SOFTWARE DEFECT PREDICTION: EFFECT OF FEATURE SELECTION AND ENSEMBLE METHODS

Supported by tetfund TERTIARY EDUCATION TRUST FUND

M. A. Mabayoje, A. O. Balogun*, A. O. Bajeh and B. A. Musa
Department of Computer Science, University of Ilorin, Kwara State, Nigeria
*Corresponding author: Balogun.ao1@unilorin.edu.ng

Received: October 18, 2017 **Accepted:** July 19, 2018

Abstract: Software defect prediction is the process of locating defective modules in software. It facilitates testing efficiency and consequently software quality. It enables a timely identification of fault-prone modules. The use of single classifiers and ensembles for predicting defects in software has been met with inconsistent results. Previous analysis say ensemble are often more accurate and are less affected by noise in datasets, also achieving lower average error rates than any of the constituent classifiers. However, inconsistencies exist in these various experiments and the performance of learning algorithms may vary using different performance measures and under different circumstances. Therefore, more research is needed to evaluate the performance of ensemble algorithms in software defect prediction. Adding feature selection reduces data sets with fewer features and improves the classifiers and ensemble performance over the datasets. The goal of this paper is to assess the efficiency of ensemble methods in software defect prediction using feature selection. This study compares the performance of four ensemble algorithms using 11 different performance metrics over 11 software defect datasets from the NASA MDP repository. The results indicate that feature selection and use of ensemble methods can improve the classification results of software defect prediction. Bagged ensemble models have the best results. In addition, Voting and Stacking also performed better than individual base classifiers. In terms of single classifier, SMO performs best as it outperformed Decision Tree (J48), MLP, and KNN with and without feature selection. Thus, it can be derived that feature selection can help improve the accuracy of both individual classifiers and ensemble methods by removing noisy and inconsistent features in the datasets.

Keywords: Classification, dataset, ensemble, feature selection, software defect prediction

Introduction

Software engineering is an engineering discipline that is concerned with all aspects of producing software from the early stages of software specification through to maintaining the system after it has gone into use (Sommerville, 2013). It can also be defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (Fenton and Bieman, 2014). In any area of software engineering, errors are sometimes inescapable and these errors mostly result into defects and failures in the software. Usually during the development process, software defects are discovered during software testing (Wohlin *et al*., 2012).

A software defect is an error or flaw in a software program or system that causes the production of unexpected result (Malhotra, 2015). A software defect can also be the case when the final software product does not meet the customer requirement or user expectation (Wahono, 2015). Defects increase the cost of software development and decrease the overall quality of the software product.

Previous studies illustrate that ensemble methods, a combination of classifiers using some mechanisms, are superior to using single classifiers in software defect prediction (Laradji *et al*., 2015, Akintola *et al*., 2018). However, other works indicated that classifiers' performances may vary in terms of different performance measures and under different circumstances (Balogun *et al*., 2015, Ameen *et al*., 2016). Furthermore, there are many ways to construct ensembles of classifiers for classification processes but caution must be exercised in terms of the overhead cost usually incurred when considering ensemble methods. There are many other ways of developing ensembles but how to pick the best ensemble methods for software defect prediction has not been fully ascertained (Peng *et al*., 2011). The aim of this study is targeted at evaluating the performance of ensemble methods and classification models with and without feature selection.

Materials and Methods

Classification models

SMO is a simple and proficient algorithm for solving the quadratic programming (QP) problem arising in support vector machines (SVM) (Keerthi and Gilbert, 2002). SMO solves the SVM QP problem by segmenting it into QP sub-problems and solving the smallest possible optimization problem, involving two Lagrange multipliers at each step (Platt, 1998). Dissimilar to the past strategies, SMO chooses to solve the smallest possible optimization problem at every step and it chooses two Lagrange multipliers to jointly optimize and finds the optimal values for these multipliers (Platt, 1998).

