SMOTE-Based Homogeneous Ensemble Methods for Software Defect Prediction

Abdullateef O. Balogun[1], Fatimah B. Lafenwa-Balogun[1], Hammed A. Mojeed[1], Victor E. Adeyemo[2], Oluwatobi N. Akande[3(✉)], Abimbola G. Akintola[1], Amos O. Bajeh[1], and Fatimah E. Usman-Hamza[1]

[1] Department of Computer Science, University of Ilorin, Ilorin 1515, Nigeria
{balogun.aol, raji.fb, mojeed.ha, akintola.ag, bajehamos, usman-hamza.fe}@unilorin.edu.ng
[2] School of Built Environment, Engineering and Computing, Leeds Beckett University, Headingley Campus, Leeds LS6 3QS, UK
v.adeyemo5225@student.leedsbeckett.ac.uk
[3] Department of Computer Science, Landmark University, Omu-Aran, Kwara State, Nigeria
akande.noah@lmu.edu.ng

Abstract. Class imbalance is a prevalent problem in machine learning which affects the prediction performance of classification algorithms. Software Defect Prediction (SDP) is no exception to this latent problem. Solutions such as data sampling and ensemble methods have been proposed to address the class imbalance problem in SDP. This study proposes a combination of Synthetic Minority Oversampling Technique (SMOTE) and homogeneous ensemble (Bagging and Boosting) methods for predicting software defects. The proposed approach was implemented using Decision Tree (DT) and Bayesian Network (BN) as base classifiers on defects datasets acquired from NASA software corpus. The experimental results showed that the proposed approach outperformed other experimental methods. High accuracy of 86.8% and area under operating receiver characteristics curve value of 0.93% achieved by the proposed technique affirmed its ability to differentiate between the defective and non-defective labels without bias.

Keywords: Software Defect Prediction · Class imbalance · Data sampling · Ensemble methods

1 Introduction

The rapid and continuous influence of software systems on human activities cannot be over-emphasized. This influence can be attributed to the comfort and pleasure derived from using these software systems [1, 2]. Developing quality and reliable software systems become imperative as the adverse effect of defective software systems may be disastrous. Aside from end-users' dissatisfaction, high over-head cost (human and capital) are some of the implications of defective software systems [3–6]. However, software quality assurance, a conventional process of ensuring quality software

O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 615–631, 2020.
https://doi.org/10.1007/978-3-030-58817-5_45

systems, is not adequate as modern software systems are implicitly large and interdependent as a result of periodic and continuous updates and upgrades [7–10]. Hence, sophisticated approaches such as software defect prediction are needed to complement conventional methods of software testing in building quality software systems.

Software Defect Prediction (SDP) is the deployment of Machine Learning (ML) methods for identifying defective software modules or components. SDP can assist software engineers to judiciously utilize available resources in software testing or maintenance by focusing on defective software modules or components before software release [11–14]. SDP models are built on details from software features such as source code complexity, software development history, software cohesion and coupling to predict defective modules in software systems. These software features are numerically quantified to determine the level of software systems quality and reliability [15–18].

Machine learning techniques are deployed for building SDP models using software features. Both supervised and unsupervised ML techniques have been used in building SDP models [3, 19–21]. The goal is to build an SDP model with high accuracy and precision on predicting defects in software systems. Nonetheless, the prediction performance of SDP models depends on the quality of software metric datasets used for developing the models. That is, software features used for building SDP models influence the prediction performance of SDP models [4, 9, 22, 23]. These software features are convoluted and distorted which can be traced to class imbalance problem. Class imbalance in SDP occurs when there is an unequal representation of class labels with non-defective instances as the majority and defective instances as a minority. It is a latent problem that occurs naturally in the software features and impedes the predictive performances of SDP models [21, 24].

Handling class imbalance has raised concerns and attention from researchers as many studies and methods have been proposed to address the imbalance problem [8, 11, 12, 21, 24–26]. From existing studies, it was observed that SDP models built with imbalanced datasets produces inaccurate results as the ensuing SDP models tend to over-fit. That is, SDP models built on imbalanced datasets recognize the majority class label more than the minority class label [12, 21, 24–26]. It is crucial to note that accurate prediction of the minority class label (defective class) is of utmost importance as a failure to predict a defective class may be detrimental. Consequently, researchers have employed methods such as data sampling, cost-sensitive learning and ensemble methods to address class imbalance problem in SDP [19, 25–28]. These methods had a good impact on SDP models; however, there is still a need for more solutions to address the class imbalance in SDP. Data sampling methods have been known to address class imbalance problem by increasing the minority class label (over-sampling) or decreasing the majority class label (under-sampling) [21, 24, 25]. Also, it has been established that class imbalance has little or no effect of ensemble methods [19, 26, 27]. Instigated by the preceding findings, this study proposes the combination of data sampling and ensemble methods to address class imbalance problem in SDP.

This study proposes a novel framework based on Synthetic Minority Oversampling Technique (SMOTE) and homogeneous (Bagging and Boosting) ensemble methods for SDP. SMOTE was used to balance the datasets while homogenous ensemble methods were used to amplify the prediction performances of SDP models. Bayesian Network (BN) and Decision Tree (DT) algorithms were implemented on the new preprocessed

datasets to develop classifiers and the prediction performances of the proposed techniques were evaluated using accuracy, Area Under the Receiver Operating Characteristics (ROC) Curve (AUC) and f-measure.

