The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison

Zhou Xu[†], Jin Liu[†*], Zijiang Yang[‡], Gege An[†], Xiangyang Jia[†]

[†]State Key Laboratory of Software Engineering, School of Computer, Wuhan University, Wuhan, Hubei, China
{zhouxullx, jinliu, gegean, jxy}@whu.edu.cn
[‡]Department of Computer Science, Western Michigan University, Kalamazoo, Michigan, USA
zijiang.yang@wmich.edu

Abstract—**Software defect prediction aims to determine whether a software module is defect-prone by constructing prediction models. The performance of such models is susceptible to the high dimensionality of the datasets that may include irrelevant and redundant features. Feature selection is applied to alleviate this issue. Because many feature selection methods have been proposed, there is an imperative need to analyze and compare these methods. Prior empirical studies may have potential controversies and limitations, such as the contradictory results, usage of private datasets and inappropriate statistical test techniques. This observation leads us to conduct a careful empirical study to reinforce the confidence of the experimental conclusions by considering several potential source of bias, such as the noise in the dataset and the dataset types. In this paper, we investigate the impact of 32 feature selection methods on the defect prediction performance over two versions of the NASA dataset (i.e., the noisy and clean NASA datasets) and one open source AEEEM dataset. We use a state-of-the-art double Scott-Knott test technique to analyze these methods. Experimental results show that the effectiveness of these feature selection methods on defect prediction performance varies significantly over all the datasets.**

Keywords—defect prediction; feature selection; Scott-Knott test;

I. Introduction

Software defect prediction utilizes historical defect data mined from software repositories to determine the quality of software modules for software quality assurance (SQA). Defect prediction can be reviewed as a binary classification problem, i.e., building a prediction model with software metrics (i.e., features) to label new software modules as defect-prone or not [1], [2].

In defect prediction, each software module is characterized by a class label and a set of metrics. The class label denotes whether this module is defective. The metrics are used to build classification models. Effective defect prediction can help SQA team efficiently inspect the potentially defective modules by allocating more software development and maintenance resources [42].

Due to the increased prevalence of data mining and machine learning, a number of classification models have been introduced during the past decade [3], [4], [5], [6], [7], [8]. Nevertheless, a challenge that threatens the modeling process is the high dimensionality of defect datasets, i.e., datasets with excessive features including irrelevant and redundant ones. Existing studies have shown that the high dimensionality problem can lead to extensive computational cost and degradation of the performance of certain specific models [9], [73]. For the foregoing reasons, a variety of feature selection methods were proposed to alleviate this issue of high dimensionality by eliminating irrelevant and redundant features.

Feature selection aims to select a feature subset to replace the original feature set [11]. This feature subset is more effective to distinguish the class labels of software modules. The mainstreams of current research topics focus on proposing and evaluating the effectiveness of a new feature selection method on the performance of defect prediction models. Since there exist a vast variety of feature selection methods, it is critical to compare the effectiveness of different feature selection methods and identify the effective ones.

A. Motivation

Despite that some researchers have conducted empirical studies to explore the impact of feature selection on the defect prediction performance [10], [74], [75], the findings of previous studies are not always consistent regarding the superiority of one feature selection method over others. For example, Gao et al [74] explored the effectiveness of seven feature ranking methods and four feature subset selection methods on a private defect dataset. The results showed that Chi-Square method performs the worst. Wang et al. [75] investigated the effectiveness of six feature ranking methods and two ensemble methods on a private dataset and two open source datasets. The results indicated that Chi-Square method has very good performance. Using different datasets may be the main reason for the inconsistent results [3].

In addition, the results of prior studies [10], [74], [75] show that the effectiveness of different feature selection methods are not significantly different from each other. These findings may be susceptible to the statistical test techniques used, such as the Nemenyi test in [10] and Tukey's honestly significant difference test in [74]. As Ghotra et al. [64] stated that these techniques have limitations for multiple comparison analysis.

[*] Corresponding author

Furthermore, the dataset, like the NASA dataset, used in some prior studies [10], [75] are known to be noisy as it contains several erroneous software modules [47], [64], [68], but no studies have focused on the impact of the noise on the experimental conclusions in the context of feature selection in defect prediction. Thus, the conclusions may be misleading since the noise can affect the conclusions of the empirical studies in software engineering domain [64].

Motivated by the aforementioned observations, we conduct a large scale empirical study on 32 feature selection methods and strive to designate a set of excellent feature selection methods that are significantly distinct to others. We apply a novel double Scott-Knott test technique [71] to rank and cluster these feature selection methods into nonoverlapping groups with statistically significant differences. Further, to observe whether the noise in dataset affects the conclusions of our empirical study, we perform experiments on both noisy and clean NASA datasets. Meanwhile, to diminish the potential impact of dataset types on the conclusions, we use an additional open source AEEEM dataset that was developed in a different setting from that of the NASA dataset.

In this study, we choose the random forest classifier as the defect prediction model and AUC as the evaluation measure. The analytic results show that these feature selection methods can be divided into different groups without overlapping across the groups on all three datasets. This indicates that the effectiveness of these methods is significantly different on the defect prediction performance.

B. Contribution

The main contributions of our empirical study are highlighted in the following four aspects:

(1) We conduct an extensive comparative study on the impact of 32 feature selection methods on the defect prediction performance. To the best of our knowledge, this is the first attempt to perform such a large-scale empirical study on feature selection methods that cover a variety of families.

(2) We employ a state-of-the-art multiple comparison technique to rank and cluster the feature selection methods into distinct groups. Different from the results derived from the post hoc statistical tests in [10] and [74], the Scott-Knott test can yield a nonoverlapping groups with significant differences.

(3) We employ two versions of the NASA dataset and one AEEEM dataset as our study benchmarks to investigate the impact of the noise and the dataset types on the experimental results.

(4) We attempt to identify a group of excellent feature selection methods instead of a single method so it gives practitioners more choices in practical applications.

C. Organization

The reminder of this paper is organized as follows. Section II introduces the preliminaries, i.e., the feature selection methods studied in this work. In section III, we elaborate the experimental setup. Section IV reports the analytic results. In section V, we present the discussion of the experimental results. Section VI and VII state the threats to validity and related work. In Section VIII, we describe the conclusion and future work.

II. Preliminaries

In this section, we only provide a brief description of the 32 feature selection methods studied in this work due to the space limit. These methods cover five families including 14 filter-based feature ranking methods, two filter-based feature subset evaluation methods, 12 wrapper-based feature subset evaluation methods, three clustering-based feature selection methods and one extraction-based feature selection method. Readers can consult with the corresponding references for further details. Table I provides an overview of these methods.

A. Filter-Based Feature Ranking Methods

Filter-based feature ranking methods evaluate each feature separately by assigning individual feature a score according to an indicator.