Multi-layer Perceptron Networks (MLP) are feed-forward artificial neural networks which is a famous model for machine learning (Kawam and Mansour, 2012). MLP was developed to replicate learning and generalization abilities of humans with an attempt to model the functions of biological neural networks and they have many potential applications in the areas of Artificial Intelligence (AI) and Pattern Recognition (PR) (Roy *et al*., 2005).

Instance Based Knowledge (IBK) or K-Nearest Neighbor classification classifies instances based on their similarities (Adeniyi *et al*., 2016). It is a type of Lazy learning where the function is only approximated locally and all computation is deferred until classification (Patra and Prasad, 2013). An object is classified by a majority of its neighbors. *K* is always a positive integer and the neighbors are selected from a set of objects for which the correct classification is known (Patra and Prasad, 2013).

J48 is a classification algorithm that belongs to the category of decision trees. Decision trees discover the way the attribute vectors behave for various instances (Kaur and Chhabra, 2014). It is a tree in which each internal node corresponds to a decision, with a sub tree at these nodes for each possible outcome of the decision and the possible solutions of the problem correspond to the paths from the root to the leaves of the decision tree (Balogun *et al*., 2015).

Ensemble models

In recent times, the use of ensembles has become prominent and it has been applied to various fields including machine learning, pattern recognition and data mining (Akintola *et al*., 2018, Ameen *et al*., 2016, Peng *et al*., 2011). In contrast to learning approaches that construct one learner from a training data, ensemble methods combines a set of learners for data analysis. Ensemble learning is also called committee based learning or learning multiple classifier system (Laradji *et al*., 2015). Several machine learning algorithms generate a single model (e.g. a decision tree or neural network), ensemble methods combine multiple models. Actually, ensemble methods are appealing mainly because they are able to boost weak learners to make accurate prediction (Zhou, 2012). A weak learner can also be called a single or base learner. Ensemble learners are machine learning methods that leverage the efficiency of multiple models to achieve better accuracy than any of the individual models could on their own (Zhou, 2012).

Boosting

Boosting is an ensemble learning technique. The term boosting refers to a family of algorithms that are able to convert weak learners to strong learners. Instinctively, a frail learner is quite recently marginally superior to anything irregular figure, while a solid learner is near flawless execution (Zhou, 2012, Rokach, 2010). In boosting, however, weights of training instances change in each iteration to force learning algorithms to emphasize on instances that were predicted incorrectly (Dietterich, 2000). Variants of boosting technique includes Adaboost and LogitBoost (Zhou, 2012)

Bagging

The name Bagging came from the abbreviation of Bootstrap Aggregating. As the name implies, the two ingredients of boosting are bootstrap and aggregation. Bagging adopts the bootstrap distribution for generating different base learners (that is, it applies bootstrap sampling to obtain the data subsets for training the base classifiers) (Dietterich, 2000). Bagging combines multiple outputs of a learning algorithm by taking a plurality vote to get an aggregated single prediction (Dietterich, 2000). The multiple outputs of a learning algorithm are generated by randomly sampling with replacement of the original training dataset and applying the predictor to the sample (Peng *et al*., 2011).

Stacking

Stacking is a meta-learning technique. Meta-learning means learning from the classifiers produced by the creators and from the classifications of these classifiers on training data (Rokach, 2010). Stacking is a general procedure where a learner is trained to combine the individual learners. Here, the individual learners are called the first-level learners, while the combiner is called the *second-level learner*, or meta-learner (Zhou, 2012)**.** Stacking is a technique for achieving the highest generalization accuracy (Rokach, 2010). Unlike bagging and boosting, stacking can be applied to combine different types of learning algorithms (Peng *et al*., 2011). In stacking, each base learner, also called “level 0” model, produces a class value for each instance then the predictions of level-0 models are then fed into the next level model which combines them to form a final prediction (Witten *et al*., 2005)

Voting

This is an ensemble algorithm that works on nominal output (Zhou, 2012). Like stacking, voting also combines a series of classifiers together to perform the classification task. In voting, the combined classifiers vote for a class label.

