Multimedia Tools and Applications (2022) 81:37033–37055
https://doi.org/10.1007/s11042-021-11488-6

1211: AIOT SUPPORT AND APPLICATIONS WITH MULTIMEDIA

Heterogeneous stacked ensemble classifier for software defect prediction

Somya Goyal[1,2] · **Pradeep Kumar Bhatia**[2]

Received: 23 September 2020 / Revised: 1 June 2021 / Accepted: 19 August 2021 /
Published online: 12 September 2021

Abstract
Software defect prediction (SDP) plays an important role to ensure that software meets quality standards; by highlighting the modules which are prone to errors and hence allows to focus the test efforts on them. Class imbalance nature of the defect dataset hinders the defect predictors to correctly classify the buggy modules. Here, we introduce a novel heterogenous ensemble classifier built with stacking methodology to overcome this problem of imbalanced datasets and hence, significant improvement in the prediction power is being proposed. Stacked ensemble is achieved with the best known classifiers from SDP literature as base classifiers (artificial neural network, nearest neighbor, tree based classifier, Bayesian classifier and support vector machines). For experimental work, five public datasets from NASA corpus are used. A comparative analysis for the proposed heterogenous stacking based ensemble method is made with the base classifiers and with the state-of-the art ensemble based SDP models over the evaluation criteria of ROC, AUC and accuracy. It is found that the proposed heterogenous stacking based ensemble classifier outperforms the base classifiers by 12% in terms of AUC score and by 8% in terms of Accuracy. It improves the performance of state-of-the-art ensemble methods by 4% in terms of AUC score and by 9% in terms of Accuracy. It can be concluded from the comparative analysis that the proposed SDP classifier is best performer among the candidate SDP classifiers statistically.

Keywords Software defect prediction (SDP) · Heterogenous ensemble · Stacking · Artificial neural network · ROC

✉ Somya Goyal
somyagoyal1988@gmail.com

Pradeep Kumar Bhatia
pkbhatia.gju@gmail.com

[1] Manipal University Jaipur, Jaipur, Rajasthan 303007, India

[2] Guru Jambheshwar University of Science & Technology, Hisar, Haryana 125001, India

1 Introduction

Software Defect Prediction (SDP) helps to improve the quality of software product allowing the assessment of the fault-proneness of modules and to forecast which part of software will be requiring more testing and quality assurance (QA) resources [16]. It reduces the testing cost and overall development cost. Machine learning (ML) techniques are finding wide applications in SDP [2, 4, 6, 7, 15, 17]. Solely machine learning methods bring sub-optimal results due to the class imbalance in the defect datasets. Class imbalance refers to the situation when one of the classes in the dataset outnumbers the rest of the classes. The class with higher number of instances is called majority class and the rest id called minority class. This imbalanced nature of defect data negatively impacts the accuracy of ML based SDP classifiers [3, 8, 13]. From the literature survey, it is seen that ensemble learning has better prediction power for software defect prediction using the historic data from the past projects with the condition of "class-imbalance" [24, 25, 28]. Galar et al. (2011) [5] and Rathore et al. (2017) [18] advocated that ensemble based classifiers have built-in capability to handle data-imbalance.

1.1 Motivation

Learning from imbalanced datasets is an open problem. All the proposed ensembles from literature are standard algorithms of random forest, bagging or boosting method. None of the technique is customized as per the application or depending upon the nature of the dataset. This work is contributing to improve the prediction power of classifiers by using a customized heterogenous stacked ensemble classification algorithm.

1.2 Contribution

This work contributes a customized stacked ensemble classifier for the task of SDP provided the data is suffering from class-imbalance. The proposed model is compared with state-of-art techniques to find the best classifier. The statistical evidence is presented to advocate that the proposed model is the best SDP classifier.

1.3 Organization

The paper is organized as follows. Section 2 covers the current state-of-the-art to handle class imbalance using ensembles along the review of the literature. The research methodology is explained in Sect. 3 along with the research questions and the experimental setup. In Sect. 4, the datasets and evaluation metrics used in experimental work are described. In Sect. 5, the experimental results are reported and analysed to answer the research questions. The conclusions are drawn in Sect. 6.

Table 1 State-of-the-art: SDP with ML and SDP with class imbalance

S.No	Year	Study	Technique	Dataset	Attributes/ features	Performance metrics	Observation drawn
1	2013	(Wang and Yao 2013) [25]	ADABOOST Ensembles	PROMISE	McCabe metrics	AUC, ROC, G-Mean	Data space is optimized with sampling, Features are selected and standard ADABOOST technique applied. No innovation is reported
2	2015	(Siers and Islam 2015) [22]	Ensemble + Decision-trees	MC1, MC2, PC1, PC3, PC2, NASA	McCabe metrics	Precision, Recall	Revealed the power of ensembles to deal with class-imbalance in training data
3	2015	(H. Laradji, et al. 2015) [12]	Ensemble	NASA, PC1, PC4, MC1	McCabe metrics	AUC, G-mean	Feature selection techniques are reported with their effect on SVMs and Boost Ensembles. Conclusions are drawn that Ensembles are robust than SVM
4	2015	(Wang et al. 2015) [26]	Ensemble / Multiple Lernel (MKEL)	NASA MDP	All attributes considered	F-Measure	Proposed ensembles reported better results than the literature work
5	2015	(Xia et al. 2015) [27]	Ensemble	Open source Software -Eclipse, Mozilla	All Features Used	F1, Effectiveness Ratio	Proposed ensembles performed better than SMOTE for imbalanced data
6	2018	(Chen et al., 2018) [3]	ADABoost	PROMISE	All Features Used	G-mean, AUC	Results are improved with under sampling and ensembles
7	2017	(Rathore and Kumar 2017) [18]	Ensemble	PROMISE Software Engineering repository data	All Features Used	AE, RE, PRED(l)	Ensemble performed better than single ML technique

Table 1 (continued)

S.No	Year	Study	Technique	Dataset	Attributes/ features	Performance metrics	Observation drawn
8	2017	(Rathore and Kumar 2017) [20]	Linear and non-linear heterogeneous ensemble	PROMISE Software Engineering repository data	All Features Used	AE, RE, PRED(l)	Ensemble performed better than single ML technique
9	2017	(Yang et al. 2017) [28]	Stacked Ensembles	OSS (Bugzilla)	OO metrics	F1-score	Improved results than baseline models
10	2018	(Tong et al. 2018) [24]	Stacked Ensembles	NASA	Static Code metrics and OO metrics	MCC, AUC, F-measure	Proposed model is robust to dataset availability
11	2018	(Huda et al. 2018) [10]	Ensembles	PROMISE	All Features Used	AUC, Accuracy, Recall	Imbalance is dealt effectively with ensemble and oversampling

2 Related works

This section highlights the contribution made by various researchers in the field of SDP using machine learning (ML) algorithms deploying ensemble approach to class-imbalance problems in order to get accurate models for software defect prediction (SDP). Table 1 shows the current trends to tackle class imbalance issue in SDP. The table is headed with the year of publication of the referenced work, the technique used in the work, the dataset(s) and its feature space considered in the respective research along with the performance measurement criteria adopted. The last column added in the table is the observation drawn by the authors of this candidate work.

Corresponding to each study, we have made some observations which are added in Table 1 (as last column). After reviewing the literature in multiple dimensions, we identified that the existing studies results are sub-optimal. All ensembles are existing traditional ensembles.