1) Statistic-Based Methods

- *Chi-Square (CS):* CS[14] measures the merit of a feature by computing its chi-squared statistic to the class label.
- C*orrelation (Cor):* Cor [15] method measures the merit of a feature by computing its Pearson correlation coefficient to the class label.
- *Clustering Variation (CV):* CV [16] method ranks the features according to their variation coefficients. Higher variance coefficient of a feature means that its values vary across a wide range, which benefits to build a more effective prediction model [17].
- *Signal-to-Noise (S2N):* S2N [18], [19] ranks the features by measuring the ratio of the signal (i.e. the class label in this work) to the noise in the feature. The S2N of a feature is defined as follows：
$$S2N = \frac{\mu_P-\mu_N}{\sigma_P-\sigma_N}$$
where μ_P and μ_N denote the mean values of the instances that belong to the positive class (i.e. defect-prone class) and negative class (i.e. non defect-prone class), respectively. σ_P and σ_N denote the corresponding standard deviations.
- *Welch T-Statistic (WTS):* WTS [20] modifies the t-statistic without assuming equal variance with each class label. This statistic is calculated as follows:
$$WTS = \frac{\mu_P - \mu_N}{\sqrt{\frac{{\sigma_P}^2}{n_P}+\frac{{\sigma_N}^2}{n_N}}}$$
where n_P and n_N denote the number of instances that belong to positive class and negative class, respectively.
- *Fisher Score (FS):* FS [21] method ranks the features based on their fisher scores. This statistic is defined as follows:
$$\mathrm{FS} = \frac{n_P(\mu_P-\mu_T)^2+n_N(\mu_N-\mu_T)^2}{{\sigma_T}^2}$$
where μ_T and σ_T denote the mean value and standard deviation of the feature over all instances.

TABLE I OVERVIEW OF THE FEATURE SELECTION METHODS STUDIED IN OUR WORK

Family	Methods		Abbreviation	Label
Filter-based Feature Ranking Methods	Statistic-based Methods	Chi-Square	CS	1
		Correlation	Cor	2
		Clustering Variation	CV	3
		Signal-to-Noise	S2N	4
		Welch T-Statistic	WTS	5
		Fisher Score	FS	6
	Probability-based Methods	Probabilistic Significnace	PS	7
		Information Gain	IG	8
		Gain Ratio	GR	9
		Symmetrical Uncertainty	SU	10
		Maximal Information Coefficient	MIC	11
	Instance-based Methods	ReliefF	RF	12
		ReliefF-Weight	RFW	13
	Classifier-based Techniques	One Rule	OneR	14
Filter-based Subset selection	Correlation-based Feature Subset selection		CFS	15
	Consistency-based Feature Subset selection		ConFS	16
Wrapper-based Subset selection	Naïve Bayes	Root Mean Squared Error	NB+RMSE	17
		Area Under the ROC Curve	NB+AUC	18
		Area Under the Precision-Recall Curve	NB+PRAUC	19
	Repeated Incremental Pruning to Produce Error Reduction	Root Mean Squared Error	RIPPER+RMSE	20
		Area Under the ROC Curve	RIPPER+AUC	21
		Area Under the Precision-Recall Curve	RIPPER+PRAUC	22
	Logistic Regression	Root Mean Squared Error	LR+RMSE	23
		Area Under the ROC Curve	LR+AUC	24
		Area Under the Precision-Recall Curve	LR+PRAUC	25
	k-Nearest Neighbor	Root Mean Squared Error	*k*NN+RMSE	26
		Area Under the ROC Curve	*k*NN+AUC	27
		Area Under the Precision-Recall Curve	*k*NN+PRAUC	28
Clustering-based Methods	FECAR		FECAR	29
	TC		TC	30
	MICHAC		MICHAC	31
Extraction-based Method	Principal Component Analysis		PCA	32

2) *Probability-Based Methods*

- *Probabilistic Significance (PS):* PS [22] is a conditional-probability-based method. Each feature is assigned a significance score according to its contribution to discriminate different class labels. A feature is significant if both the two-way associations between the feature and class label are high.
- *Information Gain (IG):* IG [23] is an entropy-based method. IG measures the reduction of uncertainty about class label after observing the feature. The bias of IG is that it tends to select the features with more values.
- *Gain Ratio (GR):* GR [24] compensates for the bias of IG by penalizing the multivalued features.
- *Symmetrical Uncertainty (SU):* SU [25] compensates for the bias of IG by being divided by the sum of the entropies of the two variables (i.e., the feature and class label in this work).
- *Maximal Information Coefficient (MIC):* MIC [26], [27] is a novel measure of relevance based on entropy. MIC has the advantage of exploring the hidden relationship between two variables and resisting noise.

3) *Instance-Based Methods*

ReliefF [28], an extension of Relief method, is an instance-based method. ReliefF is available in WEKA suite. When the parameter '*WeightByDistance*' is set as false, the method is abbreviated to *RF*, otherwise, it is abbreviated to *RFW*.

4) *Classifier-based Methods*

One Rule (OneR): OneR [29] generates one-level decision rule for individual feature and evaluates the discrimination ability of this rule. The classification error rate is used to rank features separately.

For all above methods, the larger indicator value signifies stronger relevance between the feature and class label.

B. *Filter-Based Feature Subset Evaluation*

1) Correlation-based Feature Subset Selection (CFS): CFS [30] aims to identify a feature subset in which these features have a high correlation with respect to the class label while have a low correlation within each other.

2) Consistency-based Feature Subset Selection (ConFS): ConFS [31] uses an indicator, called consistency [32], to measure the merit of a feature subset. This method aims to search the minimal subset whose consistency is equal to that of all the features.

C. *Wrapper-Based Feature Subset Evaluation*

Wrapper-based methods evaluate the merit of a feature subset with predetermined classifiers and evaluation measures. In this work, we construct 12 methods by employing four classifiers and three evaluation measures that are commonly used.

1) Classifiers

- *Naïve Bayes (NB):* NB [33] is a probability-based classifier. It assumes that the features are conditional independence. This assumption is not always valid, but it can still yield satisfactory result [34].
- *Repeated Incremental Pruning to Produce Error Reduction (RIPPER):* RIPPER [35] is a rule-based classifier. This classifier generates a rule to randomly split the training set into growing set and pruning set, then improves the fitness of the rule on the training set through the generation and prune phase [36].
- *Logistic Regression (LR):* LR [37] improves linear regression model with a logical function. This classifier is originally designed for binary classification.
- *k-Nearest Neighbor (kNN): k*NN [38] is an instance-based classifier. This classifier assigns the class label of the test instance based on the labels of its *k* nearest training instances with majority voting.

2) Evaluation measures

- *Root Mean Squared Error (RMSE):* RMSE [39] measures the deviations of the predicted values and the corresponding true values.
- *Area Under the ROC Curve (AUC):* AUC [50] is a trade-off between true positive rate (TPR) and false positive rate (FPR).
- *Area Under the Precision-Recall Curve (AUPRC):* AUPRC [51] is a trade-off between precision and recall.

D. Clustering-Based Feature Selection Methods

1) FECAR: FECAR [53] first applies the k-medoids clustering to group the features, and then selects a certain number of features with higher IG scores (cf. Section II.A) from each cluster to constitute the final feature subset.

2) TC: TC [52] first uses SU method (cf. Section II.A) to select the top-ranked features as the initial feature subset, and then employs a threshold-based clustering to eliminate redundant features.

3) MICHAC: MICHAC [54] first employs MIC method (cf. Section II.A) to select the features that have a higher correlation with the class labels, then applies Hierarchical Agglomerative Clustering (HAC) [56] to group the selected features into clusters, finally, it selects one feature with the highest MIC value from each cluster to construct the final feature subset.

All the three methods were recently designed for defect prediction. The first two methods need predetermine the number of selected features while the last method automatically determines the number with a statistic measure called inconsistency coefficient [57].

E. Extraction-based Feature Selection Methods

Principal Component Analysis (PCA): PCA [58] is an extraction-based dimension reduction methods. It transforms original variables (i.e. feature set in this work) that may exist correlations within each other into a set of new orthogonal variables. These new variables are known as principal components.

III. Experimental Setup

A. Datasets

This subsection describes the datasets used in this work. First, we conduct experiment on the noisy NASA dataset. The noise can refer to the incorrect software data caused by some unexpected reasons, such as unintentional errors in collecting or transferring the values of software features [47]. To investigate whether the noise in the dataset affects the experimental conclusion, we further use the clean NASA dataset.

