Software Defect Prediction Based on Feature Subset Selection and Ensemble Classification

Ahmad A. Saifan[1] and **Lina A. Abuwardih**[2]

ABSTRACT

This research highlights a procedure which includes a feature selection technique to single out relevant attributes, and an ensemble technique to handle the class-imbalance issue. In this research we look at two potential scenarios: (1) Ensemble models constructed from the original datasets, without feature selection, and (2) Ensemble models constructed from the reduced datasets after feature selection has been applied. Four feature selection techniques are employed: Principal Component Analysis (PCA), Pearson's correlation, Greedy Stepwise Forward selection, and Information Gain (IG). The aim of this research is to assess the effectiveness of feature selection techniques using ensemble techniques. Five datasets, obtained from the PROMISE software depository, are analysed. Tentative results indicate that ensemble methods can improve the model's performance without the use of feature selection techniques. Based on the area under curve (AUC) performance measurement PCA feature selection and bagging based on K-NN perform better than both bagging based on SVM and boosting based on K-NN and SVM. The AUC values based on K-NN with PCA enhanced for three datasets CM1, KC3, MC2. The AUC values for the three datasets are 0.726, 0.775, and 0.750 respectively. Feature selection techniques, including Pearson's correlation, Greedy Stepwise, and IG, weaken the ensemble models' performance.

Keywords: Software Engineering, Software Defect Prediction, Feature Selection, PCA, Bagging, Boosting

1. INTRODUCTION

Introducing a software system that is free from defects is a crucial aspect in the software engineering domain. Software defect prediction techniques aim to classify the defective portions of a software system prior to its release. Timely prediction of software defects permits software project managers to efficiently manage people, cost, and time, which will boost the software's quality. The existence of defects in software causes a reduction in quality which can result in the failure of a software project [1]. Software quality is measured using various software metrics including depth of inheritance tree (DIT), lines of code (LOC), number of children (NOC), and many more [2]. Selecting which software metrics are helpful in predicting the defects in software portions is an important issue, and different studies have been conducted to date to handle this issue such as [3, 4, 5]. Other studies concentrated on using machine learning mechanisms such as classification [6], clustering [7] and association rules mining [8] to identify the defective parts in the software system.

The classification technique is one type of machine learning mechanism that is employed in software defect prediction. It aims to categorize the parts of the software that include defects. Classification consists of two phases: model construction from training data, and model evaluation using testing data. Various forms of classification have been utilized in the software defect prediction field, including support vector machines [9], Bayesian Networks [10], Decision Trees [11], and Naive-Bayes [12]. Despite this diversity and availability of classification methods, software defect prediction is considered an unsolved issue. The results, based on comparisons and benchmarking of the software defect prediction that utilizes classification, show that no one model is suitable for all datasets [13,14]. An accurate defect prediction model is a major requirement for a large-scale software system.

Two major issues influence a model's performance: class imbalance distribution, and non-relevant attributes within the dataset that represent a noise. These result in weakening the performance of the classifier [15].

A feature selection method is applied when the learning process involves multi-dimensional datasets [16]. The ensemble classification method is applied when there is imbalanced class distribution within the datasets [17]. Imbalanced class distribution means that the quantum of non-defective software portions exceeds the number of defective ones. In this research, we examine the problem of feature subset selection by employing four feature subset selection methods: principal component analysis (PCA), Pearson's correlation, Greedy Stepwise Forward selection, and Information Gain (IG) to characterize the important features used in model construction.

Bagging and Adaboosting ensemble classification

Manuscript received on November 7, 2019 ; revised on March 28, 2020.

Final manuscript received on June 20,2020.

[1,2]The authors are with IS department, Faculty of IT, Yarmouk university, Jordan., E-mail: ahmads@yu.edu.jo and 2013930022@ses.yu.edu.jo

DOI: 10.37936/ecti-cit.2020142.224489

are also used for the purpose of defect prediction. These algorithms are selected because they work well with imbalanced data [18].

This research examines feature subset selection for software defect prediction based on ensemble techniques, with the aim of answering the following questions:

1. Does feature subset selection have an impact on software defect predictors?

2. Does feature subset selection, coupled with ensemble classification, perform well in software defect predictors?

Software systems are now embedded in many areas of our lives, and range from trivial to critical systems that have a great impact on us. High-quality software systems must be free from defects and operate as intended, as defects will result in exhausting a project's resources including time, cost, and effort. Thus there is increasing interest in the early detection of defects through software defect prediction (SDP) models [19].

A software defect is a fault, weakness, or deviation in the software product or in the software development process [20]. In order to maintain control of the overall development process, the quality of the software must be measured. This measurement is expressed in terms of software metrics [21, 22, 23, 24].

A software defect prediction model generally depends on machine learning, and consists of two essential phases. In the first phase, the model is constructed from a combination of training instances. These instances come from historical software datasets in which every instance acts as a system, class, function or method, and is labeled by a flag that represents its status such as clean, buggy, or has a number of bugs. The software metrics represent the features or attributes of the generated instances. In the second phase, after the model is constructed, new unknown instances are provided to validate the produced model [25]. Figure 1, shows this scenario in more detail.

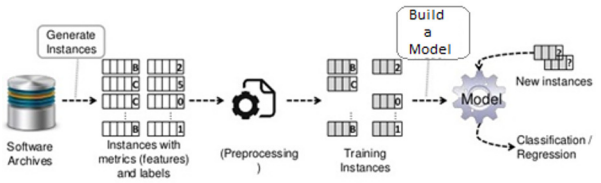

Fig.1: *The General Process of the Software Defect Prediction Model [25].*

This research is explained step by step. The first section looks at previous studies which examined feature selection and ensemble classification for software defect prediction. This is followed by a description of the methodology used in our experiments. Then the experiments and evaluation section shows the steps followed during the experiment, together with the results of the evaluation process. Finally, we present the conclusion and recommendations for future work.

2. RELATED WORK

Feature selection is considered a critical step in machine learning and has been widely applied in different domains including Text Classification [26], Biomedical Science [27], and Education [28]. Features selection is also utilized in the software engineering domain for software defect prediction. It is known that various software metrics may be related to defective parts in software systems, therefore mitigating those related metrics could improve a model's performance. Chandrashekar and Sahin [29] and Molina et al. [30] investigated FSS in machine learning, while Rathore and Gupta [27] undertook a study which compared feature-ranking and feature-subset selection mechanisms for improving fault prediction. Ensemble classification is a solution introduced to handle small-sized and imbalanced datasets [31]. The ensemble classifier is a robust classifier that is constructed from a set of trivial classifiers and works better than using a single classifier alone. Ensemble learning has been employed successfully in different domains including Biomedical Science [32], Education [33], and Concept Drift [34]. Woźniak et al. [35] undertook a study of different ensemble methods. The following sections present research that has been carried out in software defect prediction according to feature subset selection and ensemble classification.

2.1 Software Defect Prediction Depending on Feature Subset Selection

In a study by Khoshgoftaar et al. [36], four constructed models and their performances were compared. These models were inducted to handle different situations:

- Feature selection and model construction from genuine data.
- Feature selection from genuine data and model construction from observed data.
- Feature selection using observed data and model construction from genuine data.
- Feature selection using observed data and model construction from observed data.

Six filter-based features ranking mechanisms were applied: CS, IG, GR, RF, RFW, and SU. For classification purposes, K-NN and SVM classifiers were employed, and experiments were conducted using nine datasets from the PROMISE depository. The results showed that the performance of the model based on feature selection using both observed and genuine data was better than the other models. Also, there was no difference in performance between using genuine or observed data.

Khoshgoftaar et al. [37] contrasted seven filter-based feature ranking mechanisms on sixteen differ-

ent software datasets using three diverse actual software projects. They incorporated four datasets from an extremely vast telecommunications software system (LLTS), nine datasets from the Eclipse project, and three data sets from a NASA software project. They constructed the classification models using five classifiers: Naive Bayes, multilayer perceptron, K-nearest-neighbors, support vector machine, and logistic regression on the Weka environment. The evaluation was performed using area under the ROC curve (AUC) and best arithmetic mean (BAM) performance metrics. They used filter-based feature ranking mechanisms similar to the techniques used by Gao et al. [5], which included chi-squared (CS), information gain (IG), gain ratio (GR), symmetrical uncertainty (SU), and ReliefF (two types: RF and RFW). Khoshgoftaar et al. [37] also factored in the signal to noise ratio (SNR) that was not used by Gao et al. [5]. The results showed that the IG was the superior filter among all datasets. CS and SU were somewhat inferior to IG but exceeded GR, RF, and RFW. Also, SNR and IG provided a more settled performance than RF and RFW regarding the different classifiers and datasets. Their work was utilized on a fixed size selected feature set and did not optimize the size of the selected subset features that could improve the classification performance.