Feature selection technique

Feature selection is a process that selects a subset of original features. It is also known as attribute selection or reduction (Balogun *et al*., 2015). It is one of the most important techniques used in data preprocessing for data mining. Mining on a reduced set of attributes offers benefits as it reduces the number of attributes appearing in the extracted patterns. There are several types of feature selection algorithms among which is CfsSubsetEval. CfsSubsetEval evaluates the worth of a subset of features by considering the individual predictive ability of each feature along with the degree of redundancy between them, subsets of features that are highly correlated with the class while having low inter-correlation are preferred (Witten *et al*., 2005). This feature selection algorithm like other feature selection algorithm works together with a search algorithm that guides the generation of feature subsets. There are various search algorithms that can be used and they include best first search algorithm, Genetic search algorithm and Greedy step wise algorithm, etc.

Several studies have been carried out on software defect prediction; this section presents a review of studies involving defect prediction, feature selection, and ensemble methods.

Numerous predictive tools have been constructed till now to recognize the defects in software modules using machine learning and statistical approaches. Various Data Mining approaches such as Decision Trees (DT), Bayesian Belief Network (BBN), Artificial Neural Network (ANN), SVM and clustering are some techniques which are generally used to predict defects in software. A defect prediction model based on an enhanced Multilayer Perceptron Neural Network technique using data mining is proposed and explored in (Gayathri and Sudha, 2014). In which comparative analysis of modeling of defect proneness predictions using dataset of different metrics from NASA MDP (Metrics Data Program) was performed and their results indicated that Multi-Layer Perceptron improves the efficiency of prediction.

Feature selection has been used by various studies. Rodriguez *et al*. (2007) applied feature selection with three filter models and three wrapper models to five software engineering data sets. They concluded that the reduced data sets maintained the prediction capability with fewer attributes than the original data sets. Ensembles have also been applied to various diverse fields. Viola and Jones (2004) proposed a general object detection framework by combining Adaboost with cascade architecture. Huang *et al*. (2000) designed ensemble architecture for pose-invariant face recognition, particularly for recognizing faces with in-depth rotations. Ameen *et al*. (2016) worked on Heterogeneous Ensemble Methods Based on Filter Feature Selection. They studied the effects of feature selection on ensemble methods and found out that feature selection before classification processes improves classification accuracy. However, these studies did not consider software defect datasets. Laradji *et al.* (2015) proposed ensemble learning on software defect dataset based on feature selection. They indicated that greedy forward selection together with average probability ensemble (APE) performed well in SDP than other methods. However, they only consider Average Probability Ensemble as there are other methods of ensemble arrangement. Also, their study was based on Support Vector Machine (SVM) classifier and Random Forest which are not the only classification algorithms that can be to predict software defects (Rathore and Kumar, 2017).

Based on the aforementioned, this study looks to investigate the performance of ensemble methods (Bagging, Boosting, Stacking and Voting) in software defects prediction based on greedy stepwise search feature selection method. This work is unique as the proposed method is based on four different ensemble methods, four heterogeneous classification algorithms and a multi-variate feature selection based on greedy stepwise search method.

Methods

The experiment is aimed at comparing single classifiers and ensemble methods for software defect prediction with and without feature selection. The following paragraphs define the datasets, discuss the experimental design and present the results.

Data sources

The datasets used in this study are 11 public-domain software defect datasets provided by the National Aeronautics Space Administration (NASA) Facility Metrics Data Program (MDP) repository. The brief descriptions of these MDP datasets are provided below.

