ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/362262887

Empirical Analysis of Data Sampling-Based Ensemble Methods in Software Defect Prediction

Chapter · July 2022

DOI: 10.1007/978-3-031-10548-7_27

CITATIONS
2

READS
155

8 authors, including:



Abdullateef Oluwagbemiga Balogun
Universiti Tecknologi PETRONAS
99 PUBLICATIONS **970** CITATIONS
SEE PROFILE



Amos Orenyi Bajeh
University of Ilorin
48 PUBLICATIONS **360** CITATIONS
SEE PROFILE



Hammid Olanrewaju Adeleke
University of Ilorin
3 PUBLICATIONS **12** CITATIONS
SEE PROFILE



Modinat Abolore Mabayoje
University of Ilorin
16 PUBLICATIONS **71** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

SBSE: Single and Multi-Objective Next Release Problem View project

Memetic approach for solving Overtime Planning Problem (OPP) in software projects View project

All content following this page was uploaded by Abdullateef Oluwagbemiga Balogun on 01 August 2022.

The user has requested enhancement of the downloaded file.



Empirical Analysis of Data Sampling-Based Ensemble Methods in Software Defect Prediction

Abdullateef O. Balogun[1,2(✉)], Babajide J. Odejide[1], Amos O. Bajeh[1], Zubair O. Alanamu[3], Fatima E. Usman-Hamza[1], Hammid O. Adeleke[1], Modinat A. Mabayoje[1], and Shakirat R. Yusuff[4]

[1] Department of Computer Science, University of Ilorin, Ilorin PMB 1515, Nigeria
{balogun.ao1,bajehamos,usman-hamzah.fe,
mabayoje.ma}@unilorin.edu.ng
[2] Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 32610 Perak, Malaysia
abdullateef_16005851@utp.edu.my
[3] Computer Services and Information Technology (COMSIT), University of Ilorin, Ilorin PMB 1515, Nigeria
alanamu.zo@unilorin.edu.ng
[4] Department of Computer Science, Kwara State University, Malesse, Kwara State, Nigeria

Abstract. This research work investigates the deployment of data sampling and ensemble techniques in alleviating the class imbalance problem in software defect prediction (SDP). Specifically, the effect of data sampling techniques on the performance of ensemble methods is investigated. The experiments were conducted using software defect datasets from the NASA software archives. Five data sampling methods (over-sampling techniques (SMOTE, ADASYN, and ROS), and undersampling techniques (RUS and NearMiss) were combined with bagging and boosting ensemble methods based on Naïve Bayes (NB) and Decision Tree (DT) classifier. Predictive performances of developed models were assessed based on the area under the curve (AUC), and Matthew's correlation coefficient (MCC) values. From the experimental findings, it was observed that the implementation of data sampling methods further enhanced the predictive performances of the experimented ensemble methods. Specifically, BoostedDT on the ROS-balanced datasets recorded the highest average AUC (0.995), and MCC (0.918) values respectively. Aside NearMiss method, which worked best with the Bagging ensemble method, other studied data sampling methods worked well with the Boosting ensemble technique. Also, some of the developed models particularly BoostedDT showed better prediction performance over existing SDP models. As a result, combining data sampling techniques with ensemble methods may not only improve SDP model prediction performance but also provide a plausible solution to the latent class imbalance issue in SDP processes.

Keywords: Data sampling · Ensemble methods · Class imbalance · Software defect prediction

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13381, pp. 363–379, 2022.
https://doi.org/10.1007/978-3-031-10548-7_27

1 Introduction

The idea behind software defect prediction (SDP) is to deploy machine learning (ML) methods to predict software defects based on historical information such as bug reports and source code edit logs generated from the software development process [1]. SDP can help development teams use available resources more wisely in the software development process by concentrating on flawed or defect-prone modules or components before software product release [1, 2]. To anticipate defective modules in software systems, Data from software features which include source code complexity (Line of Code (LOC), McCabe, and Halstead), software development history, software cohesion, and coupling are used to build SDP models [3–5]. These software features are quantitatively measured to assess the degree of reliability and dependability of software systems [3].

SDP models can be developed using either supervised or unsupervised ML approaches [6–9]. The objective is to develop an SDP model that predicts defects in software systems with perfect certainty and accuracy. Nonetheless, the effectiveness of SDP models is reliant on the characteristics of the software datasets deployed. Particularly, the software attributes utilized to develop SDP models impact their efficacy [1, 2, 10]. By default, software attributes are muddled and skewed, which can be regarded as a class imbalance problem.