Murillo-Morera et al. [38] presented a framework for employing a genetic algorithm as a wrapper method for attribute selection as did Vivanco et al. [39] for predicting the proneness of defects in a software module. In addition, three empirical studies that utilized the genetic defect prediction framework were conducted. The first was a comparison between framework performance and the most popular framework in the literature. The second was a comparison of framework performance and run-time with the exhaustive framework. The last empirical study was a sensitivity analysis. The experiments were performed on seventeen datasets obtained from the NASA-MDP and PROMISE projects. The model was tested by NxM cross-validation. The results showed that: 1) the performance of the proposed framework was the same as the most popular framework, 2) the framework run-time did better than the exhaustive framework, and 3) the configuration regarding the sensitivity analysis was improved. Their work did not consider diversity metrics regarding performance such as precision, recall, and F-measure, or the diversity of the learning schemes.

Jia [40] proposed a hybrid feature selection method to detect defects in software, using chi-squared (cs), Information gain (IG), and Pearson Correlation coefficients. In order to build the model, Jia used a random forest classifier approach testing her model on five datasets from NASA. To evaluate the performance of the defect prediction model, she used AUC measure. Arora and Saha [41] proposed a software defect prediction model based on two classifiers, extreme learning machine (ELM), and kernel based extreme learning machine (KELM). In their approach they used five wrapper- and seven filter-based feature selection methods. To test their approach, they used seven datasets collected from the PROMISE repository. Moreover, they used the accuracy measure to evaluate the performance feature selection model. They found that ELM-based classifiers achieved a higher testing accuracy with wrapper-based feature selection methods, while KELM classifiers performed better with filter-based methods.

2.2 Software Defect Prediction Depending on Ensemble Models

A comparative study of seven ensemble methods including bagging, boosting, random trees, random forest, random subspace, stacking, and voting was conducted by Tao et al. [42] on 14 different NASA datasets. For evaluation purposes, 10-fold cross validation was applied. The performance of each of the different ensemble models was compared to the Naive Bayes model. The results showed that the model's performance was better when an ensemble model was applied than when using a single model. In addition, for the generated ensemble model, the voting and random forest methods outperformed the others. The work did not vary the base models in order to prove whether the results were affected by base models or not.

Jayaraj and Raman [43] utilized three boosting techniques: boosting with decision stump, boosting with REPtree, and boosting with M5. The experiment was conducted on the KC1 dataset using the Weka environment. For evaluation of the performance of the three boosting methods, the accuracy, root mean squared error, and mean absolute error performance measures were used. The results discovered that, of the three boosting methods, the boosting with M5 method had the best accuracy. The study was conducted on one dataset so the results could not be generalized.

Wahono and Herman [44] introduced a novel software defect predictor that depended on integration of a the genetic algorithm and the bagging technique. GA was utilized to choose the optimal and pertinent feature subset. The bagging technique was utilized to deal with the imbalance class distribution issue. Two types of experiment were performed on nine NASA datasets. The first experiment was performed on nine NASA datasets using ten classification algorithms. The second experiment used nine NASA datasets implementing the GA and the bagging technique on ten classification algorithms. The results showed that the model's performance was improved with integration between the GA and bagging methods and that it performed better than the original method (implementing single model without the integration with

the FSS).

Also, Laradji et al. [45] carried out a study to expose the effectiveness of integration between FSS and ensemble methods on software defect predictor performance. They proposed two types of ensemble model (with and without selected features) depending on the average probability ensemble (APE) method. Different feature selection techniques, including Pearsons' correlation, Fishers' criterion, and the GFS method, were used to identify the best method for improving model performance. The experiments were conducted on six datasets from the PROMISE depository. The results revealed that greedy forward selection obtained better outcomes than correlation-based forward selection. When using the average probability ensemble (APE), the model performance was enhanced and it performed better than the traditional models such as weighted SVMs and random forests. With integration of the GFS and APE ensemble methods, the AUC measures for the used datasets were close to 1.0. The study lacked a comparison between the proposed method and the existing ensemble methods which could have supported the outcomes.

A study by Petrić et al. [17] to show the improvement of the defect prediction model used an explicit diversity method with stacking ensemble. Eight datasets from the PROMISE depository were used to construct four different base models: Naive Bayes, C4.5 decision tree, K-nearest neighbor, and sequential minimal optimization. The diversity concept was based on the idea that when constructing the ensemble models, diversity should guarantee incorporating only models that produce faults in various instances, not in the same instances. For the comparison between two models at a time, weighted precision and diversity were utilized. Stacking ensembles were constructed using base measure, precision measure, Matthew correlation coefficient measure, diversity measure, and weighted accuracy diversity measure. In order to evaluate the models, a comparison was made between the proposed approach with bagging and boosting ensemble methods by using the same measures used in the stacking ensemble. The results revealed that prediction performance with the proposed diversity stacking ensemble was better than single, bagging, and boosting models. The study did not clarify the influence of various performance measures on the accurate classification of faulty instances.

Punitha and Latha [15] proposed a new approach for the detection and prevention of software defects that would solve the problem of imbalanced datasets. This involved using a genetic algorithm alongside Ant Colony Optimization (GACO) and a bagging mechanism to single out the appropriate feature. This was followed by a sampling technique which incorporated a semi-supervised learning technique to create the balanced label from the imbalanced datasets. After the balanced label was generated, Hybrid Neuro-Fuzzy Systems with the Naive Bayes method were employed. The experiments were performed on four NASA MDP datasets. Precision, recall, accuracy, and F-measure performance measures were used for performance evaluation of the proposed model. The performance of the proposed approach was compared with the hybrid BmSVM method. The results revealed that the proposed hybrid learning approach for defect prediction was very effective.

Abdou and Darwish [46] tested three types of ensemble learner models, boosting, bagging, and rotation forest, to predict defects in software. They used seven datasets collected from the PROMISE repository. Moreover, they used the accuracy measure to evaluate the performance of the models. They found that accuracy improved more when using ensemble techniques than when using only single learners, especially in conjunction with rotation forest with the resample technique that was used in most of the algorithms used in the experimental results.

Alsaeedi and Khan [47] compared different machine learning technologies with ensemble classifiers, (bagging, support vector machines, decision tree, and random forest classifiers) in order to check their performance in software fault prediction. They used ten datasets from NASA, and employed accuracy, F-measure, and ROC-AUC metrics to evaluate the performance of the different classifiers. The results showed that random forest performed better than the others.

We can conclude from the reviewed literature that no one model or technique is considered the best in the software defect prediction field, as the best prediction depends on many factors including dataset size, programming language, and the features used.

3. RESEARCH METHODOLOGY

The research described in the document was conducted using a series of experiments to study the performance of ensemble models depending on the use of features selection. Experiments were conducted on five benchmark datasets obtained from the tera-PROMISE depository [48]. These datasets were from NASA MDP software projects which were written in C, C++, and Java programming languages. Table 1 shows the selected datasets.