I. **AR1:** This dataset is from an embedded software in a white-goods product implemented in C, it consists of 121 instances, 29 static attributes.
II. **AR3:** This dataset contains 30 attributes and 63 instances.
III. **CM1:** This dataset is from a science instrument written in a C code with approximately 20 kilo-source lines of code (KLOC). It contains 498 instances and 22 attributes.
IV. **KC1:** This dataset is a system implementing storage management for receiving and processing ground data written in C++. it contains 2109 instances and 22 attributes.
V. **KC2:** This dataset is a system implementing science data processing written in C++. It contains 22 attributes and 522 instances.
VI. **KC3:** This dataset is about the collection, processing, and delivery of satellite metadata. It is written in Java with 18 KLOC, 40 attributes and has 458 instances.
VII. **MC2:** This dataset contains 40 attributes and 161 instances.
VIII. **MW1:** This dataset is about a zero-gravity experiment related to combustion written in C code containing 8 KLOC with 403 modules.
IX. **PC1:** This dataset is a flight software from an earth orbiting satellite that is no longer operational. It contains 40 KLOC of C code with 1107 modules and 22 attributes.
X. **PC3:** This dataset is a flight software from an earth orbiting satellite that is currently operational. It has 40 KLOC of C code with 1563 instances.
XI. **PC4:** This dataset is a flight software from an earth orbiting satellite that is currently operational. It has 36 KLOC of C code with 1458 modules.

Performance measures

There are various measures for judging the efficiency of a classifier. Commonly used performance measures in software defect classification are accuracy, precision, recall, F-measure, AUC, and Mean Absolute Error (Challagulla *et al.*, 2008). The definition of measures used in this study is described below.

i. Accuracy: Accuracy is the percentage of correctly classified instances (Jaiwei and Kamber, 2006). It is one of the widely used measures.

$$\text{Accuracy} = \frac{TN + TP}{TP + FP + FN + TN}$$

ii. True positive (TP) rate: TP is the number of correctly classified fault-prone modules. TP rate measures how well a classifier can recognize fault-prone modules. It also referred to as sensitivity measure.

$$\text{TP rate} = \frac{TP}{TP + TN}$$

iii. False positive (FP) rate: FP rate measures the percentage of non fault-prone modules that were incorrectly classified.

$$\text{FP rate} = \frac{FP}{FP + TN}$$

iv. Precision: This is the number of classified fault-prone modules that actually are fault-prone modules.

$$\text{Precision} = \frac{TP}{TP + FP}$$

v. Recall: This is the percentage of fault-prone modules that are correctly classified.

$$\text{Recall} = \frac{TP}{TP + FN}$$

vi. F-measure: It is the harmonic mean of precision and recall.

$$\text{F-measure} = \frac{2 * precision * recall}{precision + recall}$$

vii. AUC: also known as Area under the ROC (Receiver Operating Characteristics) curve which shows the tradeoff between TP rate and FP rate (Jaiwei and Kamber, 2006).
viii. PRC: also known as Area under the Precision-Recall Curve (PRC) is a single-value measure that originated from the area of information retrieval (Jaiwei and Kamber, 2006). The area under the PRC ranges from 0 to 1.
ix. Kappa Statistics: This performance measure estimates the relationship between the members of an ensemble in multi-classifiers system (Kuncheva, 2004).
x. Mean Absolute Error: This measures how much the predictions deviate from the true probability.
xi. Relative Mean Squared Error: it stands for root mean square value, it measures how much error there is between two datasets comparing a predicted value and an observed or known value.

Experimental procedure

This study selects 4 classifiers to build ensembles. They represent three categories of classifiers (i.e. functions, rules and trees) and were implemented in eclipse by building a path to Waikato Environment for Knowledge Analysis (WEKA) library.

For functions category, Sequential Minimal Optimization (SMO) and Multilayer Perceptron (MLP) were selected. For Rules category, K-Nearest Neighbor (KNN) was selected. For Trees category, Decision tree (J48) was selected. This study uses four popular ensemble methods (i.e. boosting, bagging, stacking, and voting) and evaluates performance on software defect datasets gotten from National Aeronautics and Space Administration (NASA) Metric Data Program (MDP) repository. For boosting, this project focuses on Adaboost. Adaboost is the abbreviation for adaptive boosting algorithm because it adapts to the errors returned by classifiers from previous iterations (Zhou, 2012).