In summary, the main contributions of this study are:

i. A novel software defect prediction framework based on homogeneous ensemble and SMOTE methods were presented.
ii. The effect of combining homogeneous ensemble and SMOTE data sampling methods on the prediction performances of SDP models was empirically validated.

The rest of the paper is outlined as follows. Section 2 presents a review of related works on class imbalance and high dimensionality in SDP. Section 3 describes the research approach employed in this study. Experimental results and analyses are discussed in Sect. 4. Section 5 concludes the study.

2 Related Work

Researchers have pointed out that the class imbalance problem negatively affects the prediction performance of SDP models. In most cases, class imbalance makes SDP models over-fits which make these models unreliable. Methods such as data sampling, ensemble methods and cost-sensitive analysis are the primary methods proposed by researchers to address class imbalance problem in

Singh, Misra and Sharma [29] conducted a study on the automation of bug (defect) severity prediction using summary extracted from bug metrics. Ensemble methods (voting and Bagging) were deployed to deal with the latent class imbalance problem from the generated bug dataset. Their results showed that ensemble methods had improved performance over single classifiers. This shows that ensemble methods work well with class imbalance.

El-Shorbagy, El-Gammal and Abdelmoez [30] in their study, combined SMOTE with a heterogeneous ensemble (stacking) method. They aimed to maximize the advantage of addressing the minority class labels by aggregating the performance of selected base classifiers. Their proposed method showed better prediction performance and outperformed other existing methods used on minority class labels. However, it is pertinent to note that stacking ensemble method consumes time in building models and requires several combinations of base classifiers to be effective [31, 32].

Balogun, Basri, Abdulkadir, Adeyemo, Imam and Bajeh [24] empirically validated the prediction performance stability of SDP models using data sampling methods. Both undersampling (Random Under-Sampling: RUS) and oversampling (SMOTE) were studied with varying imbalance ratio (IR). From their experimental results, it was observed that the presence of the class imbalance problem in SDP datasets affects the prediction performance of SDP models. Besides, they recommend the use of SMOTE technique for addressing the class imbalance problem in SDP. The findings from their study correlate positively with that of Yu, Jiang and Zhang [21].

Laradji, Alshayeb and Ghouti [33] investigated the effect of combining feature selection with ensemble methods for SDP. They aimed at addressing class imbalance problem with ensemble method and reduce feature redundancy via feature selection

methods. Their results showed that carefully selected features improve the prediction performances of SDP models. Nonetheless, the effect of using ensemble methods to address class imbalance may not be as effective as using both data sampling and ensemble methods together.

Furthermore, Song, Guo and Shepperd [26] conducted an extensive empirical analysis on the effect of class imbalance on SDP models. Their experimental results showed that class imbalance affects the prediction performance of SDP models. Also, they opined that the right combination of data sampling methods and classifiers can yield good prediction performance. Goel, Sharma, Khatri and Damodaran [11] in their study also supported the claim of using the right data sampling technique for class imbalance problem in SDP.

Malhotra and Jain [34] empirically compared the prediction performances of seven (7) (boosting) ensemble methods. Their results indicated that data sampling technique should be applied before performing the boosting ensemble technique.

Similarly, Wang and Yao [35] carried out a comparative performance analysis of selected class imbalance problem solutions in SDP. They concluded that ensemble methods are superior to other methods such as data sampling and cost-sensitive methods. Findings from the preliminary studies of Rodriguez, Herraiz, Harrison, Dolado and Riquelme [25] also arrived at the same conclusion. Invariably, the combination of these methods may produce results better than any of the individual methods.

Kumar, Misra and Rath [36] used correlation analysis and multivariate linear regression feature selection method to select important source codes metrics for defect prediction. The culled datasets were trained by variants of neural network and ensemble methods. Their experimental results showed the effectiveness of ensemble methods in SDP specifically with feature selection.

Based on the preceding analysis, this study proposes the combination of data sampling (SMOTE) and homogeneous ensemble (Bagging and Boosting) to address class imbalance problem and subsequently improve the prediction performance of SDP models.

3 Methodology

In this section, the classification algorithms, data sampling method (SMOTE), homogeneous ensembles methods (Bagging & Boosting), and software defect datasets.

3.1 Classification Algorithms

Decision Tree (DT) and Bayesian Network (BN) algorithms are used as base-line prediction models in this study. DT and BN algorithms have been widely implemented in numerous SDP studies with good prediction performance. Besides, DT and BN have been reported to be stable with imbalance datasets [21, 24]. Table 1 presents DT and BN algorithms with their parameter setting as used in this study

Table 1. Classification algorithms

Classifier	Parameter setting
Decision Tree (DT)	ConfidenceFactor = 0.25; MinObj = 2
Bayesian Network (BN)	SimpleEstimator = alpha(0.25); SearchMethod = hillClimbing; MaxNoParents = 1

3.2 Synthetic Minority Over-Sampling Technique (SMOTE)

SMOTE is a statistical technique used for generating instances. Ideally, its implementation on a given dataset leads to the generation of synthetic instances belonging to the minority instances of the population without increasing the majority instances. This, therefore, increase the total population of the dataset by reducing the imbalance ratio between the minority and the majority class such that there exists no significant difference between the majority and minority instances. Software defect data are known to suffer from significant class imbalance [8, 21, 24, 35]. As SDP intends to predict defective instances, the developed classification model must be able to significantly discriminate between defective and non-defective instances without bias.