Project CM1 describes a NASA spacecraft instrument system written in C, consisting of 498 functions, 449 of which are defective (with at least one defect) and 49 are non-defective. KC1 is a "C++" system implementing storage management for receiving and processing ground data consisting of 2109 functions, 326 of which are defective (with at least one defect) and 1783 are non-defective. KC3 is another version of KC written in Java consisting of 194 functions, 36 of which are defective (with at least one defect) and 158 are non-defective. MC2 is a Video Guidance System implemented in C, consisting of 125 functions,

Table 1: *NASA MDP Datasets Description.*

Datasets	Language	♯of Instances	♯ of Attributes	♯ of Non-Defective	♯of Defective	Imbalanced Ratio	% Defect
CM1	C	498	38	449	49	9.16	9.83
KC1	C++	2109	22	1783	326	5.46	15.45
KC3	Java	194	40	158	36	4.38	18.55
MC2	C	125	40	81	44	1.84	35.2
PC1	C	1109	22	1032	77	13.40	6.94

Table 2: *Detailed Information of Software Metrics.*

Category	Software Metrics	Definition	Description
McCabe	LOC	Line of code	The number of code lines
	v(g)	Cyclomatic complexity	The number of linearly independent paths
	ev(g)	Essential complexity	The scope to which a flow graph can be reduced
	iv(g)	Design complexity	The cyclomatic complexity of a modules reduced flow graph
Base Halstead	mul	Unique operators	The number of unique operators
	mu2	Unique operators	The number of unique operators
	N1	Operators occurrences	Total occurrences of operators
	N2	Operators occurrences	Total occurrences of operands
	N	Length	N1+N2
	mu	Vocabulary	mu1+mu2
Derived Halstead	P	Volume	N * log2(mu)
	V*	Volume on minimal implementation	(2 + mu2)*log2(2 + mu2)
	L	Program length	V*/N
	D	Difficulty	1/L
	E	Effort to write program	V/L

44 of which are defective (with at least one defect) and 81 are non-defective. PC1 is flight software for earth orbiting satellites written in C, consisting of 1109 functions, 77 of which are defective (with at least one defect) and 1032 are non-defective. These datasets are derived from different projects, and the data distributions as well as feature attributes are different. We chose the common attributes (metrics) of the dataset to build the prediction models. Table 2 shows the metrics of the software features used in the experiment.

The appropriate selection of suitable software metrics is a serious issue in software defect prediction, since the performance of a model is affected by the software metrics selected. In software defect prediction, the features are the software metrics that are extracted from the static source code. Figure 2 presents an example of static code attributes. The process in the selection of feature subsets is based on choosing the best metrics and discarding the worst ones which will have a negative impact on the classifier performance.

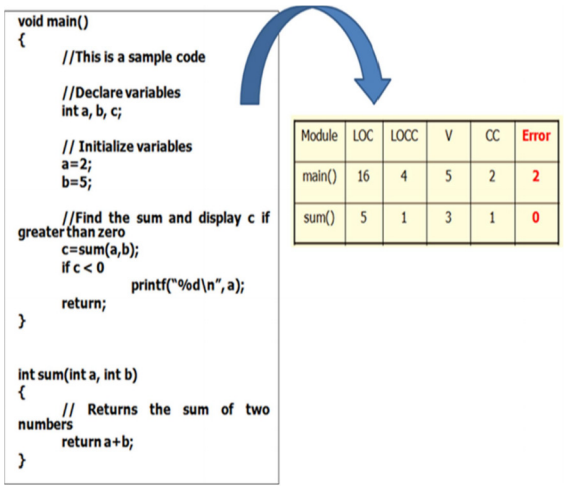

Fig.2: *Software metrics extracted from the static source code.*

In this research, we employ four different feature subset selection methods to evaluate their effectiveness on the performance of the constructed models [49]: Information Gain, Greedy Stepwise forward selection, Correlation Evaluation based on Pearson's correlation measure, and Principal Component Analysis. The first three methods are regularly used in the SDP literature, whereas Principal Component Analysis (PCA) is rarely employed. Figure 3 summarizes the research methodology.

Four-feature subset selection is used in this research with more details as follows:

- In Principal Component Analysis, a combination of features is minimized by enforcing a modification of the data. PCA adopts adequate eigenvectors to compute a proportion of the variance in the genuine data (default is 95%). The noise attribute can be cleared by changing to the PC space, minimizing some of the poorest eigenvectors, and then reverting back to the genuine space. The result is a diminutive set of ranked features.

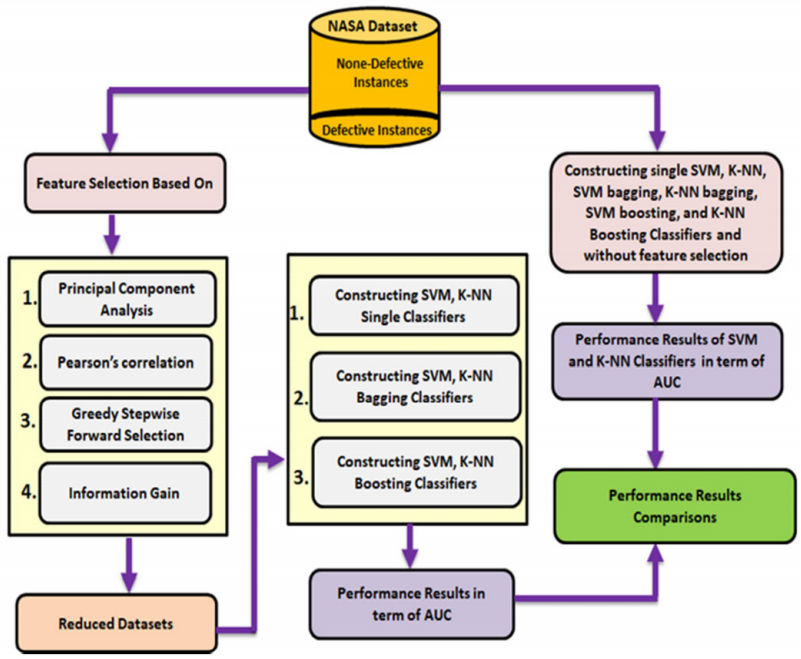

Fig.3: *The proposed research methodology.*

- Correlation Evaluation assesses attribute interestingness by gauging the Pearson's correlation between each attribute and the class.
- Greedy Stepwise tries to find the best choice by selecting the preferable features and discarding the worst ones at every step, and the process is refined until the stopping condition is satisfied. It enforces a greedy forward or backward search over all of the feature subsets. It can be launched without attributes, with all attributes, or from a specific location in the space, and is halted when the insertion/removing of any residual attributes brings a lessening to the evaluation.
- Information Gain is the information offered about the objective class attribute Y, presuming the value of attribute X. IG gauges the alleviation of the weighted average infection of the divisions compared to the infection of the complete dataset.

This research applies supervised learning methods that have good results in SDP. Ensemble techniques are used based on boosting and bagging algorithms. Two base classifiers are used with AdaBoostM1 and bagging algorithms from various families: Lazy (K-NN) [49] and Support Vector Machine (SVM) [50]. These algorithms are chosen for two reasons: first, they do not include an embedded feature selection and second, they are typically employed in software engineering and data mining areas [37].

Two classification algorithms are used in this research as follows:

- Support Vector Machine (SVM) identifies the decision planes to clarify decision limits. A decision plane is split between groups of instances with several class labels. SVM carries out the classification task by introducing a hyperplane in a high-dimensional space which distinguishes instances of several classes [51]. SVM does very well in a high dimensional space and in the binary classification problem. The SVM algorithm is employed for the five datasets without feature selection, and for the five datasets using the four-feature subset selection methods presented above. Also, SVM serves as a base classifier for constructing boosting and bagging classifiers.
- K-Nearest Neighbors (K-NN) is a case-based, passive learning algorithm. Case-based algorithms employ only the learning data without generating statistics to standardize their assumptions. The K-NN learner fulfills this by computing the distance of the inspected instance from each learning instance, and the predicted class is acquired from the K nearest neighbors [52]. The K-NN algorithm is employed for the five datasets without feature selection and is also applied to the five datasets using the four feature subset selection methods presented above. Also, K-NN serves as a base classifier for constructing boosting and bagging classifiers.
- Bagging is introduced by Breiman [53] to enhance classification accuracy by grouping classifications of arbitrarily created training sets. For the training set D of size n, bagging produces m new training sets Di, each of size ń, by selecting samples with replacement from

D evenly. Using sampling with replacement, some instances may be replicated in each Di. If ń=n, then for large n the set Di is predicted to have 63.2% of the individual instances of D. This type of sample is recognized as a bootstrap sample. The m models are adjusted using the above m bootstrap samples and assembled by averaging the outcome (for regression) or voting (for classification). The base classifiers used for the bagging algorithm are SVM and K-NN.

- For Boosting we use the AdaBoost algorithm, which is proposed by Freund et al. [54] to construct an ensemble classifier using the selected features. The AdaBoost algorithm is the most common algorithm for the boosting method, combining various weak classifiers into a single, high performance, rigorous classifier. The base classifiers used with AdaBoost algorithm are SVM and K-NN.