Some other observations made from the literature review are—1) Majority of research in the field of SDP has been carried out by utilizing publicly available datasets namely NASA Metrics Data Program and PROMISE Data Repository which comprises almost 67% of total research work carried out in past three decades, (2) the most popular evaluation metrics among software practitioners for SDP evaluation are AUC, ROC and accuracy, (3) ANN, SVM are the two most popular classifiers for software defect prediction, (4) class imbalance majorly hinders the performance of classifiers. And (5) ensembles are robust enough and possess built-in capacity to deal with class imbalance of defect dataset.

In the next section of our paper, the research methodology which is adopted for this paper is explained and the research gaps are reported as well-formed research questions.

3 Research methodology

In this section, we report the methodology adopted to carry out the research work. First up, we formulate the research questions in an empirical way to steer the research work in a systematic way. Then, we describe the configuration of the proposed stacked ensemble and the working algorithm. The experimental set-up adopted for this work along with the parameter settings for the experimental model are also discussed in detail.

3.1 Research questions

To steer the research in a systematic way, we address the following research questions:

RQ1. Does the proposed heterogenous stacked ensemble empirically outperform the existing single classifiers?
This RQ deals with comparison of proposed model with the traditional models in order to ensure that proposed model has the potential to predict the buggy modules effectively. For this purpose, five most popular classifiers from the literature are selected for comparative study. These five classifiers are artificial neural network, nearest neighbor, tree based classifier, Naïve Bayes and support vector machines. The reasons for selecting these classification algorithms are—(1) the popularity of these classifiers in SDP

[20], (2) effective prediction power in SDP domain [23] and (3) these are base classifiers of our proposed ensemble. For comparative analysis in this specific aspect, the study Goyal and Bhatia (2020) [6] is selected.

RQ2. Does the proposed customized stacked ensemble empirically outperform the state-of-the-art ensemble based SDP classifiers?

It is to investigate into the prediction power of proposed model in comparison to the state-of-the-art ensemble based SDP models. For the comparative analysis, homogenous ensemble and heterogeneous ensemble based classifiers are selected. The study Balogun et al. [1] is selected for comparison over homogenous ensemble based SDP classifiers and the study Khuat et al. [11] is selected for comparison with heterogenous ensemble based SDP classifiers.

RQ3. Are the answers to the above mentioned RQs statistically valid?

This is the most crucial RQ as it confirms that the answers to above stated RQs are valid. Proper statistical tests are selected and conducted for the statistical evidence. The Friedman test has been found suitable and hence conducted to find the statistical proof for the study.

3.2 Proposed stacked ensemble classifier

We propose stacking based ensemble combining heterogenous base learning classifiers. We use five most popular SDP classifiers from the literature namely support vector machine (SVM), artificial neural networks (ANN), naïve bayes (NB), nearest neighbor and decision trees (DT) [4, 6, 7, 17] as base learners. Then, a neural network is selected as a meta-model for this work which takes the predictions made by base classifiers (called 'Level-1' data) as inputs and returns the final predicted outputs.

The choice of base-learners and meta-model is very trivial from the literature survey for the work contributed for the SDP domain during past three decades [23]. The selection of neural network as a meta-model is to non-linearly combine [9] the predictions from the base classifiers to bring the best combination of powers and produce the most accurate final predictions regarding whether the candidate module is 'buggy' or 'clean' out of this synergism of powers.

The proposed stacked ensemble based SDP classifier is modelled as in Fig. 1. The proposed model works on the algorithm which is stated as below-

3.3 Experimental set-up

In this paper, the MATLAB™ R2019a is used for carrying out the processing and computational tasks. It is installed on Windows™ 10 Pro, Intel® Core™ i5-8265U CPU with RAM storage of 8 GB. All of the rigorous sets of experiments including data preprocessing through fitting the classifiers and validation of classifiers are executed over the same hardware and software platform. The performance for the proposed classifier is measured over selected five datasets for every selected performance evaluation criterion which includes AUC, ROC, accuracy. The data is partitioned into training dataset and testing dataset using k-fold cross validation with $k = 10$. The training subset is used to train the stacked ensemble classifier and then it is tested for testing dataset. All the experiments are performed on the above experiment set-up and design following classifiers at two levels of model (shown in Table 2).

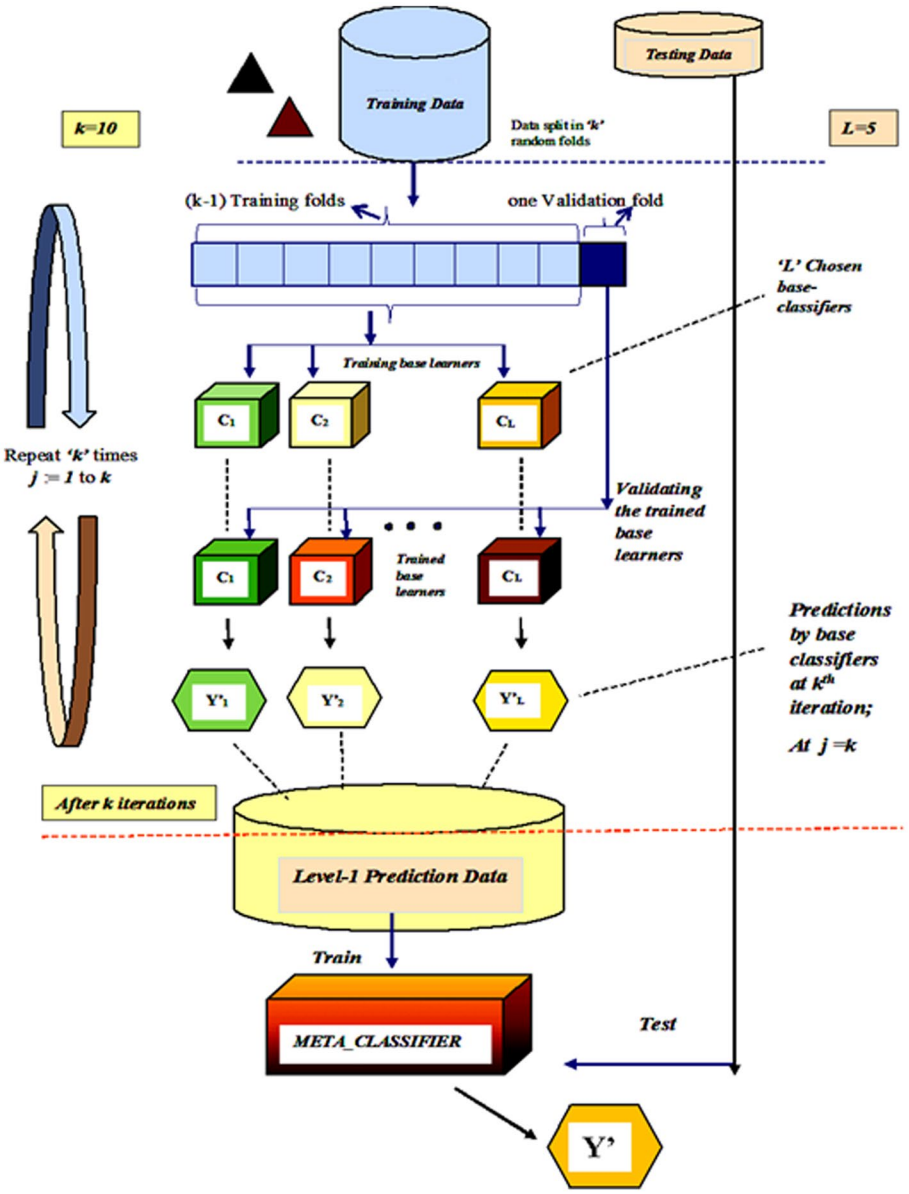


Fig. 1 Proposed stacked ensemble SDP model

The choice of base-learners and meta-model is very trivial from the literature survey for the work contributed for the SDP domain during past three decades [23]. At level-2 of stacking, neural network is utilized due to its robust capability of learning non-linear relationships among the inputs [9]. The selection of neural network as a meta-model is to non-linearly combine the predictions from the base classifiers to bring the best combination of powers and produce the most accurate final predictions regarding whether the candidate module is 'buggy'