3.3 Homogeneous Ensemble Methods

Bagging Ensemble: Bagging is a homogeneous ensemble method used for amplifying the prediction performance of classification algorithms. The base classifiers of a bagging ensemble learn from a given dataset using different samples extracted from the original dataset. An aggregation of the classifiers' output is then carried out at prediction time [37]. Thus, this aggregation technique ensures that the variance of each classifier is reduced and each classifier bias does not also increase. In simple words, bagging algorithm randomly resamples the original datasets, trains multiple base classifiers using the resampled subsets, and finally makes a prediction by using the resulting classifications of multiple base learners [38]. Bagging ensemble is outlined in Algorithm 1.

Algorithm 1. The Bagging Algorithm

Input: training set S, Inducer I, integer N (number of bootstrap samples).

1. for $i = 1$ to T {
2. S' = bootstrap sample from S (sample with replacement)
3. $C_i = I(S')$
4. }
5. $C^*(x) = \arg\max \sum_{i:C_i(x)=y} 1$ (the most frequently predicted label y)

Output: classifier C^*

Boosting Ensemble: Boosting ensemble method deploys a weak classifier in sequence, to train the re-weighted training data. In the end, it uses a majority vote

mechanism for its final decision by integrating all weak hypotheses created by the weak classifiers into the final hypothesis [39]. Boosting uses weighted averages to transform weak classifiers into stronger classifiers with each model deciding what features the next iteration focuses on.

In this study, the AdaBoost.M1 algorithm [40] outlined in Algorithm 2 was implemented.

Algorithm 2. The AdaBoost.M1 Algorithm

Input: Training set $S = \{x_i, y_i\}, i = 1 \ldots m, y_i \in Y, Y = \{c_1, c_2, \ldots, c_k\}, c_k$ is the class label; The number of Iterations T; Weak classifier I.

Initializing weights distribution of $D_1(i) = 1/m$
for $t = 1$ to T
  Train classifier $I(S, D_t)$, get a weak hypothesis $h_t = X \rightarrow \{c_1, c_2, \ldots, c_k\}$
  Compute the error rate of $h_t, \varepsilon_t \leftarrow \sum_{i=1}^{m} D_t(i)[y_i \neq h_t(x_i)]$
  If $\varepsilon_t > 0.5$ then
    $T \leftarrow t - 1$
    Continue
  End if
  Set $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$
  for $i = 1$ *to* m
    Update weight $D_{t+1}(i) = D_t(i)\beta_t^{1-[y_i \neq h_t(x_i)]}$
  End for i
End for t

Output: the final hypothesis $H(x) = \arg max \left(\sum_{t=1}^{T} \ln\left(\frac{1}{\beta_t}\right)[Y \neq h_t(X)]\right)$

Similarly, Table 2 presents the homogeneous ensemble methods and their respective parameters as used in the experimentation stage of this study.

Table 2. Homogeneous ensemble

Homogeneous ensembles	Parameter setting
Bagging	Classifier = {BN, DT}, bagSizePercent = 100; numIteration = 10; seed = 1; calcOutOfBag = False; batchSize = 100
Boosting	Classifier = {BN, DT}, weightThreshold = 100; numIteration = 10; seed = 1; useResampling = True; batchSize = 100

3.4 Software Defect Datasets

In this study, software defect datasets from NASA repository were used for training and testing the SDP models. Shepperd, Song, Sun and Mair [41] cleaned version of NASA datasets was used in the experimentation. Table 3 presents a description of the selected

datasets with their respective number of features, number of instances and imbalance ratio (IR). The IR is based on the ratio of defective instances to non-defective instances in each defect dataset. The diverse IR values of NASA datasets make it appropriate for this study. Also, NASA datasets have been widely used in existing related SDP studies [21, 24].

Table 3. Description of selected NASA software defect datasets

Datasets	# of Features	# of Modules	Imbalance Ratio (IR)
KC1	22	1162	3
KC3	40	194	4
MC2	40	124	2
PC3	38	1053	7
PC4	38	1270	6

3.5 Performance Evaluation Metrics

Existing studies have reported that the choice and selection of performance evaluation metrics is crucial in SDP [42, 43]. Using only accuracy value may be inaccurate due to the imbalance nature of the datasets used for training and testing the SDP models. Accuracy, F-Measure, and Area under Curve (AUC) were used to evaluate the prediction performances of the ensuing SDP models. These evaluation metrics have been widely used and proven to be reliable in SDP studies [4, 18, 21, 22, 27, 33].

i. Accuracy is the number or percentage of correctly classified instances to the total sum of instances.

$$Accuracy = \frac{TP + TN}{TP + FN + TN} \tag{1}$$

ii. F-Measure is defined as the weighted harmonic mean of the test's precision and recall

$$F - Measure = 2X((PrecisionXRecall)/Precision + Recall) \tag{2}$$

iii. The Area under Curve (AUC) shows the trade-off between TP and FP. It provides an aggregate measure of performance across all possible classification thresholds.

Where $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, TP = Correct Classification, FP = Incorrect Classification, TN = Correct Misclassification, and FN = Incorrect Misclassification.