There are two reasons for using two types of classifier as base classifiers in the ensemble models. First, we want to maintain diversity as each classifier behaves in a different way in dealing and constructing models. Second, each one has advantages and disadvantages, so using both will utilize the pros and avoid the cons. For classifier validation, we use 10-fold cross-validation [55]. The 10-fold cross validation method is used to determine the size of training and testing datasets. The dataset is divided into 10 groups. For each group we take one group as a hold out, or test, data set. The remaining groups serve as a training data set. We use 10-fold cross-validation because this method is typical in the software defect prediction field.

In order to evaluate and compare the above ensemble algorithms, we use AUC (area under curve) measure for the experiments. AUC is the area under the receiver operating characteristic (ROC) curve. The x-axis is the false positive rate, and the y-axis is the true positive rate. The false positive rate (FPR) gauges how many non-defective instances are predicted as defective among all non-defective instances. The true positive rate (TPR) gauges how many defective instances are predicted as defective among all defective instances. The ROC curve value lies in the [0, 1] interval [12]. The ROC curve can distinguish the trade-off between the true positive rate and the false positive rate. A classifier with a greater area under the curve is favored over a classifier with a smaller area under the curve. Lessmann et al. [6] favored the employing of AUC since it enhances the comparisons between the experiments. The AUC is employed widely in empirical software defect prediction experiments because it recognizes the prediction performance for all possible threshold values, and it acts as a general and stable measure to compare several prediction models [36].

The Weka 3.8.1 tool is used to carry out the various feature subset selection methods and to perform the SVM, K-NN, bagging, and boosting algorithms. Weka is a java environment that implements a set of machine learning algorithms for data mining purposes [56].

In addition, IBM® SPSS® statistics 19 is used to address the bivariate statistics with ANOVA as it is a thorough system for resolving data. SPSS Statistics deals with data from almost any paradigm and utilizes it to produce tabulated reports, charts, plots of distributions, descriptive statistics, and knotted statistical analyses [57].

4. EXPERIMENTS AND EVALUATION

The experiments were carried out on five public datasets from PROMISE, a depository which includes many public software engineering datasets. We were interested in software defect datasets, obtained from NASA projects, which were chosen during the development of the software and are for software written in the C, C++, and Java programming languages. These datasets are used for research in the software defect prediction field and are available for free download on **http://openscience.us/repo/defect/mccabehalsted**. The five selected datasets included different numbers of features ranging from 40 features found in the KC3 and MC2 datasets, to 22 features found in the KC1 dataset. The number of instances varied from 125 found in the MC2 dataset, which was the smallest one, to 2109 instances found in the KC1 dataset, which was the largest one. Not all features are relevant. Some may lead to inappropriate results which may affect the model prediction negatively. The attributes on all of the datasets were of numerical type, and the class labels were nominal. In this research, we used the Weka tool to apply the feature selection mechanisms to the five chosen datasets. These methods included PCA, Pearson correlation, GS, and IG.

The intent of the experiments was to assess the effectiveness of feature subset selection mechanisms coupled with ensemble methods. This assessment was based on two main scenarios as follows:

Scenario 1: Two single, two bagging, and two boosting classifiers were constructed from the original datasets using SVM and K-NN algorithms and without applying any feature subset selection. The performance results of the classifiers in terms of AUC were computed.

Scenario 2: Four feature selection mechanisms were utilized with the original datasets. The results were sets of reduced datasets. Two single, two bagging, and two boosting classifiers were constructed from the reduced datasets using SVM and K-NN algorithms. The performance results of the classifiers in terms of AUC were computed.

4.1 Without Feature Subset Selection

In this experiment, a set of single and ensemble classifiers was constructed without applying any feature subset selection techniques, and then the AUC was computed for each type of classifier.

- Single Classifier: For the five datasets, two single classifiers were constructed based on SVM and K-NN algorithms from the original datasets, without applying any feature subset selection methods. The AUC performance measures were computed and are presented in Table 3.
- Bagging Classifier: Two bagging ensemble classifiers were constructed based on the bagging algorithm using SVM and K-NN as base classifiers, without applying any feature subset selection methods. The AUC performance measurements were computed and presented in Table 3.
- AdaBoost Classifier: Two boosting ensemble classifiers were constructed based on the AdaBoostM1 algorithm using SVM and K-NN as base classifiers, without applying any feature subset selection methods. The AUC performance measurements were computed and are exhibited in Table 3.

Table 3: *AUC of SVM and K-NN as single, bagging, and boosting classifiers on five datasets - Without Feature Subset Selection.*

dataset	Single Classifier		Bagging Classifier		Boosting Classifier	
	SVM	K-NN	SVM	K-NN	SVM	K-NN
CM1	0.716	0.660	**0.731**	**0.697**	0.706	**0.692**
KC1	0.681	0.793	**0.687**	0.787	**0.725**	0.645
KC3	0.683	0.660	**0.726**	**0.717**	**0.684**	**0.693**
MC2	0.660	0.694	**0.675**	**0.705**	**0.673**	0.666
PC1	0.823	0.828	**0.843**	**0.850**	**0.831**	0.824

Table 3 shows the AUC values for the bagging and boosting classifiers using SVM and K-NN compared with the single classifier using SVM and K-NN without feature subset selection for different datasets. The bold values mean that the AUC value for bagging and boosting with base SVM and K-NN is better than those for the single classifier with base SVM or K-NN. For example, using the dataset CM1, the AUC values for bagging and boosting classifiers with base K-NN are 0.697 and 0.692 respectively. These are greater than the AUC value for the single classifier with base K-NN 0.660.

Regarding the bagging and boosting classifiers (SVM and K-NN), there is an improvement in the AUC values compared with SVM and K-NN as single classifiers. The use of ensemble methods without feature subset selection can improve the performance of the AUC classifiers for most datasets.

4.2 With feature subset selection

In this experiment, a set of single and ensemble classifiers was constructed after applying the four types of feature subset selection techniques. Then the AUC was computed for each type of classifier.

The number of features selected after applying the four types of feature selection including the class label is shown in Table 4.

Table 4: *The Number of Selected Features on the Five Datasets.*

Datasets	Feature subset selection			
	PCA	PC	GS	IG
CM1	12	7	6	7
KC1	8	6	9	6
KC3	11	7	9	6
MC2	12	7	11	7
PC1	13	7	9	7
Average	11.2	6.8	8.8	6.6

Table 4 shows the number of features selected by applying the four-feature selection mechanism. The differences in the number of chosen features depend on a number of factors such as: original feature size (datasets with a large number of features will lead to a more greatly reduced set of features when the filters are applied than those with a fewer features), number of defective instances, overall number of instances, and the programming languages used during the development process. All of these can have an impact on which features are selected by each filter.

Table 4 shows the number of attributes selected by each of the feature subset selection methods. PC and IG produce fewer numbers of selected features. PCA and GS, on the other hand, tend to select a much wider range of features to provide comparable classification results.

In the Principal Component Analysis PCA was employed to identify the reduced set of features.

- Single Classifier: For the five datasets, two single classifiers were constructed based on SVM and K-NN from the reduced datasets by PCA. The AUC performance measurements were computed and are exhibited in Table 5.
- Bagging Classifier: Two bagging ensemble classifiers were constructed based on the bagging algorithm using SVM and K-NN base classifiers from the reduced datasets by PCA. The AUC performance measurements were computed and are exhibited in Table 5.
- AdaBoost Classifier: Two ensemble classifiers were constructed based on the AdaBoostM1 algorithm and using SVM and K-NN base classifiers from the reduced datasets by PCA. The AUC performance measurements were computed and are exhibited in Table 5.

Table 5 shows that PCA feature subset selection decreased the AUC of SVM and K-NN as a single classifier for more of the datasets than AUC of SVM

Table 5: *: AUC of SVM and K-NN as single, bagging, and boosting classifiers on five datasets - With PCA Feature Subset Selection.*

Dataset	Single Classifier		Bagging Classifier		Boosting Classifier	
	SVM	K-NN	SVM	K-NN	SVM	K-NN
CM1	0.698	0.657	0.715	**0.726**	0.698	0.639
KC1	0.672	0.693	0.676	0.655	**0.732**	0.530
KC3	0.583	**0.785**	0.609	**0.775**	0.551	0.648
MC2	**0.698**	**0.721**	**0.709**	**0.750**	**0.708**	0.563
PC1	0.797	0.787	0.823	0.785	0.763	0.647

and K-NN classifiers without feature subset selection. The values in bold indicate that the use of the feature subset selection method leads to an improvement of the AUC value, in comparison with when no feature selection is used. This was compared to methods without feature selection (Table 3), the ensemble methods, more specifically K-NN bagging based on PCA feature subset selection improved the AUC over three datasets (CM1, KC3 and MC2). SVM boosting based on PCA feature subset selection improved the AUC over two datasets (KC1 and MC2). SVM bagging based on PCA feature subset selection improved the AUC only for the MC2 dataset. K-NN boosting AUCs decreased in all the datasets comparison with K-NN boosting classifiers without feature subset selection. Moreover, we observe that the simple classifiers and ensemble methods for feature selection do not bring any improvement in accuracy for the dataset PC1. The reason for that may be that the PCA was not able to identify the dependencies between the features. So it is clear that the performance of PCA feature subset selection depends largely on the dataset. The best subset of features varied from one dataset to another (see Table 4).