Table 2 Description of 5 base-learners and 1 meta-learner (level-wise)

S. No	Classifier	Learner type	Level
1	Artificial neural networks (ANN)	Base learner	Level-1
2	Decision trees (DT)	Base learner	Level-1
3	Naïve bayes (NB)	Base learner	Level-1
4	Support vector machine (SVM)	Base learner	Level-1
5	k-nearest neighbor (k-NN)	Base learner	Level-1
6	Artificial neural networks (ANN)	Meta learner	Level-2

Table 3 Parameter settings for base-learners and meta-learner

Classifier	Parameter: value;
Artificial neural networks (ANN) At Level-1 (base-learner)	Input Layer Size: 4 neurons; Hidden Layer Size: 6 neurons; Output Layer Size: 2; Num of Hidden Layers:1; Training Function: trainscg (scaled conjugate gradient); Performance Function: Cross-Entropy
Decision trees (DT) (base-learner)	Algorithm: CART; tenfold CV
Naïve bayes (NB) (base-learner)	tenfold CV
Support vector machine (SVM) (base-learner)	Kernel: Radial Basis Function; Algorithm: SMO (Sequential Minimal Optimization); tenfold CV
k-nearest neighbor (k-NN) (base-learner)	K = 5; Distance Criteria: Euclidean measure; tenfold CV
Artificial neural networks (ANN) At Level-2 (meta-learner)	Input Layer Size: 5 neurons; Hidden Layer Size: 5 neurons; Output Layer Size: 2; Num of Hidden Layers:1; Training Function: trainscg (scaled conjugate gradient); Performance Function: Cross-Entropy

or 'clean' out of this synergism of powers. The parameter settings for the proposed model is given in Table 3.

For comparative analysis, rigorous experiments are conducted following the same process including the parameter settings, tools and environment as deployed by the selected studies to ensure the fair comparison of performance [1, 6, 11]. All eight models (5 traditional ML SDP models + 3 Ensemble based classifiers) are synthesized, and experiments are repeated for all five datasets. Then, the performance is recorded over three selected evaluation criterion (ROC, AUC, Accuracy) and comparison is made statistically. The SDP models selected for comparative analysis are listed as in Table 4.

Table 4 Details of 8 SDP Classifiers Selected for Comparative Analysis

S. No	Classifier	Reference Study
1	Artificial neural networks (ANN)	Goyal and Bhatia (2020) [6]
2	Decision trees (DT)	Goyal and Bhatia (2020) [6]
3	Naïve bayes (NB)	Goyal and Bhatia (2020) [6]
4	Support vector machine (SVM)	Goyal and Bhatia (2020) [6]
5	k-nearest neighbor (k-NN)	Goyal and Bhatia (2020) [6]
6	Bagging based ensemble model (Bagging)	Balogun et al. (2020) [1]
7	Boosting based ensemble model (Boosting)	Balogun et al. (2020) [1]
8	Heterogenous ensemble with 9 base learners (Heterogenous ensemble)	Khuat et al. (2021) [11]

3.4 Mathematical background

Naïve Bayes Classifier makes classification utilizing the probability theory from the statistics. Bayes rule is applied to predict whether the module is buggy or not. It predicts that the test sample data-point belongs to that particular class which is having the highest posterior probability for that sample data-point. Suppose for defect prediction problem, vector $\boldsymbol{x}$ denotes the attribute set and $\boldsymbol{y}$ is a set with two elements {buggy, clean}; denotes the classes to which each data-point uniquely belongs. Naïve bayes classifier predicts that a specific module with attribute vector $\boldsymbol{x}$ belongs to 'buggy' class only if Eq. (1) satisfies. Otherwise, it predicts that the module belongs to 'clean' class.

$$P(buggy|\boldsymbol{x}) \geq P(clean|\boldsymbol{x}) \tag{1}$$

In Eq. (1), $P(buggy|\boldsymbol{x})$ denotes the posterior probability of class buggy, after having seen $\boldsymbol{x}$ and $P(clean|\boldsymbol{x})$ denotes the posterior probability of class clean, after having seen $\boldsymbol{x}$. Equation (1) shows that for two class classification problem, whichever class will be having highest posterior probability will be predicted by the classifier for given $\boldsymbol{x}$. The posterior probability for any class can be computed using Bayes Rule as given in Eq. (2). Equation (2) can be rewritten as Eq. (3) for class buggy and as Eq. (4) for class clean.

$$Posterior = \frac{Prior \times Likelihood}{Evidence} \tag{2}$$

$$\mathrm{P(buggy|x)} = \frac{\mathrm{p(x|buggy)} \times \mathrm{P(buggy)}}{\mathrm{p(x)}} \tag{3}$$

where p(x|buggy) denotes the prior probability for x; the probability of seeing x as input when it is known that it belongs to buggy class; satisfying inequation (5) and Eq. (6).

$$\mathrm{P(clean|x)} = \frac{\mathrm{p(x|clean)} \times \mathrm{P(clean)}}{\mathrm{p(x)}} \tag{4}$$

where p(x|clean) denotes the prior probability for x; the probability of seeing x as input when it is known that it belongs to clean class; satisfying inequation (5) and Eq. (6).

$$\mathrm{P(buggy)} \geq 0,\ \mathrm{P(clean)} \geq 0 \tag{5}$$

$$\mathrm{P(buggy)} + \mathrm{P(clean)} = 1 \tag{6}$$

And, p(x) denotes the Evidence which is the marginal probability that $\boldsymbol{x}$ is seen, regardless it belongs to buggy class or clean class. It can be computed as Eq. (7).

$$\mathrm{p(x)} = \mathrm{p(x|buggy)} \times \mathrm{P(buggy)} + \ \mathrm{p(x|clean)} \times \mathrm{P(clean)} \tag{7}$$

Equation (2) which represents Bayes rule is the basis for Naïve Bayes classifier. By applying the values from Eq. (3), (4) and (5) into Eq. (1), the prediction for given data-point that whether it belongs to 'buggy' class or not; can be made.