Each project of NASA dataset consists of a set of features characterized by static code metrics, such as LOC counts, Halstead and McCabe complexity metrics [2], [59], [60]. Many prior studies used the noisy NASA dataset over the last decade [42], [64], [65], [66], [67]. To improve the quality of the noisy NASA dataset, Shepperd et al. [68] cleaned the original NASA dataset with some preprocessing criteria. The clean NASA dataset is also prevalent in software engineering domain [64], [69], [70]. Table II presents the statistical information of the two versions of NASA dataset, including the language used in each project, number of features, modules, and the percentage of defective modules. Table III reports the changes of each project after removing noise, where 'Δ Features', 'Δ Modules', 'Δ Defective' denote the number of features, modules and defective modules deleted, respectively. The '% Defective' represents the noise percentage in term of module level. Both versions of NASA dataset are derived from Shepperd et al. [68].

To investigate whether dataset types affect the conclusions that are drew from the two versions of NASA dataset, we use four projects of AEEEM dataset [40]. This dataset was developed in a different setting compared with the NASA dataset and performs defect prediction at class level. Features in AEEEM dataset include source code metrics, such as the change metrics, source code metrics, entropy of source code metrics and churn of source code metrics. Note that no common features exist in these two datasets. Table IV shows the statistical information of the projects. All the projects were written in Java.

B. Research Questions

RQ1: Do different feature selection methods have significantly distinct effectiveness on defect prediction performance over noisy NASA dataset?

RQ2: Is the conclusion consistent with that in *RQ1* when using the clean NASA dataset?

RQ3: What is the impact of the noise on the effectiveness of feature selection methods over NASA dataset?

RQ4: Do dataset types affect the conclusions drew from the two versions of the NASA dataset?

C. Classifier

In this work, we use the *Random Forest* classifier [41] as the defect prediction model. This classifier constructs multiple classification trees during model training. The training set of each tree stems from sampling the whole training set with replacement. Each internal node of a tree is split using a random subset of the whole features. This randomly split-

TABLE II STATISTICS OF THE TWO VERSIONS OF THE NASA DATASETS

Noisy NASA Dataset				
Project	Language	# Features	#Modules	%Defective
CM1	C	40	505	9.50%
JM1	C	21	10878	19.32%
KC1	C++	21	2107	15.42%
KC3	Java	40	458	9.39%
MC1	C & C++	39	9466	0.72%
MC2	C	40	161	32.30%
MW1	C	40	403	7.69%
PC1	C	40	1107	6.87%
PC2	C	40	5589	0.41%
PC3	C	40	1563	10.24%
PC4	C	40	1458	12.21%

Clean NASA Dataset				
Project	Language	# Features	#Modules	% Defective
CM1	C	37	327	12.84%
JM1	C	21	7720	20.88%
KC1	C++	21	1162	25.30%
KC3	Java	39	194	18.56%
MC1	C & C++	38	1847	1.95%
MC2	C	39	125	35.20%
MW1	C	37	251	9.96%
PC1	C	37	696	7.90%
PC2	C	36	734	2.18%
PC3	C	37	1073	12.30%
PC4	C	37	1276	13.79%

TABLE III DIFFERENCES OF THE NOISY NASA DATASET AND THE CLEAN NASA DATASET

Project	**Δ Features**	**Δ Modules**	**Δ Defective**	**% Noise**
CM1	3	178	6	35.25%
JM1	0	3158	508	29.03%
KC1	0	945	31	44.85%
KC3	1	264	7	57.64%
MC1	1	7619	32	80.49%
MC2	1	36	8	22.36%
MW1	3	152	6	37.72%
PC1	3	411	21	37.13%
PC2	4	4855	7	86.87%
PC3	3	490	28	31.35%
PC4	3	182	2	12.48%

TABLE IV STATISTICS OF THE AEEEM DATASET

Project	**Release**	**# Features**	**#Modules**	**%Defective**
Eclipse JDT Core	3.4	76	997	20.66%
Apache Lucene	2.4.0	76	691	9.26%
Mylyn	3.1	76	1862	13.16%
Eclipse PDE UI	3.4.1	76	1497	13.96%

ting assures low correlations within all the decision trees. The class label of the output relies on the class labels of all trees with majority voting. Random forest classifier has been widely used for defect prediction with promising prediction ability [42], [43], [44], [45], [46].

D. Evaluation measure

In this work, we apply AUC to measure the prediction performance of random forest classifier built with the selected features. This measure is robust to imbalance class distribution and misclassification costs that are the characteristics of defect prediction [61]. Therefore, it is widely used as an evaluation measure for defect prediction performance [44], [62], [73].

E. Experimental Procedure

Let m denote the number of original features. For filter-based feature ranking methods, we select the top $\lceil log_2 m \rceil$ features to build random forest classifier. This parameter follows prior studies [49], [74] which suggested that various classifiers for imbalance defect datasets are appropriate to this setting. For FECAR and TC methods, we follow the original work to set the number of selected features as $\lceil log_2 m \rceil$. For PCA, filter-based and wrapper-based feature subset evaluation methods, we use WEKA suite to implement them with default parameter settings, except for the wrapper-based methods with the kNN classifier. In this case, we find that among the five test options (i.e., 10, 20, 30, 40, and 50), $k = 20, 10, 30$ can achieve a lower RMSE , a higher AUC and PRAUC for most projects on AEEEM dataset, noisy and clean NASA datasets respectively. So we choose these parameters for the wrapper-based methods with kNN classifier on the three datasets respectively. Moreover, for MIC and MICHAC, we implement them with MINE toolkit [48].

In the experiment, 10-fold cross validation strategy is performed when evaluating the performance of random forest classifier with the selected features. To mitigate the effect of module orders on the feature selection methods and random forest classifier, we randomize the module orders of the dataset 10 times before performing each feature selection method. Thus, we obtain 10 AUC values of the cross validation. Figure 1 depicts the overall framework of the experimental procedure.

F. Statistic Comparison Tests

1) Friedman Test: Friedman [63] is a non-parametric statistical test based on the rankings of performance values rather than the actual values. In this work, we use this test to detect whether the performance differences among the 32 feature selection methods are random. This statistic is calculated as follows:

$$\chi_F^2 = \frac{12p}{q(q+1)}[\sum_{i=1}^{q}(\frac{1}{p}\sum_{j=1}^{p} r_j^i)^2 - \frac{q(q+1)^2}{4}]$$

where p denotes the total number of projects of the dataset, q denotes the total number of feature selection methods, while r_j^i denotes the ranking of the ith method over the jth project.