A class imbalance occurs in SDP if there is a disproportionality of class labels, having the non-defective and defective cases as majority and minority labels respectively. In addition, the class imbalance is a dormant issue that happens spontaneously in software attributes and impairs the prediction performance of deployed prediction algorithms. Addressing class imbalance as a data quality problem has piqued the interest of experts, as several kinds of research and techniques have been presented to handle the imbalance issue [7, 11, 12]. Based on previous research, SDP models developed using imbalanced datasets provide unreliable findings because the resulting SDP models produce poor performance. In other words, SDP models developed on imbalanced datasets preferentially identify the majority instances over the minority. However, it is crucial to re-affirm that correctly predicting the minority instances (in this case defective labels) is imperative, since neglecting the defective labels may be deleterious. Consequently, in the context of SDP, several researchers have adopted methodologies such as data sampling, cost-sensitive learning, and ensemble methods to address the problem of class imbalance [7, 11, 13]. Independently, these strategies have a positive influence on deployed prediction algorithms in SDP processes; nonetheless, the development of novel approaches to addressing the class imbalance in SDP is still ongoing research. The data sampling approach that increases and decreases the proportion of minority instances (over-sampling) and majority instances (under-sampling) respectively has been reported to overcome the class imbalance problem [14, 15]. Furthermore, the class imbalance has been shown to have negligible or no effect on ensemble techniques [7, 16]. In response to the foregoing reports, this research work proposes a hybrid of data sampling and ensemble approaches to overcome the class imbalance problem in SDP.

Data sampling approaches, especially data oversampling, have been proven in studies to have a positive influence on ML algorithms [17, 18]. However, as demonstrated in [19], they still suffer from excessive volatility and instability. To alleviate the large variance and instability of data sampling methods, ensemble methods such as boosting and bagging

may be used [20]. The objective of this study is to conduct an empirical evaluation of the prediction performance data sampling-based ensemble methods. Specifically, models based on ensembled (Bagging and Boosting) Naïve Bayes (NB) and Decision Tree (DT) classifiers are deployed on newly generated datasets based on Random Over-Sampling (ROS), Synthetic Minority Over-Sampling Technique (SMOTE), Adaptive Synthetic (ADASYN), Random Under-Sampling (RUS), and Near Miss data sampling methods. Software defect datasets from the NASA repository are deployed in this study.

Summarily, the contribution of this study is as follows:

i. Validate the effectiveness of ensemble methods over the conventional ML classifiers on data-sampled datasets in SDP.
ii. Compare the effectiveness of experimented data sampling methods on studied ensemble (bagging and boosting) methods.

The remainder of this research is structured as follows: Sect. 2 outlines the associated works that have been completed in this respect. Section 3 discusses the notion of imbalanced learning, and Sect. 4 presents the experimental data and analyses. Section 5 concludes this research work.

2 Related Works

This section investigates and analyses relevant developed SDP models based on ML methods and current solutions to the class imbalanced problem.

SDP is a method for detecting defects in software systems early on. It determines the characteristics of single code components to detect whether parts of it are prone to defects [7] or to forecast the number of defects in each component or module [2]. Experts have used a variety of ways to create SDP models based on static code metrics. NB [21], DT [14], artificial neural networks (ANNs) [22], support vector machines (SVM) [23], k-nearest neighbour (KNN) [24], and logistic regression (LR) [21] are just a few of the classical classifiers that have been used directly to develop SDP models. Nonetheless, these classifiers ignore the skewness and other inherent data quality problems in software defect datasets that could affect the effectiveness of SDP models [10]. For instance, SVM and KNN tend to overlook the minority class labels as they seek to maximise the accuracy values [25]. [26] reported that software defect datasets are very susceptible to the class imbalance problem, which is an example of the data quality problem.

Class imbalance is a latent anomaly of software defect data that consists of a small number of defective instances and a big number of non-defective instances [27]. NASA dataset PC1 exemplifies this, with just 6.59% of cases belonging to the defective class label. Since most classifiers aim to build classifiers that maximize overall prediction accuracy, this feature has a significant influence on both model training and predictive performance. Therefore, such models often neglect the valued minority (defective) class.

A considerable amount of study has been recommended to address the issue of class imbalance. [28] presented an overview of approaches for decreasing the detrimental impact of imbalance on prediction performance. [29] investigated whether various classifiers detect the same problems. To do this, they used NASA datasets to conduct a

sensitivity analysis and evaluate the outcomes of RF, NB, RPart, and SVM. They concluded that certain flaws are more consistent in defect prediction than others and that each classifier identifies a distinct set of defects.