K-Nearest Neighbors is another classification algorithm from statistics. It uses similarity between data-points to predict the class. In our experimental set-up, we utilize Euclidean distance which can be computed between any two data-points namely $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ as Eq. (8). Suppose for defect prediction problem, vector $\boldsymbol{x}$ denotes the attribute set and $\boldsymbol{y}$ is a set with two elements {buggy, clean}; denotes the classes to which each data-point uniquely belongs.

$$D\left(x_i, x_j\right) \ = \ \sqrt{\sum_{i=1}^{k}(x_{ik} - x_{jk})^2} \tag{8}$$

Assume buggy is denoted with '+1' and clean with '-1', hence y = {+1,-1}. For the instance, $\boldsymbol{x_q}$, K-NN will make classification using the Eq. (9) after computing the 'k' nearest neighbors of $\boldsymbol{x_q}$ using Eq. (8). Suppose $\mathrm{N_k}$ denotes the set of 'k' neighbors of $\boldsymbol{x_q}$.

$$\hat{y}_{\mathrm{q}} \ = \ \mathrm{sign}\left(\sum_{\mathrm{x_i} \in \mathrm{N_k}} \mathrm{y_i}\right) \tag{9}$$

Decision Trees based classifiers are built using Classification and Regression Trees (CART) algorithm. Decision trees are hierarchical, non-parametric, supervised machine learning models. A tree is comprised of few internal nodes with decision functions and external leaves. In our experiments, we used the 'entropy' as a measure of impurity which in turn records the goodness of split. Let us compute entropy for a node in classification tree say node 'a'; $\mathrm{N_a}$ denotes the number of instances that reaches to node 'a'; N_a^{buggy} and N_a^{clean} denotes the number of nodes in $\mathrm{N_a}$ that belongs to class 'buggy' and class 'clean' respectively. Suppose an instance reaches node 'a' then its chances of being 'buggy' is given as Eq. (10). Similarly, its chances of being 'clean' is computed using Eq. (11). Entropy is computed as Eq. (12) for 2-class classification problem.

$$p_a^{buggy} = \frac{N_a^{buggy}}{N_a} \tag{10}$$

$$p_a^{clean} \ = \ \frac{N_a^{clean}}{N_a} \tag{11}$$

$$\text{Entropy (node 'a')} \ = \ -\left(\left(p_a^{buggy}\right)\log\left(p_a^{buggy}\right) + p_a^{clean}\log\left(p_a^{clean}\right)\right) \tag{12}$$

Artificial neural networks are implemented with standard feed-forward, error backpropagation algorithm. For $\boldsymbol{n}$-feature input data $\boldsymbol{X} = <\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}>$, there are $\boldsymbol{n}$ input neurons. For sigmoid activation function, the output $\hat{y}_i$ for $\boldsymbol{i^{th}}$ neuron is computed using Eq. (13). In this way, features are fed in forward direction from input layer to hidden layer, then from hidden to output layer. The computed output at output neuron is compared with the actual output and the error is computed as Eq. (14) as half of the sum of squares of difference between the actual output and predicted output and the error is back propagated to update weights as per Eq. (15) and learning takes place in this way to minimize the error.

$$\hat{y}_i = sig(\sum_{i=1}^{n} w_i x_i + w_o) \tag{13}$$

where w_i denotes weight for $\boldsymbol{i^{th}}$ neuron and w_o denotes the bias;

$$\text{error} = \frac{1}{2} \sum_{m} \sum_{i}^{n} (y_i - \hat{y}_i)^2 \tag{14}$$

where m denotes number of output neuron.
$\Delta w = \eta$. error. input signal

$$\eta . \frac{1}{2} \sum_{m} \sum_{i}^{n} (y_i - \hat{y}_i)^2 \cdot x_i \tag{15}$$

where η denotes learning rate.

Support vector machine works on Vapnik theory of maximum marginal methods. We used the RBF kernel setting for SVM. For 'n' instances denoted as $<X_i, y_i>$, it finds the optimal separating hyperplane between two classes denoted {buggy as +1,clean as -1} by finding w_1 and w_2 which satisfies Eq. (16).

$$y(w_2 x + w_{1)} \geq \pm 1 \tag{16}$$

SVM solves the optimal hyperplane problem by Langrangian multipliers. First, new higher dimensional mapping is achieved with function ϕ as Eq. (17) shown.

$$y = w^T \phi(x) + c \tag{17}$$

where w is weight vector and c is scalar.
The SVM has to be optimize Eq. (18)

$$\text{Minimize } \frac{1}{2} w^T w + \rho \frac{1}{2} error_i^2 \tag{18}$$

$$\text{Subject to } y = w^T \phi(x) + c + error$$

where ρ denotes the cost function.

After solving this, the prediction made by SVM classifier can be given as Eq. (19) in terms of kernel.

$$Y = \sum \left(\alpha - \alpha^T\right) . K\left(x_{centre}, x\right) + b \tag{19}$$

In Eq. (20) $K\left(x_{centre}, x\right)$ denotes kernel based on Radial basis function. In our experiments we have used RBF kernel for SVM where the centre and radius are defined by the user.

$$K\left(x_{centre}, x\right) = e^{-\frac{\left|x_{centre}-x\right|^2}{2.(radius)^2}} \tag{20}$$

4 Dataset and evaluation criteria used

In this section, the highlights on the dataset and metrics used for experimentation are brought. The performance evaluation metrics opted to measure the performance of proposed stacked ensemble based SDP model and for comparative analysis among the selected models are described.

4.1 Dataset and software metrics

The Dataset used for the experimental study is NASA defect dataset which are available publicly in PROMISE repository. The data metrics are collected from NASA projects. The experiment is designed using five datasets—CM1, KC1, KC2, PC1, and JM1. McCabe and Halstead features extractors are used to collect the data [19, 21]. Table 5 shows the used dataset name, total instances in the dataset, number of instances which are buggy and number of instances which are clean. The datasets are comprising of the most popular static code metrics. All five datasets possess 21 metrics and 1 response variable.

4.2 Performance evaluation criteria

The performance of proposed stacked ensemble is evaluated using the widely accepted evaluation metrices namely Confusion matrix, ROC, AUC, Accuracy and recall [2, 7, 10, 20, 26, 27]. These can be defined as-

- *Confusion matrix* is in the form of a matrix whose individual cell contains necessary information for performance evaluation of the classifier.

  As shown in Fig. 2a., the class '*buggy*' is considered as positive class and class '*clean*' is considered as negative class. The term '*True Positive*' refers to the 'count of modules' which are buggy in actual and classified as buggy by the classifier. The term '*True Negative*' refers to the 'count of modules' which are clean in actual dataset and predicted as clean by the classifier. It leads to two other terms which are

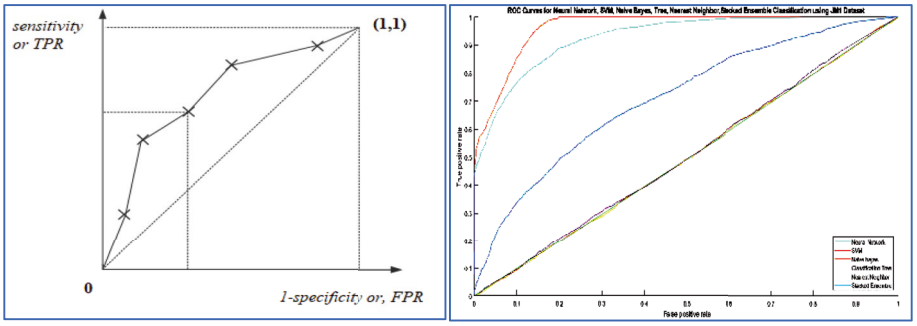
Table 5 Dataset description [2]

#	Data-set name	# Total instances	# Buggy instances	# Clean instances
1	CM1	498	49	449
2	KC1	2109	326	1783
3	KC2	522	107	415
4	PC1	1109	77	1032
5	JM1	10,885	2106	8779

ACTUAL CLASS	Classifier	
BUGGY	$True\_Positive_{Buggy>>Buggy}$	$False\_Negative_{Buggy>>Clean}$
CLEAN	$False\_Positive_{Clean>>Buggy}$	$True\_Negative_{Clean>>Clean}$

Confusion Matrix	BUGGY	CLEAN
	PREDICTED CLASS	

a



b **c**

Fig. 2 **a** Confusion matrix. **b** ROC. **c** Multiple ROCs

'*False Positive*' and '*False Negative*'. The '*False Positive'* refers to the 'count of modules' which belong to clean class in actual dataset and predicted as buggy by the classifier in consideration. The '*False Negative*' means those modules which are buggy in actual dataset and predicted as clean by the classifier.