In the Pearson's Correlation experiment, the Pearson's measure was employed to identify the combination of features that have correlation with the class. This resulted in a group of ranked features. We used log2 n, where n is the number of independent features in the genuine dataset, as a way to select the top ranked features (i.e. the relevant features) over the five datasets. The related literature does not define any specific direction on the proper number of features to adopt. Therefore, we used the same technique as [58].

- Single Classifier: For the five datasets, two single classifiers were constructed based on SVM and K-NN classifiers from the selected features by Pearson's correlation datasets. The AUC performance measures were computed and are exhibited in Table 6.
- Bagging Classifier: Two bagging ensemble classifiers were constructed based on the bagging algorithm using SVM and K-NN base classifiers based on the reduced datasets by Pearson's correlation. The AUC performance measurements were computed and are exhibited in Table 6.
- AdaBoost Classifier: Two boosting ensemble classifiers were constructed based on the AdaBoostM1 algorithm and using SVM and K-NN base classifiers based on the reduced datasets found by using Pearson's correlation. The AUC performance measurements were computed and are exhibited in Table 6.

Table 6 shows the AUC for Pearson's correlation (PC) feature subset selection for the selected classifiers. Again, the values in bold indicate that the use of the feature subset selection method leads to an improvement of the AUC value, in comparison to the case when no feature subset selection is used. The result shows that Pearson's correlation feature subset selection improves the classification performance across all datasets except PC1.

Table 6: *AUC of SVM and K-NN as single, bagging, and boosting classifiers on five datasets with Pearsons's Correlation Feature Subset Selection.*

Dataset	Single Classifier		Bagging Classifier		Boosting Classifier	
	SVM	K-NN	SVM	K-NN	SVM	K-NN
CM1	**0.754**	0.637	**0.764**	0.647	0.677	0.639
KC1	0.665	0.657	**0.732**	0.726	**0.733**	0.530
KC3	0.566	**0.693**	0.578	0.655	0.593	0.648
MC2	0.642	**0.785**	0.673	**0.775**	0.673	0.563
PC1	0.729	0.721	0.772	0.750	0.719	0.647

Compared with no feature selection (Table 3), SVM bagging based on PC feature subset selection improved the AUC over two datasets (CM1 and KC1). However, K-NN boosting was unable to improve the AUC for any of the datasets. The reason is that K-NN is very sensitive to the input selection as described in [59]. Moreover, again we observed that the simple classifiers and ensemble methods, after applying PC feature subset selection, do not bring about any improvement in accuracy for the dataset PC1. So, it is clear that the performance of PC feature subset selection depends largely on the dataset. The best subset of features varies from one dataset to another (see Table 4).

In the Greedy Stepwise experiment, Greedy stepwise feature subset selection was employed to identify the relevant set of features. This resulted in a reduced set of features (i.e. the relevant features).

- Single Classifier: For the five datasets, two single classifiers were constructed based on SVM and K-NN from the selected features by GS datasets. The AUC performance measurements were computed and are exhibited in Table 7.
- Bagging: Two bagging ensemble classifiers were constructed based on the bagging algorithm using SVM and K-NN base classifiers based on the reduced datasets by GS. The AUC performance measurements were computed and are exhibited in Table 7.

- AdaBoost Classifier: Two boosting ensemble classifiers were constructed based on the AdaBoostM1 algorithm and using SVM and K-NN base classifiers based on the reduced datasets by GS. The AUC performance measurements were computed and are exhibited in Table 7.

Table 7 shows the AUC for Greedy Stepwise (GS) feature subset selection for the selected classifiers. Again, the values in bold indicate that the use of the feature subset selection method leads to an improvement of the AUC value, compared to the case when no feature subset selection is used. GS was able to improve the AUC using different classifiers for the MC2 dataset only. Moreover, again we observed that the simple classifiers and ensemble methods, after applying GS feature subset selection, do not bring about any improvement in accuracy for the dataset PC1. So it is clear that the performance of GS feature subset selection depends largely on the dataset. The best subset of features varies from one dataset to another (see Table 4).

Table 7: *AUC of SVM and K-NN as single, bagging, and boosting classifiers on five datasets with Greedy Stepwise Feature Subset Selection.*

Dataset	Single Classifier		Bagging Classifier		Boosting Classifier	
	SVM	K-NN	SVM	K-NN	SVM	K-NN
CM1	0.688	0.646	0.679	0.647	0.652	0.646
KC1	0.674	0.637	0.684	0.726	0.720	0.637
KC3	0.593	0.657	0.602	0.655	0.600	0.639
MC2	**0.661**	0.693	**0.686**	**0.775**	**0.683**	0.530
PC1	0.795	0.785	0.818	0.750	0.763	0.648

In the Information Gain experiment, Information Gain was employed to determine the set of relevant features. The results were a set of ranked features. We used $\log_2$ n as a way to select the top ranked features (i.e. the relevant features) over the five datasets [58].

- Single Classifier: With the five datasets, two single classifiers were constructed based on SVM and K-NN from the selected features by IG datasets. The AUC performance measurements were computed and are exhibited in Table 8.
- Bagging: Two bagging ensemble classifiers were constructed based on the bagging algorithm using SVM and K-NN base classifiers and the reduced datasets by IG. The AUC performance measurements were computed and are exhibited in Table 8.
- AdaBoost Classifier: Two boosting ensemble classifiers were constructed based on AdaBoostM1 algorithm and using SVM and K-NN base classifiers using the reduced datasets by IG. The AUC performance measurements were computed and are exhibited in Table 8.

Table 8: *AUC of SVM and K-NN as single, bagging, and boosting classifiers on five datasets with Information Gain Feature Subset Selection.*

Dataset	Single Classifier		Bagging Classifier		Boosting Classifier	
	SVM	K-NN	SVM	K-NN	SVM	K-NN
CM1	**0.721**	0.629	**0.733**	0.647	**0.721**	0.629
KC1	0.647	0.646	**0.739**	0.664	**0.749**	**0.646**
KC3	0.622	0.637	0.597	0.647	0.580	0.637
MC2	0.666	0.657	0.671	**0.726**	0.630	0.639
PC1	**0.742**	0.693	0.773	0.655	0.770	0.530

Table 8 shows the AUC for information gain (IG) feature subset selection for the selected classifiers. Again, the values in bold indicate that the use of the feature subset selection method leads to an improvement of the AUC value, compared to the case when no feature subset selection is used. The result shows that all of the classifiers except the simple K-NN classifier were able to improve the AUC for some of the datasets. For example, compared to the case without feature selection (Table 3), SVM bagging and SVM boosting based on IG feature subset selection improved the AUC over two datasets (CM1 and KC1). Moreover, again we observed that the simple classifiers and ensemble methods, after applying IG feature subset selection, do not bring any improvement in accuracy for the dataset PC1. So it is clear that the performance of IG feature subset selection depends largely on the dataset. The best subset of features varies from one dataset to another (see Table 4).

In order to highlight all of the information of the experimental results, we used boxplots to visually show the classification AUC values and outliers of the various algorithms on the various datasets. Figures 4 and 5 show AUC boxplots for all datasets using SVM and K-NN algorithms, with the boxes representing the three different modeling types: single, bagging, and boosting classifiers on the four feature selection techniques and without feature subset selection. The middle line inside each box is the median. The ends of the boxes are the quartiles. The whiskers outside the boxes extend to the smallest and largest AUC values. The outlier AUC values are plotted individually. Clearly, there was no significant difference in the performance of the feature subset methods, as their respective performance and effect varies from dataset to dataset and the choice of classification algorithm. This research outcome is similar to the findings from [60, 61, 62].