2) Scott-Knott Test: Scott-Knott test [71] is a multiple comparison technique using hierarchical clustering algorithm for statistical analysis. This test ranks and clusters the methods into significantly different groups in which the methods in the same group have no significant differences while the methods in distinct groups have significant differences. The advantage of the test is that it results completely distinct groups without any overlapping. Borges et al. [72] evaluated Scott-Knott test using Monte Carlo method and found that this test presents an excellent performance. In this work, we employ the novel double Scott-Knott test [64] to cluster these methods into different groups at the significance level of 0.05. Thus, we can find a set of feature selection methods that is superior to others,

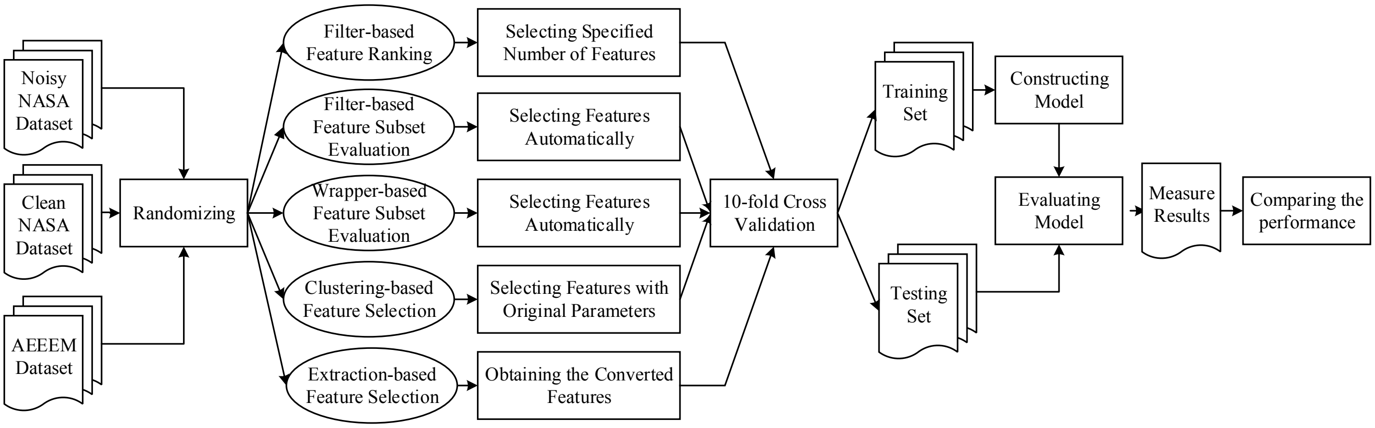
Noisy NASA Dataset
Clean NASA Dataset
AEEEM Dataset
Randomizing
Filter-based Feature Ranking
Filter-based Feature Subset Evaluation
Wrapper-based Feature Subset Evaluation
Clustering-based Feature Selection
Extraction-based Feature Selection
Selecting Specified Number of Features
Selecting Features Automatically
Selecting Features Automatically
Selecting Features with Original Parameters
Obtaining the Converted Features
10-fold Cross Validation
Training Set
Testing Set
Constructing Model
Evaluating Model
Measure Results
Comparing the performance

Figure 1. The overall framework of our experimental procedure.

not just a single one. The double Scott-Knott test first ranks the methods with the AUC values on projects-level, then ranks the methods again with their rankings from the former step on global level. The two-step ranking strategy ensures the rankings are independent of the actual AUC values. Thus, the double Scott-Knott test can perform well when the AUC values on different projects vary greatly.

The detailed steps of the double Scott-Knott test are described as follows: in the first round, we rank and cluster these methods into significantly distinct groups with the 10 AUC values on each project as the inputs. As a result, each method obtains p (p denotes the number of projects of the dataset) different rankings. In the second round, we get the final rankings of these methods with all the rankings of each method as the input.

IV. Experimental Results

In this section, we present the experimental results to answer the four research questions (cf. Section III.B) of this empirical study.

A. ***RQ1***

To answer this question, we apply the 32 feature selection methods to the 11 projects of the noisy NASA dataset. For each method of a project, we obtain 10 AUC values of 10 times cross validation. Then we conduct the Friedman test. The p-value of 6.72 E-16 indicates that the differences in performance values of these methods are not random.

Figure 2 shows the standardized box-plots of AUC values for each project on all methods to illustrate the suitability of the double Scott-Knott test for the experimental results. From the figure, we can observe that even the worst methods for MC1 project and PC4 project perform better than the best-performing methods for other projects, except for PC1 project. Also, many of the methods for PC1 project outperform the best-performing methods for many other projects. This observation manifests that it is appropriate to use the double Scott-Knott test to analyze the results on the noisy NASA dataset.

Figure 3 depicts the result of the double Scott-Knott test on the noisy NASA dataset. Each number on x-axis represents a feature selection method while the corresponding relationship is illustrated in table I. The y-axis represents the range of the rankings of each method on all projects. The dot on the line corresponds to the average ranking of each method. Different colors represent different groups. From the figure, we can observe that the 32 feature selection methods are clustered into four distinct groups without overlapping, which implies that there exist clear separations between these methods on the noisy NASA dataset.

Table V reports the methods that belong to the same group and the statistical properties of the method rankings of each group, including the median ranking, average ranking and standard deviation.

From the table, we can observe that filter-based feature subset evaluation methods, i.e., ConFS and CFS, belong to the first group in which the methods achieve the best performance.

For filter-based feature ranking methods, most of them belong to the second group or the third group. More specifically, all the probability-based methods, i.e., PS, SU, GR, MIC, and IG, belong to the second group. All the instance-based methods, i.e., RFW and RF, also belong to the second group. Most of the statistic-based methods belong to the second group as well, except FS and S2N methods which belong to the first group and the third group, respectively. The classifier-based method, i.e., OneR, belongs to the third

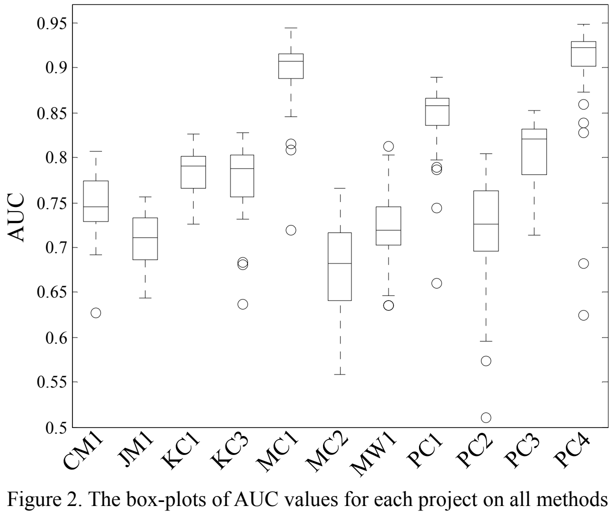

Figure 2. The box-plots of AUC values for each project on all methods over the noisy NASA dataset.

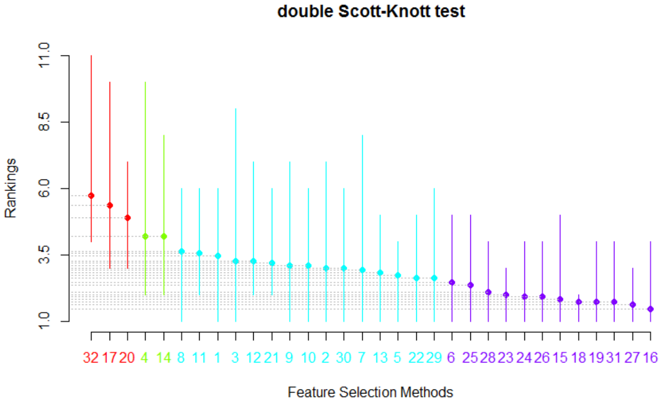

Figure 3. The result of double Scott-Knott test on the noisy NASA dataset.

group.

For wrapper-based feature subset evaluation methods, eight out of twelve methods belong to the first group. More specifically, the methods based on the kNN classifier (i.e., kNN+an evaluation measure) and the LR classifier (i.e., LR+an evaluation measure) achieve the best performance. Two of the methods based on the NB classifier (i.e., NB+an evaluation measure) belong to the first group expect the NB+RMSE method. The methods based on the RIPPER classifier (i.e., RIPPER+an evaluation measure) belong to the second group (for RIPPER+PRAUC and RIPPER+AUC) or the worst group (for RIPPER+RMSE).