- The sensitivity (*true positive rate or TPR*) and specificity (*1- false positive rate or 1-FPR*) are computed as *Eq. (21) and (22). True positive rate*, *TPR*, can be thought as hit rate, accounts for what proportion of buggy modules we correctly predict and *false positive rate*, *FPR*, refers to the proportion of clean modules we wrongly accept as buggy.

$$sensitivity(orrecall) = \frac{truepositive}{truepositive + falsenegative} \tag{21}$$

$$specificity = \frac{truenegative}{truenegative + falsepositive} \tag{22}$$

- *Receiver Operating Characteristics (ROC) curve* is plot of TPR (as y-axis) and FPR (as x-axis) (see Fig. 2b. and Fig. 2c). It is interpreted that closer the classifier gets to the upper left corner, better is its performance. To compare the performance of classifiers, the one above the other is considered better.

Table 6 Performance comparison of stacked ensemble classifier with base classifiers (in AUC)

AUC	ANN [6]	SVM [6]	NB [6]	TREE [6]	KNN [6]	Stacked ensemble
CM1	0.7286	0.6341	0.6592	0.5289	0.5868	0.7618
JM1	0.7878	0.7220	0.7442	0.6828	0.5741	0.789
KC1	0.8315	0.8021	0.7816	0.7104	0.6887	0.853
KC2	0.7187	0.6348	0.6053	0.5863	0.6050	0.824
PC1	0.7102	0.6612	0.6262	0.6227	0.5268	0.7644
Average	0.75536	0.69084	0.6833	0.62622	0.59628	0.79844

Table 7 Performance comparison of stacked ensemble classifier with base classifiers (in ACCURACY)

Accuracy	ANN [6]	SVM [6]	NB [6]	TREE [6]	KNN [6]	Stacked ensemble
CM1	0.8996	0.8996	0.8735	0.8494	0.8614	0.926
JM1	0.8482	0.8482	0.8307	0.8373	0.275	0.891
KC1	0.8371	0.7911	0.8409	0.8065	0.8084	0.8454
KC2	0.9296	0.9296	0.9017	0.9089	0.7682	0.9212
PC1	0.8107	0.8037	0.8061	0.7793	0.3930	0.847
Average	0.86504	0.85444	0.85058	0.83628	0.6212	0.88612

- *Area Under the ROC Curve (AUC)* gives the averaged performance for the classifier over different situations. AUC = 1 is considered ideal.
- *Accuracy* is computed as *Eq. (23)*

$$Accuracy = \frac{truepositive + truenegative}{truepositive + falsepositive + truenegative + falsenegative} \tag{23}$$

5 Result analysis and discussion

In this section, we report the results recorded in the experimental study. We also find the answers to the Research Questions (RQs) following an analytical approach. Let us discuss all three RQs one by one in upcoming sub-sections.

5.1 Finding the answer to RQ1-

RQ1. Does the proposed heterogenous stacked ensemble empirically outperform the existing single classifiers?

To answer RQ1, first up, we recorded the performance of all six classification algorithms (ANN, SVM, NB, KNN, Tree and stacked ensemble) which are selected in this study on all five datasets in terms of AUC and Accuracy reported in Tables 6 and 7 respectively. The ROC curves are also reported for comparison as in Fig. 3.

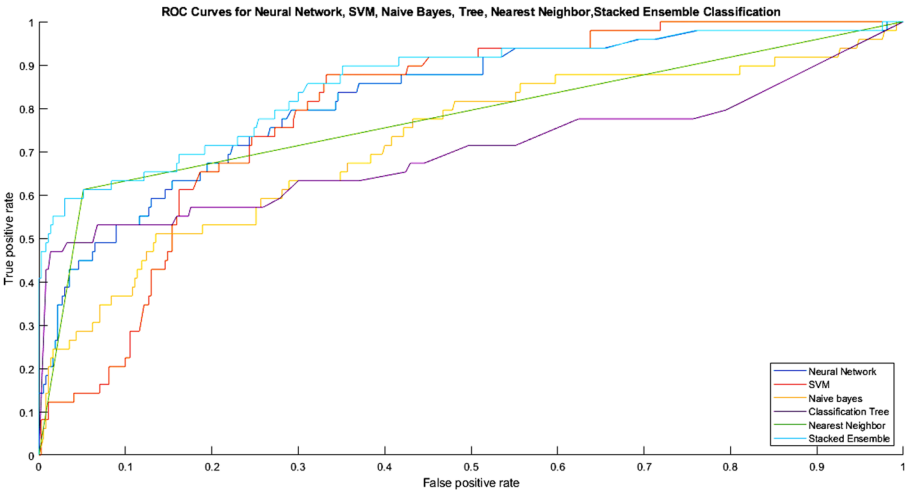


Fig. 3 ROC Curve for all six classifiers over five datasets

From the reported results in Tables 6 and 7, it can be seen that proposed model shows better performance than base classifiers (highest values are shown in bold faces). The drawn inferences are-

i. It is better than ANN by 4%, SVM by 10%, NB by 11%, Tree by 17% and KNN by 20% in terms of AUC.
ii. It is better than ANN by 2%, SVM by 3%, NB by 3%, Tree by 4% and KNN by 2% in terms of Accuracy.
iii. The proposed model shows best ROC curve among all six classifiers.

Further, the results recorded for AUC measure and Accuracy measure for the candidate classifiers are plotted as box plots for better visualization and analysis (shown in Fig. 4a and Fig. 4b respectively). From the figures, we can easily analyse the classifiers in a comparative manner. It is noted that the technique having high value of median with fewer outliners performs better than other classification algorithms. It is evident from the figures and plots that the proposed model outperforms all 5 base classifiers in AUC, Accuracy and ROC metrics.

The average performance of all 5 base classifiers and the proposed stacked ensemble based SDP classifier is plotted as Fig. 5. It can be inferred that Proposed Stacked Ensemble Model outperforms Base Classifiers on average by 12% in AUC and by 8% in Accuracy.

ANSWER to RQ1- From the results and analysis, YES! The proposed model outperforms the single base classifiers empirically.

5.2 Finding the answer to RQ2-

RQ2- Does the proposed customized stacked ensemble empirically outperform the state-of-the-art ensemble based SDP classifiers?

