical study on software defect prediction with a simplified metric set. *Information and Software Technology*, 59, 170-190.
22. Kakkar, M.; and Jain, S. (2016). Feature selection in software defect prediction: a comparative study. *Proceedings of the 6th International Conference on Cloud System and Big Data Engineering*, Noida, India, 1-10.
23. Kondo, M.; Bezemer, C.-P.; Kamei, Y.; Hassan, A.E.; and Mizuno, O., (2019). The impact of feature reduction techniques on defect prediction models. *Empirical Software Engineering*, 24, 1-39.
24. Muthukumaran, K.; Rallapalli, A.; and Murthy, N.B. (2015). Impact of feature selection techniques on bug prediction models, *Proceedings of the 8th India Software Engineering Conference*, Bangalore, India, 1-10.
25. Rathore, S.S.; and Gupta, A. (2014). A comparative study of feature-ranking and feature-subset selection techniques for improved fault prediction, *Proceedings of the 7th India Software Engineering Conference*, Chennai, India, 1-10.
26. Mabayoje, M.A.; Balogun, A.O.; Bajeh, A.O.; and Musa, B.A. (2018). Software defect prediction: Effect of feature selection and ensemble methods. *FUW Trends in Science & Technology Journal*, 3(2), 518-522.
27. Wang, H.; Khoshgoftaar, T.M.; Van Hulse, J.; and Gao, K. (2011). Metric selection for software defect prediction. *International Journal of Software Engineering and Knowledge Engineering*, 21(2), 237-257.
28. Wang, S.; and Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2), 434-443.
29. Belouch, M.; Elhadaj, S.; and Idhammad, M. (2018). A hybrid filter-wrapper feature selection method for DDoS detection in cloud computing. *Intelligent Data Analysis*, 22(6), 1209-1226.
30. Majd, A.; Vahidi-Asl, M.; Khalilian, A.; Poorsarvi-Tehrani, P.; and Haghighi, H. (2020). SLDeep: Statement-level software defect prediction using deep-learning model on static code features. *Expert Systems with Applications*, 147, 113156.
31. Shivaji, S.; Whitehead, E.J.; Akella, R.; and Kim, S. (2012). Reducing features to improve code change-based bug prediction. *IEEE Transactions on Software Engineering*, 39(4), 552-569.
32. Hall, T.; Beecham, S.; Bowes, D.; Gray, D.; and Counsell, S. (2012). A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, 38(6), 1276-1304.
33. Akintola, A.G.; Balogun, A.O.; Lafenwa-Balogun, F.; and Mojeed, H.A. (2018). Comparative analysis of selected heterogeneous classifiers for software defects prediction using filter-based feature selection methods. *FUOYE Journal of Engineering and Technology* (*JESTEC*), 3(1), 134-137.

34. Rodriguez, D.; Ruiz, R.; Cuadrado-Gallego, J.; Aguilar-Ruiz, J.; and Garre, M. (2007). Attribute selection in software engineering datasets for detecting fault modules, *Proceedings of 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)*, Lubeck, Germany, 1-6.
35. Gao, K.; Khoshgoftaar, T.M.; Wang, H.; and Seliya, N. (2011). Choosing software metrics for defect prediction: an investigation on feature selection techniques. *Software: Practice and Experience*, 41(5), 579-606.
36. Khoshgoftaar, T.M.; Gao, K.; and Napolitano, A. (2012). An empirical study of feature ranking techniques for software quality prediction. *International Journal of Software Engineering and Knowledge Engineering*, 22(02), 161-183.
37. 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.; and Bajeh, A.O. (2020). Impact of feature selection methods on the predictive performance of software defect prediction models: An extensive empirical study. *Symmetry*, 12(7), 1147.
38. Ghotra, B.; McIntosh, S.; and Hassan, A.E. (2017). A large-scale study of the impact of feature selection techniques on defect classification models. *Proceedings of 14th International Conference on Mining Software Repositories (MSR), 2017 IEEE/ACM*, Buenos Aires, Argentina, 1-12.
39. Osman, H.; Ghafari, M.; and Nierstrasz, O. (2017). Automatic feature selection by regularization to improve bug prediction accuracy. *Proceedings on IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE),* Klagenfurt, Austria, 1-6.
40. Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Adeyemo, V.E.; Imam, A.A.; and Bajeh, A.O. (2019). Software Defect Prediction: Analysis of Class Imbalance and Performance Stability. *Journal of Engineering Science and Technology* (*JESTEC*), 14(6), 3294-3308.
41. Yu, Q.; Jiang, S.; and Zhang, Y. (2017). The performance stability of defect prediction models with class imbalance: An empirical study. *IEICE TRANSACTIONS on Information and Systems*, 100(2), 265-272.
42. Menzies, T.; Greenwald, J.; and Frank, A. (2007). Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, 33(1), 2-13.
43. Lessmann, S.; Baesens, B.; Mues, C.; and Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4), 485-496.
44. Shepperd, M.; Song, Q.; Sun, Z.; and Mair, C. (2013). Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering*, 39(9), 1208-1215.
45. Wu, R.; Zhang, H.; Kim, S.; and Cheung, S.-C. (2011). *R*elink: recovering links between bugs and changes. *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of Software Engineering*, Szeged, Hungary, 1-11.