The feature selection technique used in this research work is CfsSubsetEval. It is a multivariate filter-based feature selection algorithm. This works adopts the greedy stepwise algorithm as the search method to be used with CfsSubsetEval feature selection algorithm. Greedy stepwise algorithm searches greedily through the space of attribute subsets (Witten *et al.*, 2005).

As presented in Fig. 1, the experimental process starts with dividing the software defects datasets in to training and testing dataset as it has been used in other studies (Akintola *et al.*, 2018; Laradji *et al.*, 2015; Yu *et al.*, 2017). The training dataset will be pre-processed using the multivariate feature selection method based on greedy stepwise search method. Labani *et al*. (2018) reported that multivariate filter methods are good choice for fast and reliable feature selection method. The pre-processed dataset will be trained both on ensemble methods and individual classification algorithm based on 10-fold cross validation. Predictive models generated from both the ensemble methods (Bagging, Boosting, Stacking and Voting) and the individual classifiers (SMO, MLP, J48 and KNN) will be tested based on performance metrics in Section 4.2. The parameters of the classifiers are optimized as proposed by Tantithamthavorn *et al*. (2017). The performance of the ensemble methods and the individual classifiers will be analyzed with or without the deployment of feature selection. This process is in line with the objective of this study of investigating the impact of feature selection and ensemble methods in software defect prediction.

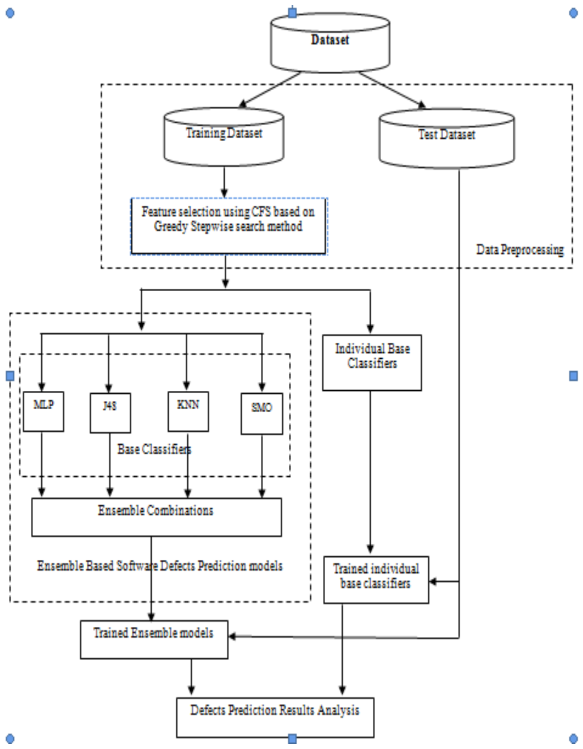
Dataset
Training Dataset
Test Dataset
Feature selection using CFS based on Greedy Stepwise search method
Data Preprocessing
Individual Base Classifiers
MLP
J48
KNN
SMO
Base Classifiers
Ensemble Combinations
Ensemble Based Software Defects Prediction models
Trained individual base classifiers
Trained Ensemble models
Defects Prediction Results Analysis

Fig. 1: Experimental process architecture

Results and Discussion

The Tables present the average of the classification of all learning models used over the 11 datasets divided into training and test datasets. Table 1 presents the results of the learning models without performing attribute reduction while the Table 2 presents the results of the learning models with attribute reduction.