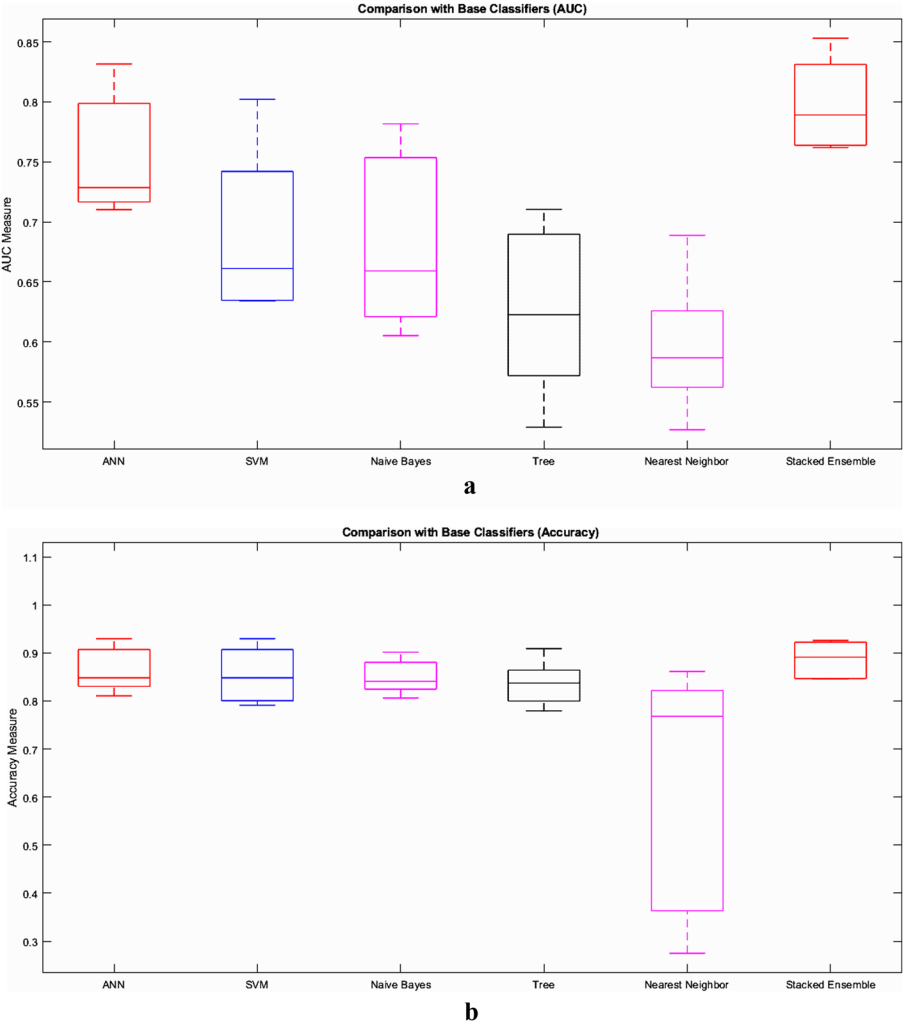


Fig. 4 **a** AUC Box Plots for all six classifiers over five datasets. **b** Accuracy Box Plots Curve for all six classifiers over five datasets

To answer this RQ, we further need to compare the performance of proposed stacked ensemble over the state-of-the-art of ensemble based SDP models. We selected two empirical studies with 3 different ensemble based SDP models for comparative analysis- 1) Balogun et al. (2020) [1] deployed standard ensemble techniques-Bagging and Boosting. 2) Khuat et al. (2021) [11] deployed heterogenous ensembles using 9 base classifiers. Tables 8 and 9 report the comparative analysis between the proposed model and the state-of-the-art models in terms of AUC and Accuracy respectively.

It is clear form the results recorded in the Tables 8 and 9 that Stacked Ensemble performs better than the state-of-the-art ensemble classifiers (highest values are reflected with bold faces). Further, for comparison, the ROC curve is plotted for all 4 SDP models as shown in Fig. 6.

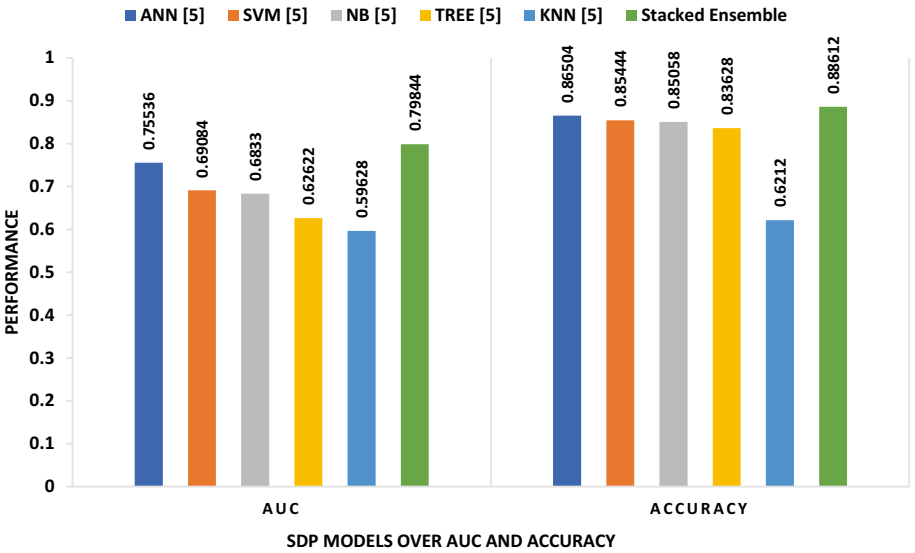


Fig. 5 Proposed Stacked Ensemble Model outperforms Base Classifiers by 12% in AUC and by 8% in Accuracy

Table 8 Performance comparison of stacked ensemble classifier with state-of-the-art ensembles (in AUC)

AUC	Bagging [1]	Boosting [1]	Heterogenous [11]	Stacked ensemble
CM1	0.712	0.706	0.751	0.7618
JM1	0.723	0.747	0.745	0.789
KC1	0.702	0.735	0.812	0.853
KC2	0.782	0.80	0.821	0.824
PC1	0.745	0.709	0.753	0.7644
Average	0.7328	0.7394	0.7764	0.79844

Table 9 Performance comparison of stacked ensemble classifier with state-of-the-art ensembles (in ACC URACY)

Accuracy	Bagging [1]	Boosting [1]	Heterogenous [11]	Stacked ensemble
CM1	0.798	0.799	0.837	0.926
JM1	0.728	0.732	0.789	0.891
KC1	0.783	0.795	0.825	0.8454
KC2	0.802	0.813	0.892	0.9212
PC1	0.798	0.736	0.801	0.847
Average	0.7818	0.775	0.8288	0.88612

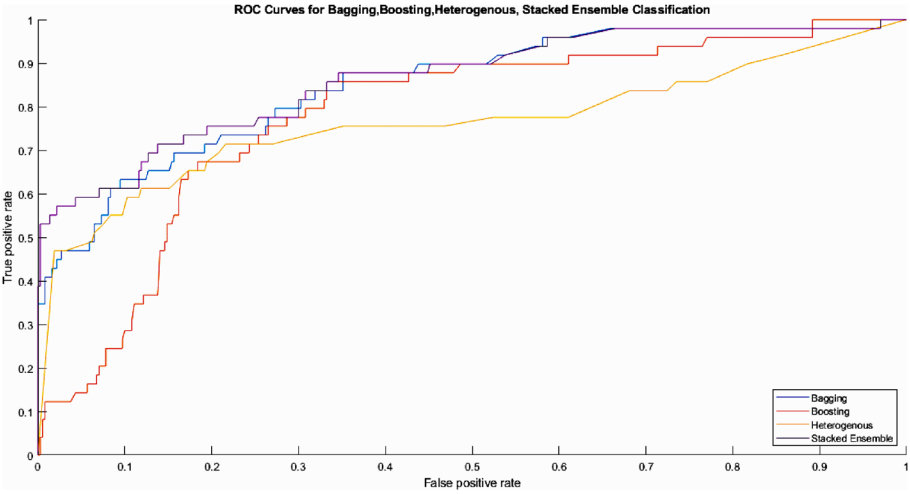


Fig. 6 ROC Curve for all four classifiers over five datasets

The drawn inferences are-

i. The proposed stacked ensemble based classifier outperforms the Bagging, Boosting and Heterogenous model by 6%, 5%, and 2% in terms of AUC.
ii. The proposed stacked ensemble based classifier outperforms the Bagging, Boosting and Heterogenous model by 10%, 11%, and 5% in terms of Accuracy.
iii. The best ROC is shown by proposed stacked model among all 4 classifiers.