4.3 One-Way ANOVA Test

The One-Way ANOVA-test (analysis of variance) compares the means of two or more independent sets in order to judge whether there is statistical evidence that the associated instances' means are significantly different [63].

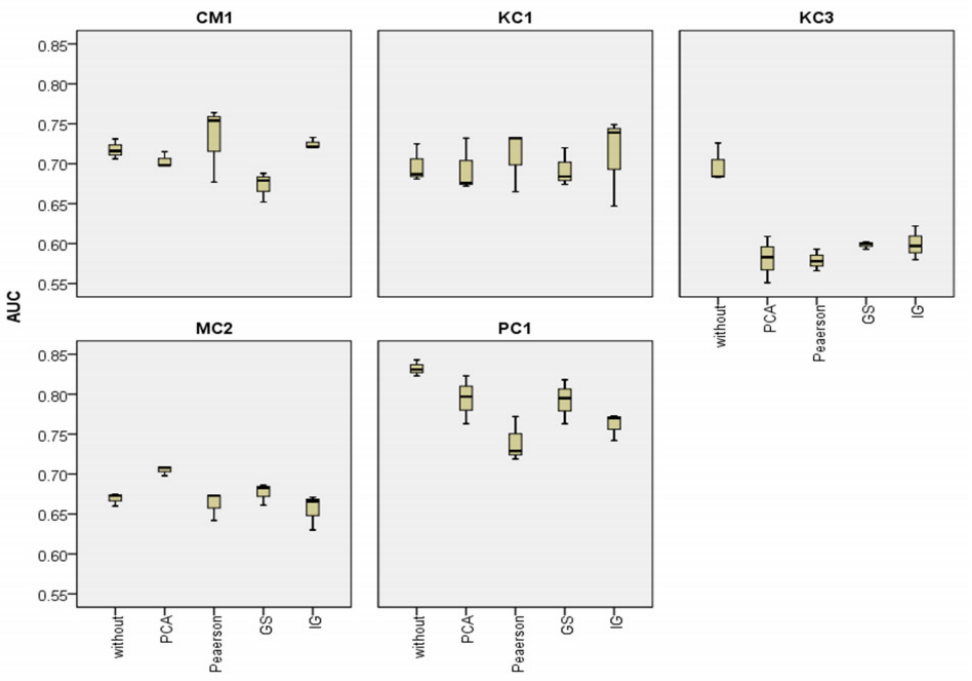

Fig.4: *AUC Boxplots of various SVM classifiers on all datasets for the different feature selection techniques.*

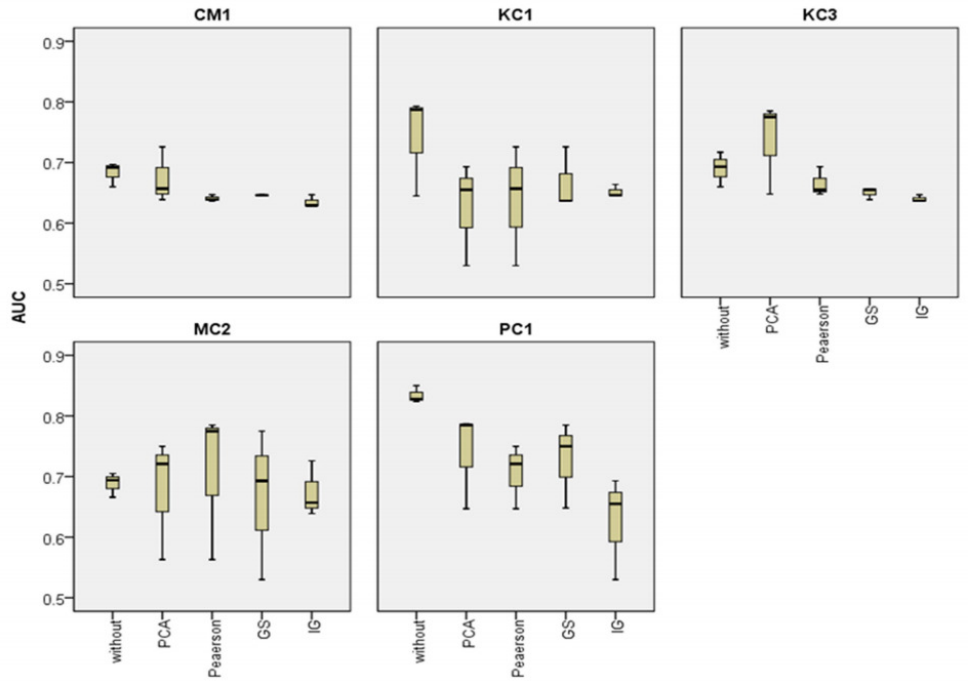

Fig.5: *AUC Boxplots of various K-NN classifiers on all datasets for the different feature selection techniques.*

We applied the One-Way ANOVA to check if there were statistical differences between the means of the two scenarios (without and with feature subset selection). The predetermined α= 0.05 is the significance level for the ANOVA experiment.

Hypotheses: The null and alternative hypotheses of one-way ANOVA can be declared as follows:
$H_0 : \mu_1 = \mu_2$ (the two scenarios' means are equal).
$H_1 : \mu_1 \neq \mu_2$ (the two scenarios' means are different).
Where, μ_i is the instances mean of the i$^{\text{th}}$ scenarios.

5. RESULTS ANALYSIS AND DISCUSSION

The accurate selection of software metrics is an important factor that has an effect on the performance of single and ensemble classifiers. The main questions of this research are: 1) Do feature subset selection methods have an impact on the performance of ensemble techniques? and 2) If so, is this impact positive (improves the model's performance) or negative (decreases the model's performance)? The answers to these questions are presented using the One-Way ANOVA, the results of which are shown in Tables 9 and 10.

Table 9: *One-way ANOVA Results for SVM.*

Feature Selection technique		Sum of Squares	df	Mean Square	F	Sig.
without	Between Groups	.001	2	.000	.117	.890
	Within Groups	.050	12	.004		
	Total	.051	14			
PCA	Between Groups	.001	2	.000	.073	.930
	Within Groups	.074	12	.006		
	Total	.075	14			
Pearson	Between Groups	.003	2	.001	.029	.753
	Within Groups	.060	12	.005		
	Total	.063	14			
GS	Between Groups	.000	2	.000	.039	.962
	Within Groups	.061	12	.005		
	Total	.061	14			
IG	Between Groups	.001	2	.001	.142	.869
	Within Groups	.056	12	.005		
	Total	.057	14			

The results concluded by the One-Way ANOVA can be presented as follows:

1. For SVM, the two scenarios are not statistically significantly different, as presented in Table 9, since the p-values for all the feature selection techniques are greater than the predetermined $\alpha = 0.05$. Therefore, we accept the null hypothesis which clarifies that all scenarios (without, PCA, Pearson, GS, and IG) have means which are equal, and that choosing the SVM as single, bagging, or boosting classifiers has no impact.
2. For K-NN, we found two feature selection techniques are statistically significantly different since the p-value is less than the predetermined $\alpha = 0.05$. Therefore, we reject the null hypothesis, and accept the alternate hypothesis which states that the two scenarios are statistically significantly different.

Table 10: *One-way ANOVA Results for K-NN.*

Feature Selection technique		Sum of Squares	df	Mean Square	F	Sig.
without	Between Groups	.006	2	.003	.544	.594
	Within Groups	.061	12	.005		
	Total	.067	14			
PCA	Between Groups	.055	2	.027	9.152	.004
	Within Groups	.036	12	.003		
	Total	.091	14			
Pearson	Between Groups	.033	2	.017	5.134	.024
	Within Groups	.039	12	.003		
	Total	.072	14			
GS	Between Groups	.022	2	.011	3.421	.067
	Within Groups	.038	12	.003		
	Total	.060	14			
IG	Between Groups	.007	2	.004	2.573	.117
	Within Groups	.016	12	.001		
	Total	.023	14			

In order to determine which pairs of means scenarios are statistically significantly different, and which are not, we performed a multiple pairwise comparison using Tukey's Honestly Significant Difference (HSD) criterion (Berenson et al., 1983). The significance level for Tukey's HSD test is α =0.05. Table 11 presents Tukey's HSD on the K-NN classifier. The bold values indicate the significant difference between pairs in each group. Table 11 shows the significant differences. These differences are found in the PCA group between single and boosting scenarios where (p =.01) $<$ ($\alpha = 0.05$). Also, in the same group (PCA), there is a difference between the bagging and boosting scenarios where (p =.006) $<$ (α = 0.05). There is also a difference in the Pearson group between the bagging and boosting scenarios where (p =.032) $<$ (α = 0.05).