46. Song, Q.; Guo, Y.; and Shepperd, M. (2019). A comprehensive investigation of the role of imbalanced learning for software defect prediction. *IEEE Transactions on Software Engineering*, 14(12), 1253-1269.
47. Nam, J.; Fu, W.; Kim, S.; Menzies, T.; and Tan, L. (2017). Heterogeneous defect prediction. *IEEE Transactions on Software Engineering*, 44(9), 874-896.
48. Tantithamthavorn, C.; McIntosh, S.; Hassan, A.E.; and Matsumoto, K. (2018). The impact of automated parameter optimization on defect prediction models. *IEEE Transactions on Software Engineering*, 45(7), 683-711.
49. Rodriguez, D.; Herraiz, I.; Harrison, R.; Dolado, J.; and Riquelme, J.C. (2014). Preliminary comparison of techniques for dealing with imbalance in software defect prediction. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, New York, USA, 1-10.
50. Tantithamthavorn, C.; McIntosh, S.; Hassan, A.E.; and Matsumoto, K. (2016). An empirical comparison of model validation techniques for defect prediction models. *IEEE Transactions on Software Engineering*, 43(1), 1-18.
51. James, G.; Witten, D.; Hastie, T.; and Tibshirani, R. (2013). *An introduction to statistical learning*. Vol. 112, Springer Cham, Berlin/Hedielberg, Germany.
52. Kuhn, M.; and Johnson, K. (2013). *Applied predictive modeling*. Vol. 26. Springer Cham, Berlin/Hedielberg, Germany.
53. Alsariera, Y.A.; Adeyemo, V.E.; Balogun, A.O.; and Alazzawi, A.K. (2020). Ai meta-learners and extra-trees algorithm for the detection of phishing websites. *IEEE Access*, 8, 142532-142542.
54. Alsariera, Y.A.; Elijah, A.V.; and Balogun, A.O. (2020). Phishing Website Detection: Forest by Penalizing Attributes Algorithm and Its Enhanced Variations. *Arabian Journal for Science and Engineering*, 45, 10459-10470.
55. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I.H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1), 10-18.
56. Crawley, M.J. (2012). *The R book*. John Wiley & Sons.
57. Adeyemo, V.E.; Balogun, A.O.; Mojeed, H.A.; Akande, N.O.; and Adewole, K.S. Ensemble-based logistic model trees for website phishing detection. *Proceedings of 2nd International Conference on Advances in Cyber Security*. 2020, Penang, Malaysia, 627-641.
58. Yadav, S.; and Shukla, S. (2016). Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification, *Proceedings of IEEE 6th International Conference on Advanced Computing (IACC)*, Bhimavaram, India, 78-83.
59. Arlot, S.; and Lerasle, M. (2016). Choice of V for V-fold cross-validation in least-squares density estimation. *The Journal of Machine Learning Research*, 17(1), 7256-7305.
60. Balogun, A.O.; Basri, S.; Jadid, S.A.; Mahamad, S.; Al-momani, M.A.; Bajeh, A.O.; and Alazzawi, A.K. (2020). *S*earch-Based Wrapper Feature Selection Methods in Software Defect Prediction: An Empirical Analysis. *Proceedings of 6th Computer Science On-line Conference*. Zlin, Czech Republic, 492-503.