Further, the results recorded for AUC measure and Accuracy measure for the candidate classifiers are plotted as box plots for better visualization and analysis (shown in Fig. 7a and b respectively). From the figures, we can easily analyse the classifiers in a comparative manner. It is noted that the technique having high value of median with fewer outliners performs better than other classification algorithms. It is evident from the figures and plots that the proposed model outperforms all state-of-the-art ensemble methods in AUC, Accuracy and ROC metrics.

The average performance of all 3 ensembles (from literature) and the proposed stacked ensemble based SDP classifier is plotted as Fig. 8. It can be inferred that Proposed Stacked Ensemble Model outperforms Base Classifiers on average by 4% in AUC and by 9% in Accuracy.

ANSWER to RQ2- From the results and analysis, YES! The proposed model outperforms the state-of-the-art ensemble classifiers empirically.

5.3 Finding the answer to RQ3

RQ3. Are the answers to the above mentioned RQs statistically valid?

From the above experimental results, analysis and inferences; *proposed stacked Ensemble based classifier is the best SDP classifier among all 8 selected SDP models from the literature*.

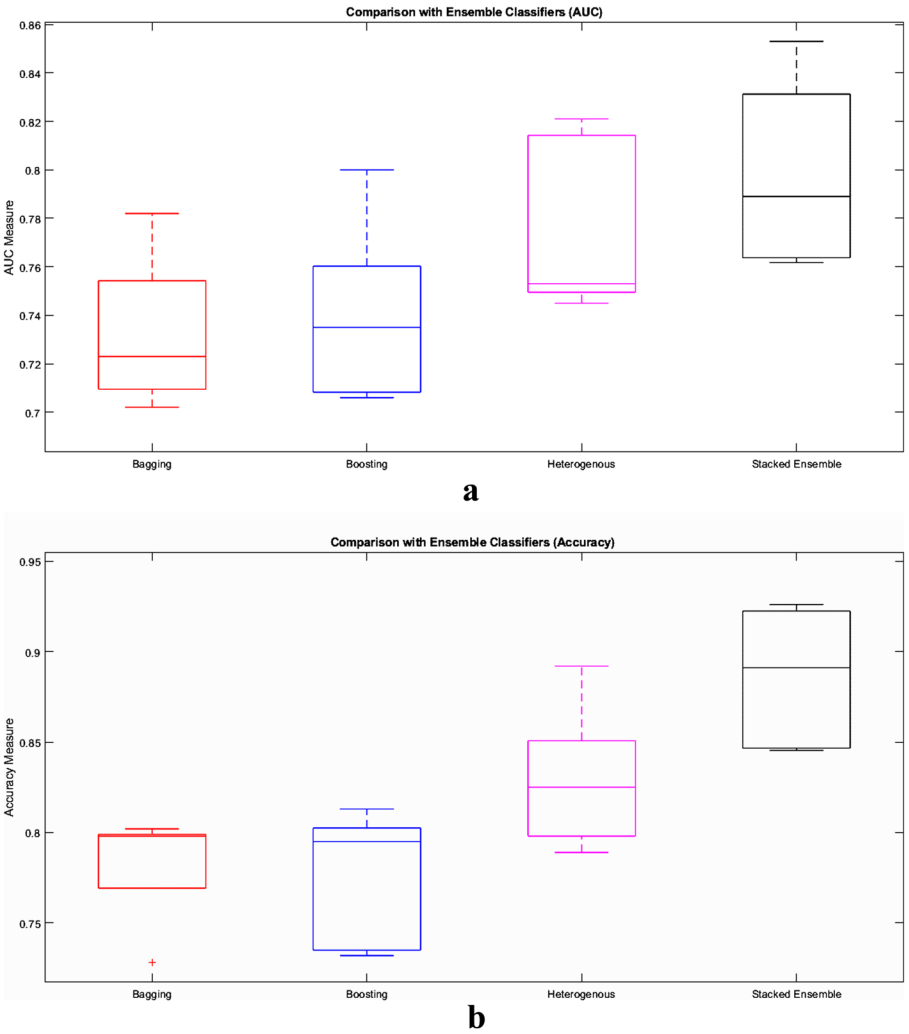


Fig. 7 **a** ROC Curve for all four classifiers over five datasets. **b** ROC Curve for all four classifiers over five datasets

Before drawing any final conclusions, we statistically validated the inferences drawn in above two subsections and sought the statistical evidence to the answers reported for RQ1 and RQ2. The Friedman's test is found suitable for non-parametric comparison among more than two samples [14, 23].

In respect to RQ1, we assume H0: "The performance reported by stacked ensemble and the performance reported by other 5 base classifiers are not different". And the alternate hypothesis- H1: 'The performance reported by stacked ensemble and the performance reported by other 5 base classifiers are different".

We conducted the test with 95% of confidence. The results of the statistical tests are shown in Fig. 9. It can be seen clearly that the value of p-static is 0.0058 which is smaller than 0.05.

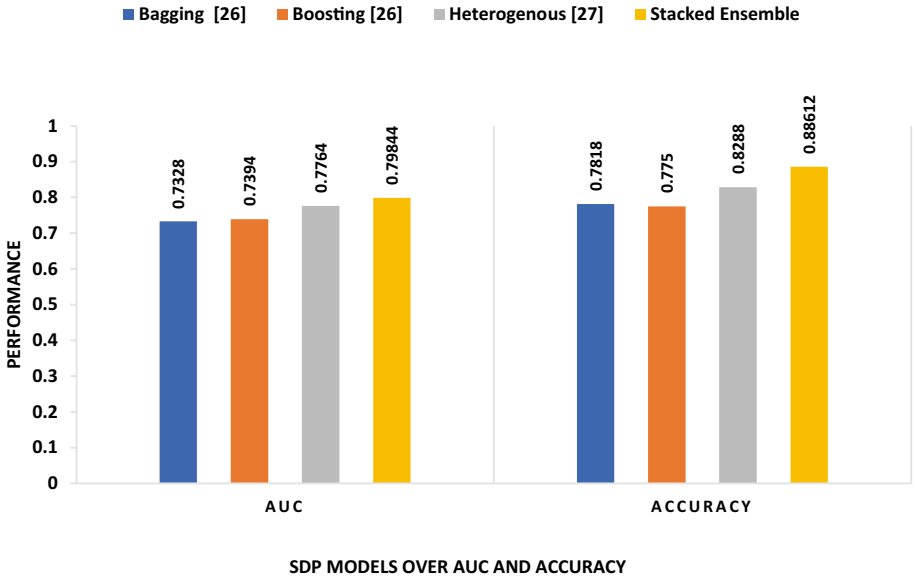


Fig. 8 Proposed Stacked Ensemble Model outperforms Base Classifiers by 4% in AUC and by 9% in Accuracy

Friedman's ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	56.4	5	11.28	16.4	0.0058
Error	29.6	20	1.48		
Total	86	29			

Fig. 9 Friedman Test with p-static 0.0058

Friedman's ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	23.4	3	7.8	14.04	0.0029
Error	1.6	12	0.13333		
Total	25	19			

Fig. 10 Friedman Test with p-static 0.0029

It means, the null hypothesis: H0 is to be rejected and alternate hypothesis:H1 is to be accepted. (It can be inferred that -Answer reported to RQ1 in Sect. 5.1- The proposed stacked ensemble outperforms the single base classifiers over all datasets- is statistically validated.

In respect to RQ2, we assume H0: "The performance reported by stacked ensemble and the performance reported by other 3 state-of-the-art ensemble classifiers are not different". And the alternate hypothesis- H1: 'The performance reported by stacked ensemble and the performance reported by other 3 state-of-the-art ensemble classifiers are different".