We can conclude that:

1. PCA feature subset selection improves the K-NN performance when it is used as a boosting classifier over a single classifier.
2. PCA feature subset selection improves the K-NN performance when it is used as a bagging

Table 11: *Tukey's HSD Multiple Comparisons for K-NN Classifier.*

Dependent variable	(I) Scenario	(J) Scenario	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Without	single	bagging	-.0242000	.0452765	.856	-.144991	.096591
		boosting	.0230000	.0452765	.869	-.097791	.143791
	bagging	single	.0242000	.0452765	.856	-.096591	.144991
		boosting	.0472000	.0452765	.566	-.073591	.167991
	boosting	single	-.0230000	.0452765	.869	-.143791	.097791
		bagging	-.0472000	.0452765	.566	-.167991	.073591
PCA	**single**	bagging	-.0096000	.0346195	.959	-.101960	.082760
		boosting	.1232000	.0346195	**.010**	.030840	.215560
	bagging	single	.0096000	.0346195	.959	-.082760	.101960
		boosting	.1328000	.0346195	**.006**	.040440	.225160
	boosting	**single**	-.1232000	.0346195	**.010**	-.215560	-.030840
		bagging	-.1328000	.0346195	**.006**	-.225160	-.040440
Peaerson	single	bagging	-.0120000	.0359414	.941	-.107887	.083887
		boosting	.0932000	.0359414	.057	-.002687	.189087
	bagging	single	.0120000	.0359414	.941	-.083887	.107887
		boosting	.1052000	.0359414	**.032**	.009313	.201087
	boosting	single	-.0932000	.0359414	.057	-.189087	.002687
		bagging	-.1052000	.0359414	**.032**	-.201087	.009313
GS	single	bagging	-.0270000	.0355652	.734	-.121883	.067883
		boosting	.0636000	.0355652	.215	-.031283	.158483
	bagging	single	.0270000	.0355652	.734	-.067883	.121883
		boosting	.0906000	.0355652	.062	-.004283	.185483
	boosting	single	-.0636000	.0355652	.215	-.158483	.031283
		bagging	-.0906000	.0355652	.062	-.185483	.004283
IG	single	bagging	-.0154000	.0233529	.791	-.077702	.046902
		boosting	.0362000	.0233529	.304	-.026102	.098502
	bagging	single	.0154000	.0233529	.791	-.046902	.077702
		boosting	.0516000	.0233529	.110	-.010702	.113902
	boosting	single	-.0362000	.0233529	.304	-.098502	.026102
		bagging	-.0516000	.0233529	.110	-.113902	.010702

classifier over a boosting classifier.

3. Pearson's correlation feature subset selection improves the K-NN performance when it is used as a bagging classifier over a boosting classifier, but does not improve it in comparison with bagging K-NN and without feature selection.

6. THREAT TO VALIDITY

In this empirical study we employed four different feature selection techniques to show their effectiveness on three different ensemble classification methods. Five datasets obtained from the PROMIS repository were analyzed. The factors affecting the external validity were the quality of the datasets and generalization. The characteristics of the datasets were different as described in Table 1. For example, the datasets were written in different languages, had a different numbers of instances, different numbers of attributes, etc. The datasets were created by NASA, are publicly available, and are often used by researchers as described in the literature review section. However, we cannot generalize the results for this experiment to other software defect datasets. The other validity issue is the choice of classification algorithms and the feature selection techniques used in the experiment. We selected three classification algorithms and three feature selection techniques which have been used successfully in software defect prediction. However, new research can be conducted using other algorithms and techniques in this area. The third validity concern is the performance measure used to evaluate the performance of prediction models. In the experiment, the area under curve (AUC) was used to measure the percentage of the correctly classified instances. However, other performance measurements could be used, such as accuracy and F-measure.

7. CONCLUSIONS

Software defect prediction is the process of categorizing software modules into defective or non-defective entities. Building an efficient automated software defect predictor that can detect defects early is an important issue in software defect prediction, because the existence of a defect within a software system results in a lower quality software system. That in turn, may have a critical effect, especially with systems that have an impact on our lives.

The primary aim of this research was to assess the effectiveness of feature subset selection techniques coupled with ensemble methods on software defect prediction models and to construct an efficient automated software defect model that has the ability to improve the defect prediction process. Therefore, in this research we investigated a new technique for constructing an automated software defect model based

on a hybrid machine learning technique and a good feature selection method. The hybrid machine learning based on ensemble methods is bagging, thus we selected this method as it has the ability to improve prediction. It is efficient and has improved classification results in a wide variety of other research fields. The principal component analysis feature selection technique is employed in the software defect model since it has been shown to be a good feature selection technique in research in other fields. We applied the model to five datasets, consisting of different software metrics, which were obtained from NASA projects in the PROMISE depository. Then we applied principal component analysis and bagging ensemble methods in addition to applying single classifiers to confirm the effectiveness of that method. The results demonstrate that K-NN bagging based on PCA is better than the K-NN single classifier, and that it improves classification results. Ensemble methods can improve a model's performance without any feature selection techniques. The three feature selection techniques employed in this research (Pearson's correlation, Greedy stepwise, and IG) actually decrease the ensemble model's performance.

As a future work, we plan to assess the effectiveness of more feature subset selection mechanisms. Also, different ensemble methods will be considered such as stacking and voting. In addition, experiments using datasets from different software projects will also be included to try and discover whether the programming languages used have an impact on defect prediction or not. Furthermore, we plan to use more base classifiers with ensemble methods to identify the effect of base classifiers on ensemble model performance.

ACKNOWLEDGEMENTS

This paper is part of the Master's graduation project submitted by Lina Abuwardih to IS department, Faculty of IT, Yarmouk University.

References

[1] R. Chang, X. Mu., and L. Zhang, "Software defect prediction using non-negative matrix factorization," *Journal of Software*, vol.6, no.11, pp.2114–2120, 2011.

[2] N. Fenton, and J. Bieman, *Software metrics: a rigorous and practical approach*, Third Edition (3rd. ed.). CRC Press, Inc., USA., 2014

[3] S. Agarwal, and D. Tomar, "A feature selection based model for software defect prediction," *International Journal of Advanced Science and Technology*, vol.65 , pp.39-58, 2014.

[4] S. Liu, X. Chen, W. Liu, J. Chen, Q. Gu, and D. Chen, "Fecar: A feature selection framework for software defect prediction," *2014 IEEE 38th Annual Computer Software and Applications Conference*, Vasteras, 2014, pp. 426–435.

[5] 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*, vol.41, issue 5, pp.579–606, 2011.

[6] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, S, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," in *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485-496, July-Aug. 2008.

[7] M. Jureczko, and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, page 9. ACM, 2010

[8] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Information Sciences*, vol.264, pp.260–278, 2014.

[9] K. O. Elish, and M. O. Elish, "Predicting defect-prone software modules using support vector machines," *Journal of Systems and Software*, vol.81, issue 5, pp.649–660, 2008.

[10] A. Okutan, A. and O. T. Yıldız, "Software defect prediction using bayesian networks," *Empirical Software Engineering*, vol.19, issue 1, pp.154–181, 2014.

[11] P. Knab, M. Pinzger, and A. Bernstein, "Predicting defect densities in source code files with decision tree learners," in *Proceedings of the 2006 international workshop on Mining software repositories*, pp. 119–125. ACM, 2006.

[12] T. Menzies, J. Greenwald and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," in *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2-13, Jan. 2007

[13] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *IEEE Transactions on Software Engineering*, vol.37, no.3, pp.356–370, 2011.

[14] B. Ghotra, S. McIntosh and A. E. Hassan, "Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models," *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Florence, 2015, pp. 789-800.

[15] K. Punitha, and B. Latha, "Sampling imbalance dataset for software defect prediction using hybrid neuro-fuzzy systems with naive bayes classifier," *Tehnicki vjesnik/Technical Gazette*, vol.23, no.6, pp.1795-1804, 2016.

[16] M. C. Prasad, L. Florence, and A. Arya, "A study on software metrics based software defect prediction using data mining and machine learning techniques," *International Journal of Database Theory and Application*, vol.8, no.3, pp.179–190, 2015.