3.6 Experimental Framework

Figure 1 presents the experimental framework developed in this study. To empirically assess the efficacy of the proposed method on SDP models, the experimental

framework is used on 5 defect datasets (See Table 3) with the base classifiers (DT and BN) (See Table 1) and homogeneous ensemble (Boosting and Bagging) (See Table 2). *K*-fold (where $k = 10$) cross-validation technique is used for the evaluation of the SDP models in this study. Our choice of 10-fold CV is in line with existing studies and its ability to build SDP models with low bias and variance [21, 22, 42, 43].

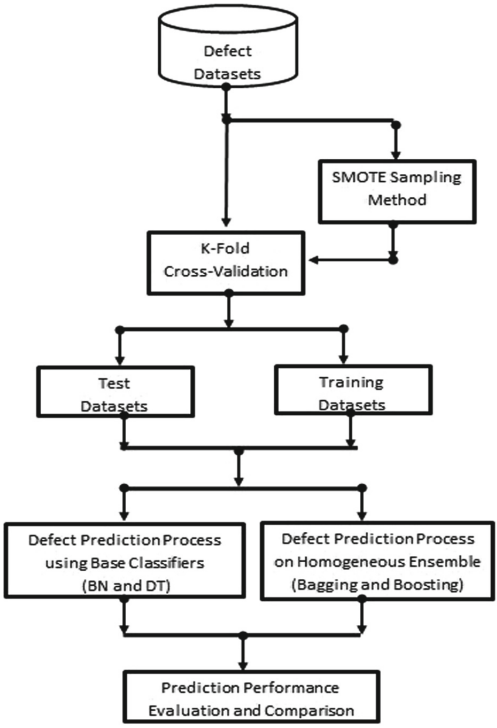

Fig. 1. Experimental framework

From the experimental framework, the majority and minority classes in each dataset are balanced using the SMOTE sampling technique. The balanced representation of the classes was based on 50% defective class and 50% non-defective class as used in existing studies [21, 24]. The essence of this is to ensure the resulting SDP models were trained with each class labels and to give credibility to the ensuing SDP models in predicting the appropriate class labels (defective or non-defective). Our choice of SMOTE as sampling technique is based on its performance and relevance in existing studies [11, 12, 24, 25, 30].

Thereafter, the homogeneous ensembles and the base classifiers are then applied on the original and balanced datasets based on 10-fold cross-validation (CV) technique. CV technique will ensure better usage of the datasets as each instance will be used for training and testing iteratively. Details on how CV works are reported and can be

referenced in [44, 45]. Consequently, the prediction performance of the ensuing models will be evaluated using accuracy, f-measure and AUC.

Also, SDP models based on BN and DT classification algorithm with and without SMOTE and homogeneous ensemble (–i- BN, –ii- BN and SMOTE (BN + SMOTE), iii- Bagged BN, -iv- Boosted BN, -v- DT, -vi- DT and SMOTE (DT + SMOTE), -vii- Bagged DT, -viii- Boosted DT) were developed to create unprejudiced comparison and to measure the effect of SMOTE and ensemble on the prediction performance of the base classifiers. All experiments were carried out using the WEKA machine learning tool [46].

4 Results

The results obtained after evaluating the various developed models are presented and discussed in this section. It is important to showcase the significant impact of SMOTE sampling technique on SDP model development. More so, the efficacy of the ensemble methods over the base-line classifier is another focal point of this study. Thus, the results will be presented to reflect these impacts concerning each base-line classifiers.

Table 4 presents the prediction performances of base classifiers (BN and DT) on the original SDP datasets. The BN classifier, as seen in Table 4, yielded an average accuracy of 71.35%, average AUC of 0.69 and average F-measure of 0.73. Its accuracy scores range from 67.61% to 77.83%. Likewise, AUC scores ranged from 0.584 to 0.81, and F-measure scores from 0.693 to 0.775. The DT classifier also had good prediction performances with an average accuracy of 77.14%, average AUC of 0.65 and an average F-measure of 0.76. DT lowest (60.5%) and highest (86.93%) accuracy scores were from MC2 and PC4 datasets respectively.

Table 4. Prediction performance of BN and DT on the original datasets

Datasets	Original datasets					
	Accuracy (%)		AUC		F-Measure	
	BN	DT	BN	DT	BN	DT
KC1	68.33	74.18	0.681	0.604	0.698	0.717
KC3	77.83	79.4	0.584	0.653	0.775	0.783
MC2	70.16	60.5	0.614	0.589	0.693	0.608
PC3	67.61	84.71	0.779	0.591	0.731	0.839
PC4	72.83	86.93	0.81	0.789	0.767	0.869
Average	71.35	77.14	0.69	0.65	0.73	0.76

From the experimental results in Table 4, it is evident that the base classifiers (BN and DT) have good prediction performance on the original datasets (with class imbalance problem) with their respective average accuracy (BN:71.35%; DT:77.14%) and f-measure (BN:73%; DT:76%) values greater than 70% while there AUC (BN:69%; DT:65%) values are very close to 70%. The relative average prediction

performances of BN and DT in this case although acceptable can be attributed to the occurrence of the latent class imbalance in the SDP datasets. Hence, the removal of the class imbalance problem by balancing the class labels may generate a better prediction performance for the base classifiers (BN and DT) [21, 24].