For clustering-based feature selection methods, MICHAC method can achieve the best performance compared with most of the filter-based feature ranking methods and other two clustering-based methods, i.e., FECAR and TC. This observation is consistent with the conclusion in [54]. In addition, the FECAR and TC belong to the second group.

For extraction-based feature selection method, PCA is outperformed by nearly all other methods. The reason may be that PCA aims to find the optimal linear projection of the feature set by minimizing the mean square error, but it ignores the class label, so when there exist nonlinear properties in the feature set, the projection direction may not necessarily benefit for classification. In addition, PCA performs well when the dataset follows Gaussian distribution, but the premise is not always established. Thus, PCA may not achieve an anticipated performance.

TABLE V STATISTICAL RESULTS OF THE DOUBLE SCOTT-KNOTT TEST ON THE NOISY NASA DATASET

Overall Ranking	**Feature Selection Methods**	**Median Ranking**	**Average Ranking**	**Standard Deviation**
1	ConFS, kNN+AUC, MICHAC, NB+PRAUC, NB+AUC, CFS, kNN+RMSE, LR+AUC, LR+RMSE, kNN+PRAUC, LR+PRAUC, FS	1.86	1.90	0.29
2	FECAR, RIPPER+PRAUC, WTS, RFW, PS, TC, Cor, SU, GR, IG, RIPPER+AUC, RF, CV, CS, MIC,	3.09	3.08	0.31
3	OneR, S2N	4.18	4.18	0
4	RIPPER+RMSE, NB+RMSE, PCA	5.36	5.33	0.41

RQ1 Summary. As mentioned above, the analytic results indicate that the differences between the effectiveness of the 32 methods on the defect prediction performance are significant over the noisy NASA dataset. These methods are clearly divided into four groups with statistically significant differences.

B. *RQ2*

This question aims to explore whether the conclusion stays consistent after removing noise from the NASA dataset. We apply 32 feature selection methods to the 11 projects of clean NASA dataset. The p-value of $3.17E$-19 for Friedman test indicates that the performance differences among these methods are also not random.

Figure 4 also shows the standardized box-plots of AUC values for each project on all the methods. From the figure, we can find that the worst methods for PC4 project perform well than the best-performing methods for all other projects, and many of the methods for PC1 project outperform the best-performing methods for many other projects, except for PC4 project. So the double Scott-Knott test is also suitable to analyze our experimental results on the clean NASA dataset.

Figure 5 depicts the result of the double Scott-Knott test on the clean NASA dataset. From the figure, we can observe that the 32 feature selection methods are also clustered into four non-overlapping groups. It indicates that these methods are also distinct from each other on the clean NASA dataset.

Table VI reports the methods that belong to each group and the statistics of the method rankings for each group.

From the table, we can observe that filter-based feature subset evaluation methods belong to the best group again.

Most of the filter-based feature ranking methods also belong to the second or third group. More specifically, most of the probability-based methods belong to the second group, except that MIC belongs to the third group. All the instance-based methods belong to the second group. Most of the statistic-based methods belong to the second group, with two exceptions of S2N and CV which belong to the third group

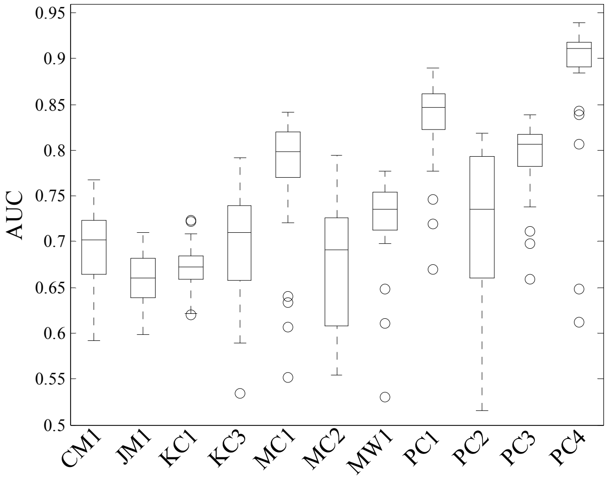

Figure 4. The box-plots of AUC values for each project on all methods over the clean NASA dataset.

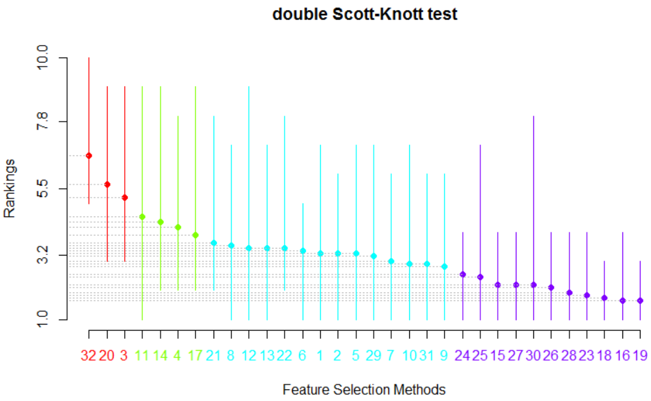

Figure 5. The result of double Scott-Knott test on the clean NASA dataset.

and the worst group, respectively. The classifier-based method belongs to the third group again.

For wrapper-based evaluation methods, eight out of twelve methods belong to the best group. More specifically, the methods based on the kNN classifier and the LR classifier have the best performance again. Two of the methods based on the NB classifier belong to the best group expect the NB+RMSE method. The methods based on the RIPPER classifier belong to the second group (for RIPPER+PRAUC and RIPPER+AUC) and the worst group (for RIPPER+RMSE). This observation is relatively consistent with the experimental results on the noisy NASA dataset.

For clustering-based methods, TC method achieves the best performance while other two methods, i.e., MICHAC and FECAR, belong to the second group. This observation is quite different from that on the noisy NASA dataset, where the MICHAC method performs the best.

For extraction-based feature selection method, PCA is also outperformed by nearly all the other methods.

RQ2 Summary. To sum up, the above observations show that the effectiveness of these feature selection methods exhibits significant differences on the defect prediction performance over the clean NASA dataset with four distinct groups. It also indicates that the conclusion drew from the noisy NASA dataset is not affected by the noise in the dataset.

TABLE VI STATISTICAL RESULTS OF THE DOUBLE SCOTT-KNOTT TEST ON THE CLEAN NASA DATASET

Overall Ranking	Feature Selection Methods	Median Ranking	Average Ranking	Standard Deviation
1	NB+PRAUC, ConFS, NB+AUC, LR+RMSE, kNN+PRAUC, kNN+RMSE, TC, kNN+AUC, CFS, LR+PRAUC, LR+AUC	2.09	2.03	0.31
2	GR, MICHAC, SU, PS, FECAR, WTS, Cor, CS, FS, RIPPER+PRAUC, RFW, RF, IG, RIPPER+AUC	3.27	3.25	0.26
3	NB+RMSE, S2N, OneR, MIC	4.27	4.25	0.27
4	CV, RIPPER+RMSE, PCA	5.64	5.82	0.74

C. *RQ3*

Although the noise does not affect the conclusion that the differences of the effectiveness of these feature selection methods are significant, it has some impacts on the actual effectiveness of these methods. We explore the question from two aspects.

By observing the change of the rankings of the feature selection methods before and after removing the noise from NASA dataset, we find that the rankings of most methods remain unchanged with six exceptions, i.e., FS, CV, MIC, TC, MICHAC, and NB+RMSE. From this perspective, it implies that most methods exhibit stably distinguishable abilities in spite of the noise in the dataset.