[17] J. Petrić, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "Building an ensemble for software defect prediction based on diversity selection," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 46. ACM, 2016.

[18] S. Wang, and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, vol.62, no.2, pp.434–443, 2013.

[19] A. A. Saifan, and N. Alsmadi, "Source Code-Based Defect Prediction Using Deep Learning and Transfer Learning," *Journal of Intelligent Data Analysis*, vol.23, no.6, pp.1243-1269, 2019.

[20] R. Ozakinci, and A. Tarhan, "The role of process in early software defect prediction: Methods, attributes and metrics," in *International Conference on Software Process Improvement and Capability Determination*, pp. 287–300. Springer, 2016.

[21] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "Software measurement data reduction using ensemble techniques," *Neurocomputing*, vol.92, pp.124–132, 2012.

[22] A. A. Saifan, H. Alsghaier, K. Alkhateeb, "Evaluating the Understandability of Android Applications," *International Journal of Software Innovation (IJSI)*, vol.6, no.1, pp.44-57, 2017.

[23] A. A. Saifan, A. Alrabadi, "Evaluating Maintainability of Android Applications," *The 8th International Conference on Information Technology ICIT 2017*, Amman, Jordan, 2017.

[24] F. Hanandeh, A. A. Saifan, M. Akour, N. Al-Hussein, K. Shatnawi, "Evaluating Maintainability of Open Source Software: A Case Study," *International journal of open source software and processes*, vol.8, issue 1, 2017.

[25] J. Nam, "Survey on software defect prediction," *Department of Computer Science and Engineering, The Hong Kong University of Science and Technology*, Tech. Rep, 2014.

[26] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of machine learning research*, vol.3, pp.1289–1305, 2003.

[27] P. Gupta, and T. Dallas, "Feature selection and activity recognition system using a single triaxial accelerometer," *IEEE Transactions on Biomedical Engineering*, vol.61, no.6, pp.1780–1786, 2014.

[28] Q. A. Al-Radaideh, E. M. Al-Shawakfa, and M. I. Al-Najjar, "Mining student data using decision trees," in *International Arab Conference on Information Technology (ACIT'2006)*, Yarmouk University,2006.

[29] G. Chandrashekar, and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol.40, no.1, pp.16–28, 2014.

[30] L. C. Molina, L. Belanche, and A. Nebot, "Feature selection algorithms: A survey and experimental evaluation," *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, Maebashi City, Japan, 2002, pp.306-313.

[31] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol.42, no.4, pp.463–484, 2012.

[32] H. I. Elshazly, A. M. Elkorany, A. E. Hassanien, and A. T. Azar, "Ensemble classifiers for biomedical data: performance evaluation," in *Computer Engineering & Systems (ICCES), 2013 8th International Conference on*, pp.184–189, 2013.

[33] S. Kotsiantis, K. Patriarcheas, and M. Xenos, "A combinational incremental ensemble of classifiers as a technique for predicting students performance in distance education," *Knowledge- Based Systems*, vol.23, no.6, pp.529–535, 2010

[34] M. Scholz, and R. Klinkenberg, "An ensemble classifier for drifting concepts," in *Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams*, Porto, Portugal, pp.53–64, 2005.

[35] M. Woźniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol.16, pp.3–17, 2014.

[36] T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction," in *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, vol. 1, pp. 137–144, 2010.

[37] T. M. Khoshgoftaar, K. Gao, and A. Napolitano, "An empirical study of feature ranking techniques for software quality prediction," *International Journal of Software Engineering and Knowledge Engineering*, vol.22, no.2, pp.161–183, 2012.

[38] J. Murillo-Morera, C. Castro-Herrera, J. Arroyo, and R. Fuentes-Fernández, "An automated defect prediction framework using genetic algorithms: A validation of empirical studies," *Inteligencia Artificial*, vol.19, no.57, pp.114–137, 2016.

[39] R. Vivanco, Y. Kamei, A. Monden, K. Matsumoto, and D. Jin, "Using search-based metric selection and oversampling to predict fault prone modules," in *Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on*, pp.1–6, 2010.

[40] L. Jia, "A hybrid feature selection method for software defect prediction," *IOP Conf. Ser. Mater. Sci. Eng.*, vol.394, no.3, pp.032035, 2018.

[41] I. Arora, and A. Saha, "Software Defect Prediction Using ELM and KELM Based Feature Selection Models," *Proceedings of International*

Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur - India, February, pp.26-28, 2019.

[42] W. Tao, L. Weihua, S. Haobin, and L. Zun, "Software defect prediction based on classifiers ensemble," *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, vol.8, no.16, pp.4241–4254, 2011.

[43] V. Jayaraj, and N.S. Raman, "Software defect prediction using boosting techniques," *International Journal of Computer Applications*, vol.65, no.13, pp.1-4, 2013.

[44] R. S. Wahono, and N.S. Herman, "Genetic feature selection for software defect prediction," *Advanced Science Letters*, vol.20, pp.239–244, 2014.

[45] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, vol.58, pp.388–402, 2015.

[46] A. Abdou and N. Darwish, "Early Prediction of Software Defect using Ensemble Learning: A Comparative Study," *International Journal of Computer Applications*, vol.179, no.46, pp.29-40, June 2018.

[47] A. Alsaeedi, and M. K Khan, "Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study," *Journal of Software Engineering and Applications*, vol.12, no.5, pp.85-100, `https://doi.org/10.4236/jsea`, 2019.

[48] J. S. Shirabad, and T.J. Menzies, "The promise repository of software engineering databases," *School of Information Technology and Engineering*, University of Ottawa, Canada, 24, 2005.

[49] I. H. Witten, E. Frank, M .A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*, Fourth Edition, Morgan Kaufmann, (Fourth Edition), 2017.

[50] A. W. Moore, "Support vector machines," Tutorial. School of Computer Science of the Carnegie Mellon University, 2001, Available at `http://www.cs.cmu.edu/œ~awm/tutorials.`[Accessed August 16, 2009].

[51] J. A. Suykens, and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol.9. issue 3, pp.293–300, 1999.

[52] G. H. John, and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp.338–345. Morgan Kaufmann Publishers Inc, 1995.

[53] L. Breiman, "Bagging predictors," *Machine learning*, vol.24, issue 2, pp.123–140, 1996.

[54] Y. Freund, R. E. Schapire, et al., "Experiments with a new boosting algorithm," in *icml*, vol. 96, pp. 148–156, 1996.

[55] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*, Third Edition, Elsevier, 2012

[56] U. Waikato, "Weka tool," `http://www.cs.waikato.ac.nz/ml/weka/index.html`, accessed June 22, 2020

[57] M. J. Norušis, *IBM SPSS statistics 19 statistical procedures companion*, Prentice Hall, 2012.

[58] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," in *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, vol. 2, pp. 310–317, 2007.

[59] N. García-Pedrajas, and D. Ortiz-Boyer,"Boosting k-nearest neighbor classifier by means of input space projection," *Expert Systems with Applications*, vol.36, issue 7, 10570–10582, 2009.

[60] Z. Xu, J. Liu, Z. Yang, G. An and X. Jia, "The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison," 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, ON, 2016, pp. 309-320.

[61] M. Kondo, C.-P. Bezemer, Y. Kamei, A. E. Hassan, O. Mizuno,"The impact of feature reduction techniques on defect prediction models," *Empir. Softw. Eng.*, pp.1–39, 2019.

[62] K. Muthukumaran, A. Rallapalli, N. Murthy, "Impact of feature selection techniques on bug prediction models," in *Proceedings of the 8th India Software Engineering Conference*, Bangalore, India, 18–20 February 2015; ACM: New York, NY, USA, 2015; pp.120–129, 2015.

[63] A. Field, *Discovering statistics using IBM SPSS statistics*, Fourth Edition, Sage, 2013.

Ahmad Saifan is an associate professor in the department of information systems at Yarmouk University (YU) in Jordan. He obtained his Ph.D degree in software engineering from Queen's University (Canada). His master degree in computer science from YU. He had B.Sc degree in computer science from YU. His research interest include: datamining for software engineering, model-based testing, regression tests, software testing.

Lina A. Abuwardih obtained her Master degree in Computer Information Systems (CIS) from Yarmouk University, Jordan, 2017. She is working as a teacher assistant in Jordan University of Science and Technology (JUST). Her research interests include: Information Retrieval, Data Mining, software engineering and Information Security.