Table 5. Prediction performance of BN and DT on the balanced datasets

Datasets	Balanced datasets					
	Accuracy (%)		AUC		F-Measure	
	BN + SMOTE	DT + SMOTE	BN + SMOTE	DT + SMOTE	BN + SMOTE	DT + SMOTE
KC1	72.45	79.70	0.800	0.807	0.724	0.797
KC3	75.87	82.20	0.849	0.858	0.756	0.821
MC2	70.44	64.20	0.729	0.657	0.700	0.641
PC3	88.29	88.00	0.781	0.902	0.734	0.880
PC4	86.10	91.31	0.954	0.915	0.861	0.913
Average	78.63	81.08	0.820	0.830	0.760	0.810

Furthermore, Table 5 shows the prediction performances of BN and DT classifiers on the balanced datasets. This is to reveal the effect of data sampling (in this case SMOTE technique) on the prediction performance of SDP models. That is, to empirically validate if the removal of class-imbalance via SMOTE data sampling technique will positively improve the prediction performance of BN and DT classifiers.

It was observed that when BN was applied on the balanced datasets, an average accuracy of 78.63%, average AUC of 0.82 and an average f-measure of 0.76 was recorded. Besides, BN had its highest accuracy value (88.29%) on PC3 and its lowest accuracy value (70.44%) on MC2. On the other hand, when DT was applied on the balanced datasets, an average accuracy of 81.08%, average AUC of 0.83 and an average f-measure score of 0.81 was recorded. Also, DT had its peak (91.31%) and lowest accuracy (64.2%) values on PC4 and MC2 datasets respectively. It can be seen that both classifiers had their lowest accuracy values on MC2. This may be due to the high number of features in MC2 (40 features) (See Table 3).

Comparatively, the prediction performances of BN and DT on balanced datasets (See Table 5) were improved and better than their prediction performance with original datasets (See Table 4). Specifically, BN with balanced datasets (BN + SMOTE) had +10.2% increment in average accuracy, +18.8% increment in average AUC and +4% increment in f-measure values over BN with original datasets. The same trend was observed in DT on balanced datasets (DT +SMOTE). There was a percentage increase of +5.1% in the average accuracy, +27.7% in the average AUC and +6.6% in the average f-measure values of DT + SMOTE over DT. Consequently, the percentage increase observed in the prediction performance of BN + SMOTE and DT + SMOTE indicates that balancing via SMOTE technique has a positive effect on the prediction performances of BN and DT. Thus, our findings revealed that class imbalance impedes the performance of SDP models and can be resolved using data sampling technique (in this case SMOTE) [12, 21, 24, 25, 30, 34].

Tables 6 and 7 present the prediction performance of the homogeneous ensembles (Bagging and Boosting) respectively on the original SDP datasets. From Table 6, BaggedBN on the original datasets had an average accuracy of 71.50%, an average AUC of 0.73 and an average f-measure value of 0.74. Also, Bagged DT recorded an average accuracy of 80.66%, average AUC of 0.78 and average f-measure of 0.79 value across the studied datasets.

As presented in Table 7, BoostedBN had an average accuracy of 79.58%, average AUC of 0.7 and average f-measure 0.77 while BoostedDT had 78.68% average accuracy, 0.76 average AUC and 0.78 average f-measure value. From both tables (Table 6 and Table 7), there are no clear cut superior ensemble methods. That is, the prediction performance of both boosting and bagging ensemble methods depends on the choice of datasets and base classifiers. However, when compared with experimental results of the base classifiers on original datasets (See Table 4), the ensemble methods are superior in prediction performance.

Table 6. Prediction performance of bagging ensemble on original datasets

Datasets	Original datasets					
	Accuracy (%)		AUC		F-Measure	
	Bagged BN	Bagged DT	Bagged BN	Bagged DT	Bagged BN	Bagged DT
KC1	67.73	77.54	0.687	0.719	0.692	0.752
KC3	76.29	84.54	0.688	0.729	0.768	0.827
MC2	71.77	66.13	0.663	0.736	0.711	0.639
PC3	68.00	85.94	0.781	0.789	0.734	0.835
PC4	73.70	89.13	0.810	0.914	0.773	0.886
Average	71.50	80.66	0.730	0.780	0.740	0.790

Table 7. Prediction performance boosting ensemble on original datasets

Datasets	Original datasets					
	Accuracy (%)		AUC		F-Measure	
	Boosted BN	Boosted DT	Boosted BN	Boosted DT	Boosted BN	Boosted DT
KC1	74.01	73.24	0.670	0.691	0.680	0.721
KC3	81.44	79.38	0.590	0.712	0.799	0.777
MC2	70.16	66.94	0.596	0.701	0.693	0.668
PC3	85.66	85.66	0.785	0.788	0.820	0.846
PC4	86.61	88.19	0.868	0.893	0.855	0.879
Average	79.58	78.68	0.700	0.760	0.770	0.780

BaggedDT had a +4.56% increase of average accuracy values when compared with DT on original datasets. On the other hand, BaggedBN had an insignificant increment (+0.02) in its average accuracy value over BN on the original datasets. Also, BoostedBN had +11.4% increments in its average accuracy value over BN on the original

datasets while BoostedDT had approximately +2% increments. The homogeneous (Bagging and Boosting) ensemble methods amplified the respective prediction performances of base classifiers BN and DT. Furthermore, the superiority of the homogeneous ensemble methods over the base classifiers can be attributed to its ability to cope with class imbalance. However, the prediction performances of the base classifiers (BN and DT) on the balanced datasets (BN + SMOTE and DT + SMOTE) were superior to the prediction performances of the homogeneous ensemble methods on original datasets. Hence, this study concludes that ensemble methods can amplify the prediction performances of base classifiers and accommodates class imbalance but ensemble methods are not as effective as data sampling methods in addressing the class imbalance in SDP.