By observing the change of the actual performance values of the feature selection methods, we find that the performance of all methods declines, varying from 0.3% (for NB+RMSE method) to 7.9% (for CV method) after cleaning the dataset. From table III, we observe that the noise percentage varies from 12.48% to 86.97% with an average percentage of 43.20%. It indicates that nearly half of the software modules are removed from the original dataset on average. However, these deleted modules may contain important information that is beneficial to distinguish the labels of the modules. Thus it may lead to the reduction of the performance due to the loss of information.

RQ3 Summary. To conclude the above observations, we find that noise in the dataset has little impact on the rankings of the group for most methods. However, the performance of all the methods declines after removing the noise from the dataset due to the loss of some important information.

D. *RQ4*

This question investigates whether dataset types affect the conclusions drew from the two versions of the NASA dataset. In this work, we apply the 32 feature selection methods to a publicly open source AEEEM dataset. The dataset was developed in a different setting compared with the NASA dataset. The p-value of $3.30E$-16 for Friedman test also indicates that the performance differences among these methods are not random.

Figure 6 also provides an explanation of the suitability for the double Scott-Knott test to analyze the experimental results. As we can see that many of the methods for Eclipse JDT Core project (JDT) outperform the best method for Eclipse PDE UI project (PDE).

Figure 7 depicts the result of the double Scott-Knott test on the AEEEM dataset. The figure shows that the 32 feature selection methods are clustered into three distinct groups without overlapping. This indicates that these methods have significantly different effectiveness on the defect prediction performance over the AEEEM dataset.

Table VII reports relevant statistics of the method rankings for each group.

From the table, we can observe that the filter-based feature subset evaluation methods belong to the best group again.

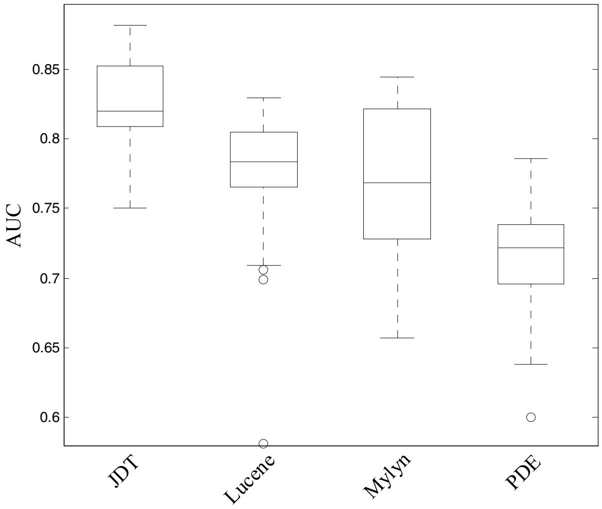

Figure 6. The box-plots of AUC values for each project on all methods over the AEEEM dataset.

Nearly all the filter-based feature ranking methods belong to the second group or the worst group, except the WTS method. More specifically, two out of five probability-based methods, i.e., IG and MIC, belong to the second group, while others belong to the worst group. All the instance-based methods belong to the second group. Four out of six statistic-based methods belong to the second group with two exceptions of WTS and CV which belong to the best group and the worst group, respectively. The classifier-based method belongs to the second group.

Eight out of the twelve wrapper-based feature subset evaluation methods belong to the best group. The methods based on the *k*NN classifier and the LR classifier achieve the best performance. Two of the methods based on the NB classifier belong to the best group expect the NB+RMSE method. The methods based on the RIPPER classifier belong to the worst group. This observation is also similar to that on the two versions of NASA dataset.

All the three clustering-based feature selection methods belong to the second group.

For extraction-based feature selection method, PCA belongs to the second group. This observation is contrary to that on two versions of the NASA dataset. The reason may be that the feature set of the AEEEM dataset has a higher linearity or follows Gaussian distribution compared with the NASA dataset. So, for the AEEEM dataset, the features extracted by PCA are more conductive to defect prediction.

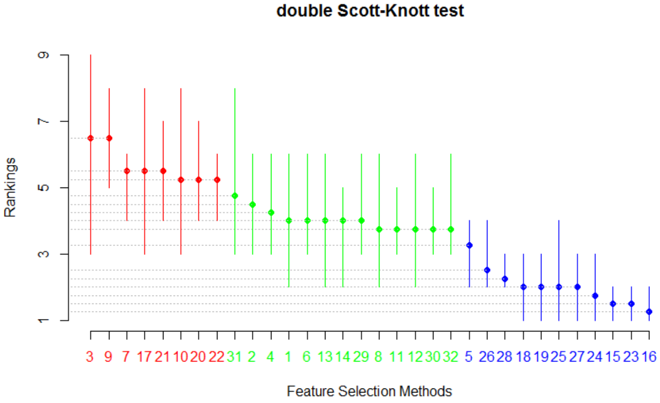

Figure 7. The result of double Scott-Knott test on the AEEEM dataset.

TABLE VII STATISTICAL RESULTS OF THE DOUBLE SCOTT-KNOTT TEST ON THE AEEEM DATASET

Overall Ranking	Feature Selection Methods	Median Ranking	Average Ranking	Standard Deviation
1	WTS, *k*NN+RMSE, *k*NN+PRAUC, NB+AUC, NB+PRAUC, LR+PRAUC, *k*NN+AUC , LR+AUC, CFS, LR+RMSE, ConFS	2	2	0.55
2	MICHAC, Cor, S2N, CS, FS, RFW, OneR, FECAR, IG, MIC, RF, TC, PCA	4	4.02	0.31
3	CV, GR, PS, NB+RMSE, RIPPER+AUC, SU, RIPPER+RMSE, RIPPER+PRAUC	5.5	5.66	0.53

RQ4 Summary. The experimental results on the AEEEM dataset show that the effectiveness of the 32 feature selection methods on defect prediction performance can be significantly different from each other. It also indicates that the conclusions drew from the two versions of the NASA dataset are little affected by the dataset types.

V. DISCUSSION

All the results of RQ1, RQ2 and RQ4 show that the effectiveness of different feature selection methods on the performance of random forest classifier exhibits significant differences, even among the methods of the same family. In addition, we also observe similarities and differences from the experimental results over the three benchmarks. More specifically, the filter-based subset evaluation methods can always achieve the best performance. The wrapper-based methods that based on the *k*NN, LR and NB also always perform the best except the NB+RMSE method. The wrapper-based methods based on the RIPPER never perform well regardless of evaluation measures. It seems to imply that the classifiers affect the performance of the wrapper-based methods more than the evaluation measures. The clustering-based methods always belong to the best group or the second best group. The MICHAC method performs particularly well on the noisy NASA dataset. It confirms the previous work [55] that the MICHAC method is not sensitive to the noise in the dataset due to the characteristics of MIC and HAC. For the filter-based feature ranking methods, the effectiveness of some methods is affected by the dataset types and the noise. For extraction-based method, i.e., PCA, its effectiveness is affected by the dataset types due to its own limitations, such as the linear constraint and the demand for the distribution of dataset.

Generally, the wrapper-based methods outperform others. However, they are more time-consuming during the experimental process, especially when the feature space becomes larger. This observation is consistent with the previous study [55]. In addition, although the filter-based feature subset evaluation methods can always perform best, they tend to select more features than other methods. This may lead to more complex models. The effectiveness of the filter-based feature ranking methods and clustering-based methods is not as good as that of wrapper-based and filter-based feature subset evaluation methods. However, these methods are faster and simpler to understand. Furthermore, they can still yield satisfactory results with less features.

VI. Threats to Validity

A. Construct Validity

Threats to construct validity focus on the bias of the measures used to evaluate the prediction performance. In the work, we employ the widely used AUC as the evaluation measure. Nonetheless, other comprehensive measures, such as F-measure, g-measure can also be considered.