Table 1 presents the averaged results of the algorithms/ensembles without feature selection. The results shows that, in terms of accuracy, Bagged J48 achieves the best results followed by SMO and Boosted J48, in terms of FP rate, SMO achieves the best result followed by Bagged SMO and Boosted SMO. In terms of TP rate, Bagged J48 performs best followed by SMO and Boosted J48, in terms of Precision, boosted J48 performs best followed by Boosted MLP and Bagged J48, in terms of Recall, Bagged J48 achieves the best result followed by SMO and Boosted J48, in terms of F-Measure, Boosted J48 achieves the best results followed by Bagged J48 and Boosted MLP, in terms of ROC, Voting achieves the best results followed by Bagged J48 and Bagged MLP, in terms of PRC, Voting achieves the best results followed by Bagged J48 and Boosted J48. With this observation, it can be proposed that ensemble algorithms perform better than single classifiers.

Table 2 presents the average results of the algorithms/ensembles used with feature selection. The following observations can be made. In terms of accuracy, Bagged J48 achieves the best results followed by Bagged KNN and Bagged MLP which produce the same result. In terms of FP rate, Bagged SMO achieves the best result followed by SMO and Boosted SMO. In terms of TP rate, Stacking performs best followed by Voting and SMO. In terms of Precision, Boosted J48 performs best followed by Stacking and Voting. In terms of Recall, Stacking achieves the best result followed by Voting and SMO. In terms of F-measure, Stacking achieves the best results followed by Bagged MLP and Voting. In terms of ROC, Voting achieves the best results followed by Bagged J48 and Bagged MLP. In terms of PRC, Bagged MLP achieves the best results followed by Bagged J48 and MLP. With this observation, it can also be proposed that ensemble algorithms perform better than single classifiers.

Comparing the respective ensemble models from Table 1 and Table 2, it shows that feature selection enhanced the results of the ensemble models. The percentage of accuracy for the ensemble methods and the individual classifiers with feature selection is better than that with no feature selection. Same is observed with the precision, recall and the FP rate for the ensemble methods with feature selection is lower and better.

Table 1: Average result of classifiers performance (%) with feature Selection

performance measures	SMO	J48	KNN	MLP	Boosted SMO	Boosted J48	Boosted KNN	Boosted MLP	Bagged SMO	Bagged J48	Bagged KNN	Bagged MLP	Stackin g	Voting
Correctly classified	87.78	87.51	87.18	86.89	86.17	85.31	88.07	87.59	88.20	89.27	88.37	88.37	88.24	87.93
Incorrectly classified	12.22	12.57	17.01	13.12	12.84	14.12	16.97	13.57	12.25	13.35	16.62	19.52	11.76	12.07
Kappa statistics	9.99	20.78	22.14	21.68	15.58	25.10	23.10	24.13	8.43	19.35	22.34	22.71	21.97	18.54
Mean absolute error	12.22	17.90	17.56	17.00	18.23	15.27	17.46	18.08	12.53	17.33	17.58	18.02	18.17	16.25
Root mean square error	34.23	32.25	40.25	31.77	31.24	34.01	40.32	33.66	33.75	31.05	34.54	31.20	30.95	30.77
FP rate	80.46	68.81	61.16	68.85	75.08	63.81	60.05	64.84	81.08	69.81	62.05	67.87	70.49	73.75
TP rate	87.78	87.43	82.99	86.88	87.17	85.87	83.03	86.44	87.75	86.65	83.37	86.80	88.24	87.93
Precision	82.29	85.16	82.91	83.91	82.41	85.97	83.08	84.26	81.33	84.34	83.55	84.71	85.54	84.50
Recall	87.78	87.43	82.99	86.88	87.17	85.87	83.03	86.44	87.75	86.65	83.37	86.80	88.24	87.93
F- measure	83.06	84.08	82.75	84.52	83.77	83.81	82.86	84.62	83.02	84.37	82.85	84.67	85.15	84.65
ROC Area	53.65	64.14	66.16	77.12	73.76	74.16	62.14	70.61	57.70	79.25	73.94	78.04	62.68	80.09
PRC Area	79.48	82.54	82.23	87.81	86.14	85.68	80.96	85.55	81.54	88.16	86.33	88.53	83.17	85.10