The prediction performances of the homogeneous ensemble (Bagging and Boosting) methods on the balanced SDP datasets are presented in Table 8 and Table 9 respectively. BaggedBN + SMOTE had an average accuracy of 80.09%, an average AUC of 0.86 and an average f-measure value of 0.79. Also, BaggedDT + SMOTE recorded an average accuracy of 85.12%, average AUC of 0.79 and an average f-measure of 0.85 across the studied datasets.

Table 8. Prediction performance of bagging ensemble on balanced datasets

Datasets	Balanced datasets					
	Accuracy (%)		AUC		F-Measure	
	BaggedBN + SMOTE	BaggedDT + SMOTE	BaggedBN + SMOTE	BaggedDT + SMOTE	BaggedBN + SMOTE	BaggedDT + SMOTE
KC1	73.95	82.65	0.807	0.896	0.739	0.826
KC3	78.73	85.40	0.855	0.898	0.787	0.854
MC2	71.77	74.21	0.716	0.831	0.653	0.742
PC3	88.56	89.86	0.965	0.967	0.886	0.899
PC4	87.43	93.46	0.959	0.979	0.874	0.935
Average	80.09	85.12	0.860	0.910	0.790	0.850

Also, as presented in Table 9, BoostedBN + SMOTE had an average accuracy of 81.94%, average AUC of 0.88 and average f-measure of 0.82 while BoostedDT + SMOTE had had an average accuracy of 86.8%, an average AUC of 0.93 and an average f-measure value of 0.87. The results recorded revealed that BoostedBN + SMOTE and BoostedDT + SMOTE were superior to BaggedBN + SMOTE and BaggedDT + SMOTE. This may be due to boosting ensemble iterative nature of model building against the independent model building of the bagging method [31, 47].

As presented in Table 10, the prediction performances of the proposed approaches (BaggedBN + SMOTE, BaggedDT + SMOTE, BoostedBN + SMOTE and BoostedDT + SMOTE) were superior to the experimented methods (BN, BN + SMOTE, DT, DT + SMOTE, BaggedBN, BaggedDT, BoostedDT, BoostedBN). Specifically, BaggedBN + SMOTE recorded a significant positive increment of 12%, 17.8%, and 6.7% in average accuracy, average AUC and average f-measure values respectively

while BaggedDT + SMOTE also recorded a significant positive increment of 5.5%, 16.67%, and 7.6% in average accuracy, average AUC and average f-measure values respectively over the prediction performance of BaggedBN and BaggedDT. BoostedBN + SMOTE achieved an increment of 2.96%, 25.7%, 6.49% in average accuracy, average AUC and average f-measure compared to an increment of 10.32%, 22.37% and 11.54% recorded by BoostedDT + SMOTE. From these analyses, it could be concluded that the prediction performances of homogeneous ensemble methods can also be amplified using the appropriate data sampling technique (in this case SMOTE technique).

Table 9. Prediction performance boosting ensemble on balanced datasets

Datasets	Balanced datasets					
	Accuracy (%)		AUC		F-Measure	
	Boosted BN + SMOTE	Boosted DT + SMOTE	Boosted BN + SMOTE	Boosted DT + SMOTE	Boosted BN + SMOTE	Boosted DT + SMOTE
KC1	74.41	80.46	0.816	0.880	0.744	0.805
KC3	82.22	86.35	0.904	0.926	0.822	0.863
MC2	70.44	80.50	0.732	0.877	0.700	0.805
PC3	91.38	91.65	0.965	0.970	0.914	0.917
PC4	91.27	95.02	0.976	0.985	0.913	0.950
Average	81.94	86.80	0.880	0.930	0.820	0.870

Table 10. Prediction performance comparison of the implemented SDP models

Prediction models	Average accuracy (%)	Average AUC	Average F-measure
BN	71.35	0.69	0.73
DT	77.14	0.65	0.76
BN + SMOTE	78.63	0.82	0.76
DT + SMOTE	81.08	0.83	0.81
BaggedBN	71.50	0.73	0.74
BoostedBN	79.58	0.70	0.77
BaggedDT	80.66	0.78	0.79
BoostedDT	78.68	0.76	0.78
*BaggedBN + SMOTE	80.09	0.86	0.79
*BoostedBN + SMOTE	81.94	0.88	0.82
*BaggedDT + SMOTE	85.12	0.91	0.85
*BoostedDT + SMOTE	86.80	0.93	0.87

(indicates proposed methods)*

5 Conclusions

This study has exhaustively discussed an SDP approach based on the combination of a homogeneous ensemble (Bagging and Boosting) and data sampling (SMOTE). The effects of data sampling and homogeneous ensemble methods are empirically validated. The experimental results showed that the SMOTE technique can improve the prediction performance of not only the base classifier (BN and DT) but the homogeneous ensemble methods inclusive. Also, the proposed approaches (BaggedBN + SMOTE, BaggedDT + SMOTE, BoostedBN + SMOTE and BoostedDT + SMOTE) significantly outperformed the base classifiers (BN and DT). This indicates that the combination of SMOTE and homogeneous ensemble does not only address the class imbalance problem but also positively increase the prediction performance of the base classifiers. Future work will attempt to optimize ensemble parameters and degree of data sampling on SDP approaches.