B. Internal Validity

Threats to internal validity refer to the bias of the choice of defect prediction classifiers, feature selection methods. In this work, we only use the random forest classifier due to its popularity in defect prediction. In addition, we choose 32 methods covering five feature selection families to make our empirical study more fruitful.

C. External Validity

Threats to external validity mainly concern the generalization of the experimental results. Although the datasets used in this work have been extensively studied in defect prediction, we still cannot claim that our conclusions can be generalized to other software projects. Nevertheless, this work provides a detailed experimental description, including parameter settings, thus other researchers can easily replicate this empirical study on new datasets.

D. Conclusion Validity

Threats to conclusion validity focus on the statistical analysis method used. In this work, we use the Scott-Knott test to statistically analyze the 32 feature selection methods. Existing work has suggested that the Scott-Knott test is superior to other post-hoc tests [12], [13], [64].

VII. Related Work

Feature selection is introduced into software engineering domain to alleviate the high dimensionality issue by eliminating irrelevant and redundant features. Many prior studies have investigated the effectiveness of feature selection methods on the performance of defect prediction models. Shivaji et al [73] explored the impact of four feature ranking methods and two wrapper methods on the code change-based bug prediction over 11 software projects. They found that feature selection can improve the prediction performance even eliminating 90 percent of original features. Muthukumaran et al. [10] investigated seven feature ranking methods, two wrapper methods and one embedded method on the noisy NASA dataset and AEEEM dataset. They found that there have no significant differences among the 10 methods. Gao et al. [74] applied seven feature ranking methods and four feature subset selection methods to a private dataset. They found that various feature ranking methods, except for CS method, have similar effectiveness. Wang et al. [75] conducted an empirical study on six feature ranking methods and two ensemble methods over three datasets. The results on NASA dataset indicated that the effectiveness of the eight methods have no significant separations.

Different from the work of the above studies, in this work, we do not aim to propose a new feature selection method or compare the performance of a few feature selection methods on a small number of datasets, but to conduct an extensive comparison of 32 feature selection methods on three publicly available datasets and consider the impact of noise and the dataset types on the experimental conclusions.

In addition, this work is also inspired by two analogous studies in defect prediction [42], [64]. Lessmann et al. [42] conducted an empirical study to investigate the performance of 22 prediction models on noisy NASA dataset and Ghotra et al. [64] extended this study by investigating 31 prediction models on two versions of the NASA dataset and an open source software dataset. Different from this two studies, we focus on the effectiveness of feature selection methods rather than prediction models in this work.

VIII. Conclusion and Future Work

In this paper, we conduct a larger scale empirical study to investigate the impact of 32 feature selection methods on the defect prediction performance on account of some potential controversies and limitations in previous studies. To explore whether the noise in the dataset and the dataset types affect the analytic conclusions, we employ two versions of the NASA dataset and one public AEEEM dataset as our study benchmarks. To compare different feature selection methods and identify a set of outstanding methods, we use a state-of-the-art multiple comparison technique to analyze these methods.

The analytic results indicate that the effectiveness of these feature selection methods exhibits significant differences on all the three datasets. It also shows the noise and the dataset types have little impact on the conclusions. Generally, the filter-based and wrapper-based feature subset evaluation methods can achieve the best performance. However, these methods tend to select more features or spend more time. The clustering-based method and most of the filter-based feature ranking methods can achieve acceptable results with fewer features and less time. These methods are also easy to understand. Thus, this empirical study can offer a valuable guideline for practitioner to select appropriate feature selection methods based on some additional criteria, such as computational overhead and simplicity.

In the future, we plan to employ other classifiers to validate the generalization of the derived conclusions and explore the interactions between the classifiers used for the wrapper-based methods and for the defect prediction.

Acknowledgment

This work is partly supported by National Natural Science Foundation of China (NSFC) (grant No. 61572374, U1135005, 61472318), and National Science Foundation (DGE-1522883, CCF-1500365).

REFERENCES

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell. A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, 38(6): 1276-1304, 2012.

[2] T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, 33(1): 2-13, 2007.

[3] M. Shepperd and G. Kadoda. Comparing software prediction techniques using simulation. *IEEE Transactions on Software Engineering*, 27(11): 1014-1022, 2001.

[4] I. Myrtveit, E. Stensrud, and M Shepperd. Reliability and validity in comparative studies of software prediction models. *IEEE Transactions on Software Engineering*, 31(5): 380-391, 2005.

[5] H. Lu, E. Kocaguneli, and B. Cukic. Defect Prediction between Software Versions with Active Learning and Dimensionality Reduction. In *Proceedings of the 25th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 312-322, 2014.

[6] X. Y. Jing, S. Ying, Z. W. Zhang, S. S. Wu, and J. Liu. Dictionary learning based software defect prediction. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*. ACM, 414-423, 2014.

[7] K. Herzig. Using Pre-Release Test Failures to Build Early Post-Release Defect Prediction Models. In *Proceedings of the 25th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 300-311, 2014.

[8] K. O. Elish and M. O. Elish. Predicting defect-prone software modules using support vector machines. *Journal of Systems and Software*, 81(5): 649-660, 2008.

[9] H. Lu, B. Cukic, and M. Culp. Software defect prediction using semisupervised learning with dimension reduction. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 314-317, 2012.

[10] K. Muthukumaran, A. Rallapalli, and N. L. Murthy. Impact of feature selection techniques on bug prediction models. In *Proceedings of the 8th India Software Engineering Conference(ISEC)*. ACM, 120-129, 2015.

[11] H. Liu and H. Motoda. Feature selection for knowledge discovery and data mining. *Springer Science & Business Media*, 2012.

[12] N. Mittas and L. Angelis. Ranking and clustering software cost estimation models through a multiple comparisons algorithm. *IEEE Transactions on Software Engineering*, 39(4): 537-551, 2013.

[13] H. Khalid, M. Nagappan, E. Shihab, and A. E. Hassan. Prioritizing the devices to test your app on: A case study of android game apps. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering(FSE)*, 610-620, 2014.

[14] H. Liu and R. Setiono. Chi2: Feature selection and discretization of attributes. In *Proceedings of the 7th International Conference on Tools with Artificial Intelligence (ICTAI)*, 388, 1995.

[15] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. The Journal of Machine Learning Research, 3: 1157-1182, 2003.

[16] S. Fong, J. Liang, R. Wong, and M. Ghanavati. A novel feature selection by clustering coefficients of variations. In *Proceedings of the 9th International Conference on Digital Information Management (ICDIM)*. IEEE, 205-213, 2014.

[17] S. Fong, J. Liang, and Y. Zhuang. Improving classification accuracy using Fuzzy Clustering Coefficients of Variations (FCCV) feature selection algorithm. In *Proceedings of the 15th International Symposium on Computational Intelligence and Informatics (CINTI).* IEEE, 147-151, 2014.

[18] M. Wasikowski and X. Chen. Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 22(10): 1388-1400, 2010.

[19] C. H. Yang, C. C. Huang, K. C. Wu, and H. Y. Chang. A novel ga-taguchi-based feature selection method. *In Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning–(IDEAL)*. Springer Berlin Heidelberg, 112-119, 2008.

[20] H. Wang, T. M. Khoshgoftaar, R. Wald, and A. Napolitano. A Study on First Order Statistics-Based Feature Selection Techniques on Software Metric Data, In *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 467-472, 2013.

[21] Q. Gu, Z. Li, and J. Han. Generalized fisher score for feature selection. arXiv preprint arXiv:1202.3725, 2012.

[22] A. Ahmad and L. Dey. A feature selection technique for classificatory analysis. *Pattern Recognition Letters*, 26(1): 43-56, 2005.