Table 2: Average result of classifiers performance (%) without feature selection

Performance measures	SMO	J48	KNN	MLP	Boosted SMO	Boosted J48	Boosted KNN	Boosted MLP	Bagged SMO	Bagged J48	Bagged KNN	Bagged MLP	Stacking	Voting
Correctly classified	87.36	86.99	85.23	86.83	86.53	87.24	84.44	86.69	87.06	87.58	85.48	86.12	85.49	87.02
Incorrectly classified	12.64	13.01	14.77	13.17	13.47	12.76	15.56	13.31	12.94	12.42	14.52	13.88	14.51	12.98
Kappa statistics	9.23	24.43	27.38	27.43	15.97	33.38	26.71	29.25	9.79	26.31	27.74	24.18	15.87	17.59
Mean absolute error	13.43	16.13	15.91	16.23	18.17	13.43	15.65	16.91	12.85	16.47	16.52	16.97	19.27	15.28
Root mean square error	34.63	33.55	36.45	32.97	32.34	32.39	36.87	32.98	33.72	29.50	32.78	31.10	32.96	30.09
FP rate	80.28	64.39	58.48	64.59	72.77	54.11	57.15	60.87	79.45	62.98	57.95	64.85	70.83	72.23
TP rate	87.35	86.99	76.37	86.84	86.53	87.24	84.43	86.69	87.05	87.60	85.47	86.13	85.48	87.00
Precision	82.75	84.41	83.80	85.79	83.16	86.28	83.64	85.42	82.56	85.25	84.04	84.01	81.65	82.66
Recall	87.35	86.99	85.23	86.84	86.53	87.24	84.43	86.69	87.05	87.60	85.47	86.13	85.48	87.00
F- measure	82.85	85.18	84.43	85.05	83.39	86.63	83.95	85.31	82.75	85.96	84.66	84.33	82.69	84.12
Roc Area	53.55	68.06	66.81	76.15	71.83	80.55	63.64	72.25	60.09	80.95	75.95	78.29	63.90	81.37
Prc Area	79.35	84.39	82.45	87.85	84.65	89.30	82.50	85.88	82.12	89.58	87.38	88.25	83.11	90.21

Conclusions and Future Work

Though Some studies have reported that the use of ensembles is often more accurate than using single classifiers in the task of defect prediction, variations still exist and the efficiency of learning algorithms may vary using different performance measures and under different circumstances (Peng *et al*., 2011). Therefore, more research is needed to improve our understanding about the performance of ensemble algorithms in software defect prediction. Observing that results using different performance measures over different datasets. The use of feature selection can help improve the accuracy of classifiers by removing noisy and inconsistent features. This work can be extended by introducing Multi-Criteria Decision Methods (MCDM) to validate the best approach considering selected evaluation metrics.

References

Ahmad AK & Nashat M 2012. Metaheuristic optimization algorithms for training artificial neural networks. *Int. J. Comp. & Information Techn.*, 1(2): 1 – 6.

Akintola AG, Balogun AO, Lafenwa-Balogun FB & Mojeed HA 2018. Comparative analysis of selected heterogeneous classifiers for software defects prediction using filter-based feature selection methods. *FUOYE J. Engr. & Techn.*, 3(1): 134 – 137.

Ameen AO, Balogun AO, Usman G & Fashoto SG 2016. Heterogenous ensemble methods based on filter feature selection. *Computing, Information Sys., Devt. Informatics & Allied Res. J.*, 7(4): 63 – 78.

Aruna S, Dilsha D, Radhika R & Swathi JN 2016. Cost sensitive classification and feature selection for software defect prediction. *Int. J. Advanced Res. Comp. Sci. & Software Engr.,* 6(4): 1 – 2.