References

1. Basri, S., Almomani, M.A., Imam, A.A., Thangiah, M., Gilal, A.R., Balogun, A.O.: The organisational factors of software process improvement in small software industry: comparative study. In: Saeed, F., Mohammed, F., Gazem, N. (eds.) IRICT 2019. AISC, vol. 1073, pp. 1132–1143. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-33582-3_106
2. Mojeed, H.A., Bajeh, A.O., Balogun, A.O., Adeleke, H.O.: Memetic approach for multi-objective overtime planning in software engineering projects. J. Eng. Sci. Technol. **14**, 3213–3233 (2019)
3. Balogun, A., Oladele, R., Mojeed, H., Amin-Balogun, B., Adeyemo, V.E., Aro, T.O.: Performance analysis of selected clustering techniques for software defects prediction. Afr. J. Comput. ICT **12**, 30–42 (2019)
4. Balogun, A.O., Basri, S., Abdulkadir, S.J., Hashim, A.S.: Performance analysis of feature selection methods in software defect prediction: a search method approach. Appl. Sci. **9**, 2764 (2019)
5. Bajeh, A.O., Oluwatosin, O.-J., Basri, S., Akintola, A.G., Balogun, A.O.: Object-oriented measures as testability indicators: an empirical study. J. Eng. Sci. Technol. **15**, 1092–1108 (2020)
6. Gupta, A., Suri, B., Kumar, V., Misra, S., Blažauskas, T., Damaševičius, R.: Software code smell prediction model using Shannon, Rényi and Tsallis entropies. Entropy **20**, 372 (2018)
7. Bashir, K., Li, T., Yohannese, C.W., Mahama, Y.: Enhancing software defect prediction using a supervised-learning based framework. In: 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), pp. 1–6. IEEE (2017)
8. Chen, L., Fang, B., Shang, Z., Tang, Y.: Tackling class overlap and imbalance problems in software defect prediction. Softw. Qual. J. **26**(1), 97–125 (2016). https://doi.org/10.1007/s11219-016-9342-6
9. Ghotra, B., McIntosh, S., Hassan, A.E.: A large-scale study of the impact of feature selection techniques on defect classification models. In: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), pp. 146–157. IEEE (2017)

10. Chaturvedi, K., Bedi, P., Misra, S., Singh, V.: An empirical validation of the complexity of code changes and bugs in predicting the release time of open-source software. In: 2013 IEEE 16th International Conference on Computational Science and Engineering, pp. 1201–1206. IEEE (2013)
11. Goel, L., Sharma, M., Khatri, S.K., Damodaran, D.: Implementation of data sampling in class imbalance learning for cross project defect prediction: an empirical study. In: 2018 Fifth International Symposium on Innovation in Information and Communication Technology (ISIICT), pp. 1–6. IEEE (2018)
12. Hamdy, A., El-Laithy, A.: SMOTE and Feature Selection for More Effective Bug Severity Prediction. Int. J. Softw. Eng. Knowl. Eng. **29**, 897–919 (2019)
13. Iqbal, A., Aftab, S.: A classification framework for software defect prediction using multi-filter feature selection technique and MLP. Int. J. Mod. Educ. Comput. Sci. **12**(1), 18–25 (2020). https://doi.org/10.5815/ijmecs.2020.01.03
14. Oluwagbemiga, B.A., Shuib, B., Abdulkadir, S.J., Sobri, A.: A hybrid multi-filter wrapper feature selection method for software defect predictors. Int. J Supply Chain Manag. **8**, 9–16 (2019)
15. Kamei, Y., Shihab, E.: Defect prediction: accomplishments and future challenges. In: IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), vol. 5, pp. 33–45. IEEE (2016)
16. Kondo, M., Bezemer, C.-P., Kamei, Y., Hassan, A.E., Mizuno, O.: The impact of feature reduction techniques on defect prediction models. Empir. Softw. Eng. **24**(4), 1925–1963 (2019). https://doi.org/10.1007/s10664-018-9679-5
17. Li, Z., Jing, X.-Y., Zhu, X.: Progress on approaches to software defect prediction. IET Softw. **12**, 161–175 (2018)
18. Mabayoje, M.A., Balogun, A.O., Jibril, H.A., Atoyebi, J.O., Mojeed, H.A., Adeyemo, V.E.: Parameter tuning in KNN for software defect prediction: an empirical analysis. Jurnal Teknologi dan Sistem Komputer **7**, 121–126 (2019)
19. Tong, H., Liu, B., Wang, S.: Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning. Inf. Softw. Technol. **96**, 94–111 (2018)
20. Usman-Hamza, F.E., Atte, A.F., Balogun, A.O., Mojeed, H.A., Bajeh, A.O., Adeyemo, V. E.: Impact of feature selection on classification via clustering techniques in software defect prediction. J. Comput. Sci. Appl. **26**(1), 73–88 (2019). https://doi.org/10.4314/jcsia.v26i1.8
21. Yu, Q., Jiang, S., Zhang, Y.: The performance stability of defect prediction models with class imbalance: An empirical study. IEICE Trans. Inf. Syst. **100**, 265–272 (2017)
22. Xu, Z., Liu, J., Yang, Z., An, G., Jia, X.: The impact of feature selection on defect prediction performance: an empirical comparison. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 309–320. IEEE (2016)
23. Gupta, A., Suri, B., Misra, S.: A systematic literature review: code bad smells in java source code. In: Gervasi, O., et al. (eds.) ICCSA 2017. LNCS, vol. 10408, pp. 665–682. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62404-4_49
24. Balogun, A.O., Basri, S., Abdulkadir, S.J., Adeyemo, V.E., Imam, A.A., Bajeh, A.O.: Software defect prediction: analysis of class imbalance and performance stability. J. Eng. Sci. Technol. **14**, 3294–3308 (2019)
25. Rodriguez, D., Herraiz, I., Harrison, R., Dolado, J., Riquelme, J.C.: Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–10 (2014)