[23] T. M. Cover and J. A. Thomas. Elements of information theory. John Wiley & Sons, 2012.

[24] J. R. Quinlan. C4.5: programs for machine learning. Elsevier, 2014.

[25] S. S. kannan and N. Ramaraj. A novel hybrid feature selection via Symmetrical Uncertainty ranking based local memetic search algorithm. *Knowledge-Based Systems*, 23(6):580-585, 2010.

[26] D. N. Reshef, Y. A. Reshef, H. K. Finucane, et al., Detecting Novel Associations in Large Data Sets, *Science*, 334(6062):1518-1524, 2011.

[27] J. B. Kinney and G. S. Atwal. Equitability, mutual information, and the maximal information coefficient. *National Academy of Sciences*, 111(9): 3354-3359, 2014.

[28] I. Kononenko. Estimating attributes: analysis and extensions of RELIEF. *Machine Learning*, 171-182, 1994.

[29] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1): 63-90, 1993.

[30] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, Stanford University, 359-366, 2000.

[31] M. Dash, H. Liu, and H. Motoda. Consistency based feature selection. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 98-109, 2000.

[32] M. Dash and H. Liu. Consistency-based search in feature selection. *Artificial intelligence*, 151(1): 155-176, 2003.

[33] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 338-345, 1995.

[34] K. P. Murphy. Machine learning: a probabilistic perspective. *Mathematics Education Library*, 58(8):27-71, 2012.

[35] W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*. 115-123, 1995.

[36] M. Thangaraj and C. R. Vijayalakshmi. Performance Study on Rule-based Classification Techniques across Multiple Database Relations. *International Journal of Applied Information Systems (IJAIS), Foundation of Computer Science FCS*, New York, USA, 5(4), 2013.

[37] A. Agresti. Building and applying logistic regression models. *Categorical Data Analysis*, Second Edition, 211-266, 2007.

[38] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21-27, 1967.

[39] T. Chai and R. R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3): 1247-1250, 2014.

[40] M. D'Ambros, M. Lanza, and R. Robbes. An extensive comparison of bug prediction approaches. In *Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 31-41, 2010.

[41] L. Breiman. Random forests. *Machine learning*, 45(1): 5-32, 2001.

[42] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4): 485-49, 2008.

[43] X. Xia, D. Lo, E. Shihab, X. Wang, and X. Yang. Elblocker: Predicting blocking bugs with ensemble imbalance learning. *Information and Software Technology*, 61: 93-106, 2015.

[44] X. Xia, D. Lo, S. McIntosh, E. Shihab, and A. E. Hassan. Cross-project build co-change prediction. In *Proceedings of the 22nd International Conference on. Software Analysis, Evolution and Reengineering (SANER)*, 311-320, 2015.

[45] H. V. Garcia and E. Shihab. Characterizing and predicting blocking bugs in open source projects. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*. ACM, 72-81, 2014.

[46] S. McIntosh, B. Adams, M. Nagappan, and A. Hassan. Mining co-change information to understand when build changes are necessary. In *Proceedings of the 30th International Conference Software Maintenance and Evolution (ICSME)*. IEEE, 241-250, 2014.

[47] G. A. Liebchen and M. Shepperd. Data sets and data quality in software engineering. International Workshop on Predictor Models in Software Engineering, 39-44, 2008.

[48] http://www.exploredata.net/.

[49] T. M. Khoshgoftaar, M. Golawala, and J. V. Hulse. An empirical study of learning from imbalanced data using random forest. In *Proceedings of the 19th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2: 310-317, 2007.

[50] Fawcett T. An introduction to ROC analysis. *Pattern recognition letters*, 27(8): 861-874, 2006.

[51] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. ACM, 233-240, 2006.

[52] J. Chen, S. Liu, W. Liu, X. Chen, Q. Gu, and D. Chen. A two-stage data preprocessing approach for software defect prediction. In *Proceedings of the 8th International Conference Software Security and Reliability (SERE)*. IEEE, 20-29, 2014.

[53] S. Liu, X. Chen, W. Liu, X. Chen, Q. Gu, and D. Chen. FECAR: A feature selection framework for software defect prediction. In *Proceedings of the 38th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 426-435, 2014.

[54] Z. Xu, J. Xuan, J. Liu, and X. Cui. MICHAC: Defect Prediction via Feature Selection based on Maximal Information Coefficient with Hierarchical Agglomerative Clustering. In *Proceedings of the 23rd International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Page to appear, 2016.

[55] J. Huang, Y. Cai, and X. Xu. A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recognition Letters*, 28(13):1825-1844, 2007.

[56] M. L. Zepeda-Mendoza and O. Resendis-Antonio. Hierarchical agglomerative clustering. *Encyclopedia of Systems Biology*. Springer New York, 886-887, 2013.

[57] D. Cordes, V. Haughton, J. D. Carew, K. Arfanakis, and K. Maravilla. Hierarchical clustering to measure connectivity in fMRI resting-state data. Magnetic resonance imaging, 20(4): 305-317, 2002.

[58] R. Bro and A. K. Smilde. Principal component analysis. *Analytical Methods*, 6(9): 2812-2831, 2014.

[59] M. H. Halstead. Elements of Software Science. New York: Elsevier North Holland, 1977.

[60] T. J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 308-320, 1976.

[61] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine learning*, 42(3): 203-231, 2001.

[62] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu. A general software defect-proneness prediction framework. *IEEE Transactions on Software Engineering*, 37(3): 356-370, 2011.

[63] G. W. Corder and D. I. Foreman. Nonparametric statistics: a step-by-step approach. John Wiley & Sons, 2014.

[64] B. Ghotra, S. McIntosh, and A. E. Hassan. Revisiting the impact of classification techniques on the performance of defect prediction models. In *Proceedings of the 37th International Conference on Software Engineering (ICSE)*. IEEE, 789-800, 2015.

[65] T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, 33(1): 2-13, 2007.

[66] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener. Defect prediction from static code features: current results, limitations, new approaches. *Automated Software Engineering*, 17: 375-407, 2010.

[67] A. Tosun and A. Bener. Reducing false alarms in software defect prediction by decision threshold optimization. In *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement(ESEM)*. IEEE, 477-480, 2009.

[68] M. Shepperd, Q. Song, Z. Sun, and C. Mair. Data Quality: Some Comments on the NASA Software Defect Datasets. *IEEE Transactions on Software Engineering*, 39(9):1208-1215, 2013.

[69] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto. Automated parameter optimization of classification techniques for defect prediction models. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, page to appear, 2016.

[70] J. Nam and S. Kim. Heterogeneous defect prediction. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering(FSE)*. ACM, 508-519, 2015.

[71] A. J. Scott and M. Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 507-512, 1974.

[72] L. C. Borges and D. F. Ferreira. Power and type I errors rate of Scott-Knott, Tukey and Newman-Keuls tests under normal and no-normal distributions of the residues. *Revista de Matemática e Estatística*, 21(1): 67-83, 2003.

[73] S. Shivaji, E. J. Whitehead, R. Akella, and S. Kim. Reducing features to improve code change-based bug prediction. *IEEE Transactions on Software Engineering*, 39(4): 552-569, 2013.

[74] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya. Choosing software metrics for defect prediction: an investigation on feature selection techniques. *Software: Practice and Experience*, 41(5): 579-606, 2011.

[75] H. Wang, T. M. Khoshgoftaar, J. V. Hulse, and K. Gao. Metric selection for software defect prediction. *International Journal of Software Engineering and Knowledge Engineering*, 21(02): 237-257, 2011.