Appendix A

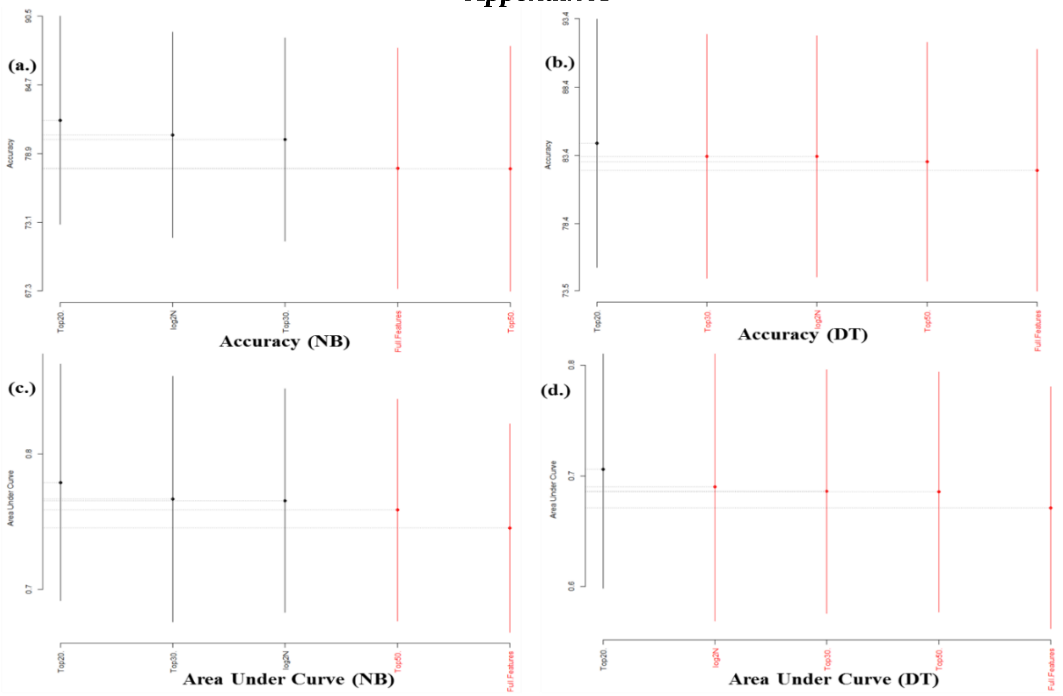

Fig. A-1. Scott-Knott Statistical Test of the predictive performance of models based on CSF with different threshold values on all studied datasets.

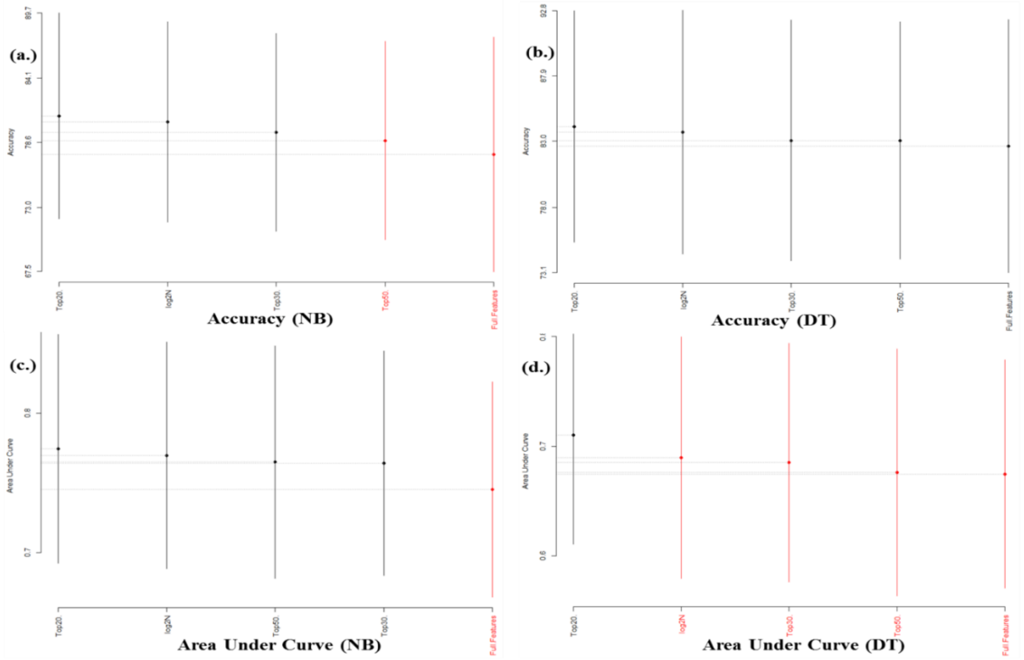

Fig. A-2. Scott-Knott Statistical Test of the predictive performance of models based on COF with different threshold values on all studied datasets.