26. Song, Q., Guo, Y., Shepperd, M.: A comprehensive investigation of the role of imbalanced learning for software defect prediction. IEEE Trans. Softw. Eng. **45**, 1253–1269 (2018)
27. Yang, X., Lo, D., Xia, X., Sun, J.: TLEL: a two-layer ensemble learning approach for just-in-time defect prediction. Inf. Softw. Technol. **87**, 206–220 (2017)
28. Yohannese, C.W., Li, T.: A combined-learning based framework for improved software fault prediction. Int. J. Comput. Intell. Syst. **10**, 647–662 (2017)
29. Singh, V., Misra, S., Sharma, M.: Bug severity assessment in cross-project context and identifying training candidates. J. Inf. Knowl. Manag. **16**, 1750005 (2017)
30. El-Shorbagy, S.A., El-Gammal, W.M., Abdelmoez, W.M.: Using SMOTE and heterogeneous stacking in ensemble learning for software defect prediction. In: Proceedings of the 7th International Conference on Software and Information Engineering, pp. 44–47 (2018)
31. Zhou, Z.-H.: Ensemble Methods: Foundations and Algorithms. CRC Press, Boca Raton (2012)
32. Ardabili, S., Mosavi, A., Várkonyi-Kóczy, A.R.: Advances in machine learning modeling reviewing hybrid and ensemble methods. In: Várkonyi-Kóczy, A.R. (ed.) INTER-ACADEMIA 2019. LNNS, vol. 101, pp. 215–227. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-36841-8_21
33. Laradji, I.H., Alshayeb, M., Ghouti, L.: Software defect prediction using ensemble learning on selected features. Inf. Softw. Technol. **58**, 388–402 (2015)
34. Malhotra, R., Jain, J.: Handling imbalanced data using ensemble learning in software defect prediction. In: 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 300–304. IEEE (2020)
35. Wang, S., Yao, X.: Using class imbalance learning for software defect prediction. IEEE Trans. Reliab. **62**, 434–443 (2013)
36. Kumar, L., Misra, S., Rath, S.K.: An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes. Comput. Stand. Interfaces **53**, 1–32 (2017)
37. Collell, G., Prelec, D., Patil, K.R.: A simple plug-in bagging ensemble based on threshold moving for classifying binary and multiclass imbalanced data. Neurocomputing **275**, 330340 (2018)
38. Lee, S.-J., Xu, Z., Li, T., Yang, Y.: A novel bagging C4. 5 algorithm based on wrapper feature selection for supporting wise clinical decision making. J. Biomed. Inform. **78**, 144–155 (2018)
39. Sun, B., Chen, S., Wang, J., Chen, H.: A robust multi-class AdaBoost algorithm for mislabeled noisy data. Knowl.-Based Syst. **102**, 87–102 (2016)
40. Yijing, L., Haixiang, G., Xiao, L., Yanan, L., Jinling, L.: Adapted ensemble classification algorithm based on multiple classifier systems and feature selection for classifying multiclass imbalanced data. Knowl.-Based Syst. **94**, 88–104 (2016)
41. Shepperd, M., Song, Q., Sun, Z., Mair, C.: Data quality: some comments on the NASA software defect datasets. IEEE Trans. Softw. Eng. **39**, 1208–1215 (2013)
42. Balogun, A.O., Bajeh, A.O., Orie, V.A., Yusuf-Asaju, W.A.: Software defect prediction using ensemble learning: an ANP based evaluation method. FUOYE J. Eng. Technol. **3**, 50–55 (2018)
43. Jimoh, R., Balogun, A., Bajeh, A., Ajayi, S.: A PROMETHEE based evaluation of software defect predictors. J. Comput. Sci. Appl. **25**, 106–119 (2018)

44. Yadav, S., Shukla, S.: Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In: 2016 IEEE 6th International Conference on Advanced Computing (IACC), pp. 78–83. IEEE (2016)
45. Arlot, S., Lerasle, M.: Choice of V for V-fold cross-validation in least-squares density estimation. J. Mach. Learn. Res. **17**, 7256–7305 (2016)
46. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM Sig. Exp. **11**, 10–18 (2009)
47. Singhal, Y., Jain, A., Batra, S., Varshney, Y., Rathi, M.: Review of bagging and boosting classification performance on unbalanced binary classification. In: 2018 IEEE 8th International Advance Computing Conference (IACC), pp. 338–343. IEEE (2018)