We conducted the test with 95% of confidence. The results of the statistical tests are shown in Fig. 10. It can be seen clearly that the value of p-static is 0.0029 which is smaller than 0.05.

It means, the null hypothesis: H0 is to be rejected and alternate hypothesis:H1 is to be accepted. (It can be inferred that -Answer reported to RQ2 in Sect. 5.2- The proposed stacked ensemble outperforms state-of-the-art ensemble classifiers over all datasets- is statistically validated.

ANSWER to RQ3- From the results and analysis, YES! The answers to RQ1 and RQ2 are statistically valid.

6 Conclusion

In this paper, we proposed a novel heterogenous ensemble utilizing the stacking methodology and deployed it for effective software defect prediction (SDP). The proposed model is robust enough to handle the defect dataset having class imbalance issues. Software defect prediction plays an important role in targeting the testing efforts to the faulty modules and hence to save time and cost. From literature, it is evident that plethora of ML based SDP models have been contributed by the researchers. The imbalance nature of defect datasets has always been a hurdle in achieving a good classification accuracy. The class imbalance means the number of instances belonging to one class outnumbers the number of instances of other class. The outnumbering instances introduce biasing in the classification algorithms and hinder the performance. Due to this reason, the traditional ML based SDP models result sub-optimal results when trained with imbalanced datasets. We found the studies reported in the literature dealing with class imbalance using ensemble techniques as ensembles have built-in capacity to deal with the class imbalance nature of datasets. Still there is huge scope for the improvement in the accuracy of ensemble based defect predictors.

This work is dedicated to build an effective SDP classifier using heterogenous stacking ensemble method. It has been built upon the five best classifiers (reported from literature) as the base classifiers at level-1, then utilizing two level stacking and at level-2, ANN algorithm has been used to combine the outputs from heterogenous classifiers of level-1. The performance of proposed model is empirically evaluated over three most popular criteria for evaluation—AUC, ROC and accuracy. A statistical comparison is also made among the performances of the proposed stacked ensemble classifier with that of the 5 base classifiers and 3 state-of-the-art ensemble techniques. From the reported results, it can be inferred that the proposed model built up of two-level stacking of heterogeneous ensemble (with 5 base classifiers) at level-1 and ANN at level-2; is best with highest value for AUC measure (85.3 = %) with best ROC curve and highest value for accuracy measure (= 92.6%). In future, we propose to replicate the study other defect datasets extracting from live projects.

Declarations

Conflict of interest Authors have no Conflicts of interest/Competing interests.

References

1. Balogun AO, Lafenwa-Balogun FB, Mojeed HA, Adeyemo VE, Akande ON, Akintola AG, Bajeh AO, Usman-Hamza FE (2020) SMOTE-Based Homogeneous Ensemble Methods for Software Defect Prediction. Computational Science and Its Applications – ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020. Proceedings, Part VI 12254:615–631. https://doi.org/10.1007/978-3-030-58817-5_45
2. Boucher A, Badri M (2018) Software metrics thresholds calculation techniques to predict fault-proneness: an empirical comparison. Inf Softw Technol 96:38–67
3. Chen L, Fang B, Shang Z et al (2018) Tackling class overlap and imbalance problems in software defect prediction. Softw Qual J 26:97–125. https://doi.org/10.1007/s11219-016-9342-6
4. Erturk E, Sezer EA (2015) A comparison of some soft computing methods for software fault prediction. Expert Syst Appl 42:1872–1879
5. Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F (2011) A review on ensembles for the class imbalance problem: bagging-, boosting- and hybrid-based approaches. IEEE Trans Syst Man Cybernetics Part C 42(4):463–484
6. Goyal S, Bhatia P (2020) Comparison of machine learning techniques for software quality prediction. Int J Knowl Syst Sci IJKSS 11(2):21–40. https://doi.org/10.4018/IJKSS.2020040102
7. Goyal S, Bhatia PK (2020) Empirical software measurements with machine learning. In: Bansal A, Jain A, Jain S, Jain V, Choudhary A (eds) Computational intelligence techniques and their applications to software engineering problems, pp 49–64. CRC Press, Boca Raton. https://doi.org/10.1201/9781003079996
8. Haixiang G, Yijing Li, Jennifer Shang Gu, Mingyun HY, Bing G (2017) Learning from class-imbalanced data: review of methods and applications. Expert Syst Appl 73:220–239
9. Haykin S (2010) Neural networks and learning machines, 3/e. PHI Learning, India
10. Huda S, Liu K, Abdelrazek M, Ibrahim A, Alyahya S, Al-Dossari H, Ahmad S (2018) An Ensemble Oversampling Model for Class Imbalance Problem in Software Defect Prediction. IEEE Access 6:24184–24195. https://doi.org/10.1109/access.2018.2817572
11. Khuat TT, Le MH (2020) Evaluation of Sampling-based ensembles of classifiers on imbalanced data for software defect prediction problems. SN Comput Sci 1:108. https://doi.org/10.1007/s42979-020-0119-4
12. Laradji IH, Alshayeb M, Ghouti L (2015) Software defect prediction using ensemble learning on selected features. Inf Softw Technol 58:388–402
13. Lee HK, Kim SB (2018) An overlap-sensitive margin classifier for imbalanced and overlapping data. Expert Syst Appl 98:72–83
14. Lehmann EL, Romano JP (2008) Testing statistical hypothesis: springer texts in statistics. Springer, New York
15. Miholca, D., G., Czibula, I., Czibula. A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. J. Information Sciences. Feb 2018
16. (NASA 2015) https://www.nasa.gov/sites/default/files/files/Space_Math_VI_2015.pdf. Accessed 23 Aug 2018
17. Ozakıncı R, Tarhan A (2018) Early software defect prediction: ¨a systematic map and review. J Syst Softw 144:216–239. https://doi.org/10.1016/j.jss.2018.06.025
18. Rathore S, Kumar S (2017) Towards an ensemble-based system for predicting the number of software faults. Expert Syst Appl 82:357–382
19. (PROMISE) http://promise.site.uottawa.ca/SERepository. Accessed 23 Aug 2018
20. Rathore SS, Kumar S (2019) A study on software fault prediction techniques. Artif Intell Rev 51(2):255–327. https://doi.org/10.1007/s10462-017-9563-5
21. Sayyad S, Menzies T (2005) The PROMISE repository of software engineering databases. Canada: University of Ottawa, http://promise.site.uottawa.ca/SERepository.
22. Siers MJ, Islam MZ (2015) Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. Inf Syst 51:62–71
23. Son LH, Pritam N, Khari M, Kumar R, Phuong PTM, Thong PH (2019) Empirical study of software defect prediction: a systematic mapping. Symmetry. MDPI AG. https://doi.org/10.3390/sym11020212
24. Tong H, Liu B, Wang S (2018) Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning. Inf Softw Technol 96:94–111. https://doi.org/10.1016/j.infsof.2017.11.008

25. Wang S, Yao X (2013) Using class imbalance learning for software defect prediction. IEEE Trans Reliab 62(2):434–443
26. Wang T, Zhang Z, Jing X, Zhang L (2015) Multiple kernel ensemble learning for software defect prediction. Autom Softw Eng 23:569–590
27. Xia X, Lo D, Shihab E, Wang X, Yang X (2015) ELBlocker: Predicting blocking bugs with ensemble imbalance learning. Inf Softw Technol 61:93–106
28. Yang X, Lo D, Xia X, Sun J (2017) TLEL: a two-layer ensemble learning approach for just-in-time defect prediction. Inf Softw Technol 87:206–20