Asha GK, Jayaram MA & Manjunath AS 2010. Feature subset selection problem using wrapper approach in supervised learning. *Int. J. Comp. Applic.*, 1(7): 1 – 2.

Bauer E & Kohavi R 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning,* 36(1/2): 105–139.

Breiman L 1994. Bagging Predictors, Technical Report, Department of Statistics, University of California, Berkeley, USA.

Dietterich TG 2009. Ensemble Methods in Machine Learning. First International Workshop on Multiple Classifier Systems, pp. 1–15.

Freund Y & Schapire R 1996. Experiments with a new Boosting Algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148-156.

Gaganjot K & Amit C 2014. Improved J48 classification algorithm for the prediction of diabetes. *Int. J. 109csComputer Applic*., 98(22): 1-5.

Gayathri M & Sudha A 2014. Software Defect Prediction System using Multilayer Perceptron Neural Network with Data Mining. *Int. J. Recent Techn. and Engr.*, 3: 54-59.

Huang FJ, Zhou H, Zhang J & Chen T 2000. Pose invariant face recognition. In Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition, pp. 245–250.

Hui 2014. Software Defect Classification Prediction Based on Mining Software Repository, Department of Information Technology, Uppsala University, Sweden.

IEEE 1990. IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.

Keerthi SS & Gilbert EG 2002. Convergence of a Generalized SMO Algorithm for SVM ClassifierDesign.

Labani M, Moradi P, Ahmadizar F & Jalili M 2018. A novel multivariate filter method for feature selection in text classification problems. *Engr. Applic. Artificial Intelligence*, 70: 25-37.

Lan Sommerville 2009. Software Engineering, Boston, United States of America.

Laradji IH, Alshayeb M & Ghouti L 2015. Software defect prediction using ensemble learning on selected features. *Information and Software Technology*, 58: 388-402.

Lessmann S, Baesens B, Mues C & Pietsch S 2008. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4) 485–496.

Lior 2010. Ensemble based classifiers.33, 1-39. doi 10.1007/s10462-009-9124-7.

Mardani A, Jusoh A, Zavadskas EK, Cavallaro F & Khalifah Z 2015. Sustainable and renewable energy: An overview of the application of multiple criteria decision making techniques and approaches. *Sustainability*, 7: 13947–13984.

Naheed A & Shazia U 2011. Analysis of data mining based software defect prediction techniques. *Global J. Comp. Sci. and Techn.*, 11(16): 1-2.

Nikung CO 2005. Ensemble Data Mining Methods. NASA Ames Research Center, USA.

Opitz D & Maclin R 1999. Popular ensemble methods: An empirical study. *J. Artificial Intelligence Res.*, 11: 169–198.

Peng JJ, Wang JQ, Wang J, Yang LJ & Chen XH 2015. An extension of ELECTRE to multi-criteria decision-making problems with multi-hesitant fuzzy sets. *J. Information Sci.*, 307: 113–126.

Peng Y, Kou G, Wang G, Wang H & Ko F 2009. Empirical evaluation of classifiers for software risk management. *Int. J. Infor. Techn. and Decision Making*, 8(4): 749–768.

Platt J 1998. Fast Training of SVMs using Sequential Minimal Optimization.

Rathore SS & Kumar S 2017. A study on software fault prediction techniques. *Artificial Intelligence Review*, 1-73.

Rodriguez D, Ruiz R, Cuadrado-Gallego J & Aguilar-Ruiz J 2007. Detecting fault modules applying feature selection to classifiers. *Proceedings of Eighth IEEE International Conference on Information Reuse and Integration*, Las Vegas, Nevada,pp. 667–672.

Roy K, Chaudhuri C, Kundu M, NASIPURI M & Basu DK 2005. Comparison of the multilayer perceptron and the nearest neighbor classifier for handwritten numeral recognition. *J. Information Sci. and Engr.*, 21: 1-13.