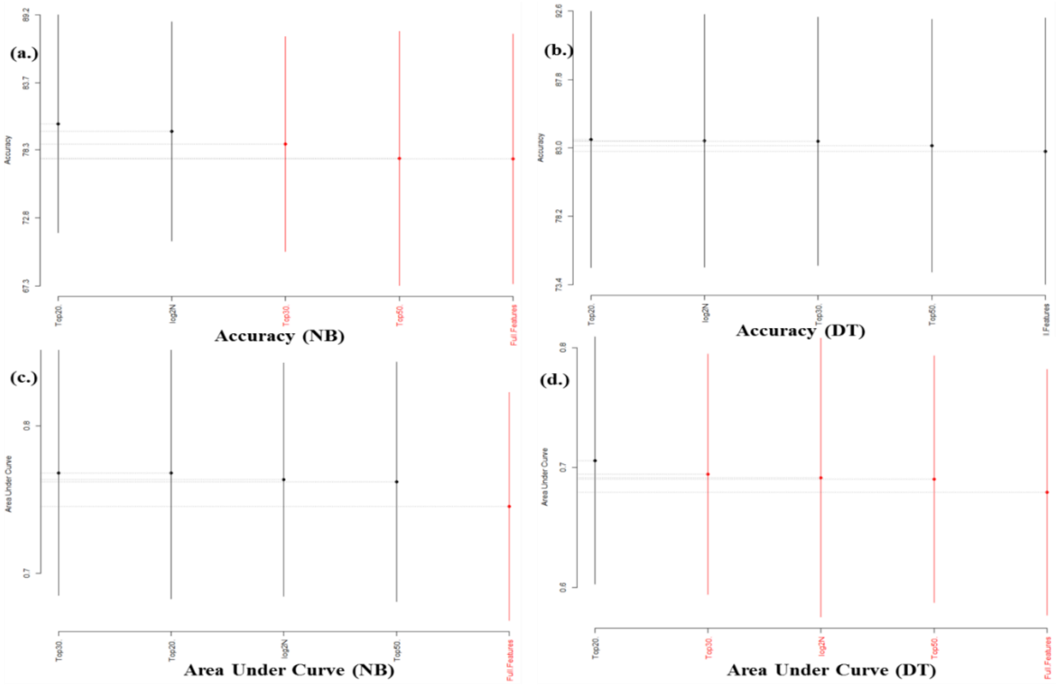

Fig. A-3. Scott-Knott Statistical Test of the predictive performance of models based on IGF with different threshold values on all studied datasets.

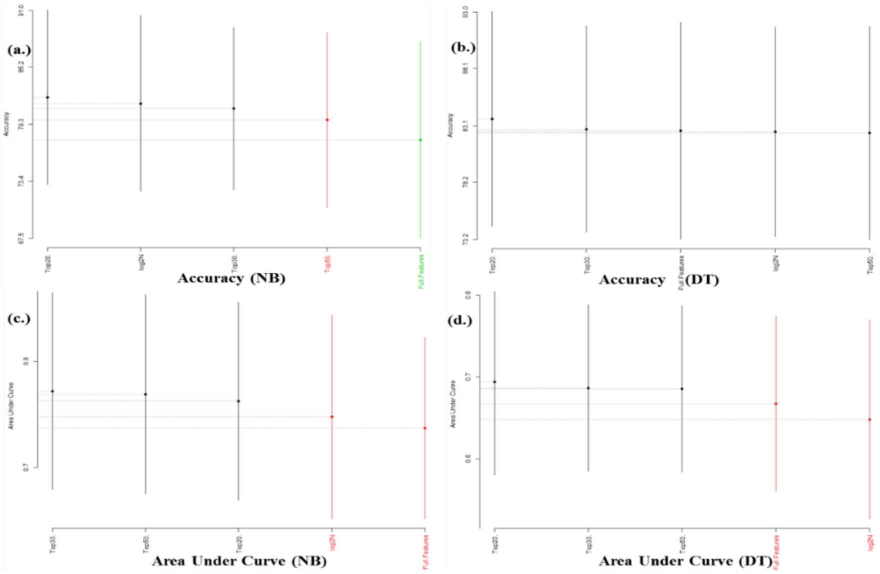

Fig. A-4. Scott-Knott Statistical Test of the predictive performance of models based on IGF with different threshold values on all studied datasets.