[30] addressed two procedures, undersampling and oversampling and then asserted that both data sampling approaches were successful. Furthermore, it was discovered that the adoption of advanced sampling procedures did not give any discernible improvement in addressing the class imbalance issue. Similarly, [31] observed that the RUS approach typically outperforms more complicated undersampling algorithms. In addition to undersampling procedures, oversampling methods are prominently deployed to solve the class imbalance issue. Aside from ROS with replacement, numerous more sophisticated data sampling techniques have been created. [15] proposed a unique oversampling approach termed the SMOTE, in which new minority samples are generated based on feature space similarities between existing minority cases. [32] used the borderline-SMOTE to oversample minority class samples close to the borderline. Both [18] and [13] found that oversampling outperforms undersampling. It can be noted that the aforementioned research on data sampling techniques is not conducted on software defect issues, and there is limited literature on evaluating data sampling methods for SDP. In [33], Tomek-Link, an undersampling technique was combined with Random Undersampling (RUS) and Synthetic Minority Oversampling SMOTE). It was reported that this combination showed an improvement in performance than other experimented methods.

Cost-Sensitive Learning is another technique that has been investigated by researchers. [34] evaluated data sampling techniques and MetaCost learning and there concluded that the sampling methods with replacement are effective for imbalanced learning. However, this method is still vague and needs to be more thoroughly investigated because assigning a cost penalty is not generic, it rather is dependent on some factors such as the dataset used, and the level of misclassification [35].

More recently, ensemble methods have been explored by researchers [7, 11, 16]. [36] suggested an ensemble technique for SDP based on object-oriented (OO) modules and compared the proposed method to some existing ML methods. They reported that the proposed method performed better than the studied ML methods. Furthermore, ensemble techniques have been also combined with resampling methods, which is known as the hybrid approach. [11] proposed a hybrid-SMOTE ensemble technique in what they simply referred to as <SMOTE + classifier>. In their experiment, they first resampled the dataset using SMOTE, and then the process of the ensemble was done using RF, AdaBoost, and Bagging. They observed that the suggested approach could effectively enhance the prediction accuracy of studied SDP models. [19] also researched this area, although their work was not in the domain of SDP, they demonstrated the effectiveness of ensembling resampled data over a single base classifier. They conducted their experiments using both simulated and real-world datasets.

3 Methodology

This section outlines and describes the data sampling techniques, prediction algorithms, ensemble methods, defect datasets examined, performance measures, and experimental strategy employed in this research work.

3.1 Data Sampling Method

In this study, five (5) data sampling approaches (SMOTE, ADASYN, ROS, RUS, and NearMiss) are studied. Data sampling techniques are broadly classified into two types: oversampling methods and undersampling methods [15]. The oversampling method's fundamental idea is to balance the dataset by raising or increasing the amount or frequency of minority class instances to an equal number of classes as majority class instances. In contrast, in undersampling approaches, the majority class instances are downsampled or lowered to the same number or frequency as the minority class occurrences. ROS, SMOTE, and ADASYN are instances of oversampling methods, while RUS and NearMiss are examples of undersampling methods. The samples to replicate in ROS are selected at random. This duplication of minority class instances often leads to overfitting and a poor prediction model. In SMOTE, however, the samples are selected using a (K-Nearest Neighbour) k-NN. The Euclidean distance between a feature vector and its nearest neighbour is used to generate a new vector [15]. ADASYN is an oversampling strategy based on k-NN that produces data adaptively based on density distribution [18]. RUS is an undersampling approach that removes examples of the majority class at random until both sets of instances are equal. Although some information may be lost in this process, it increases computation speed and power. NearMiss is another undersampling technique, but instead of randomly selecting samples to eliminate, it uses the k-NN approach [13].

3.2 Prediction Algorithms

In this study, NB and DT algorithms are used as prediction algorithms. NB is a probability-based classifier that is predicated on the Bayes theorem and the presumption that every pair of features is independent of one another [37]. DT is a type of non-parametric classifier whose goal is to create a model that predicts the value of a pre-determined instance using simple decision rules derived from data variables. The classifiers were chosen to bring heterogeneity into the prediction models and are based on their relative use and performance in previous SDP research. Table 1 gives a summary of the chosen models and their parameter values as employed in this research work.

Table 1. Selected prediction algorithms with parameter settings

Prediction algorithms	Parameter settings
DT	ConfidenceFactor = 0.25; MinObj = 2
NB	NumDecimalPlaces = 2; UseKernelEstimator = True

3.3 Ensemble Methods

This study investigated boosting and bootstrap aggregating (Bagging) ensemble methods. Ensemble methods generally combine multiple weak classifiers into a single strong and robust model to improve the effectiveness and stability of the model [26]. To learn

the re-weighted training data, the boosting ensemble method utilizes a weak classifier in sequence. Finally, it uses a majority vote mechanism for its final judgement, including all weak hypotheses created by the weak classifiers into the final hypothesis [14]. In other words, boosting use weighted averages to transform weak classifiers into stronger classifiers, with each model choosing which qualities the next iteration focuses on. In the case of the Bagging ensemble, a bagging ensemble's baseline classifiers learn from a given dataset by extracting multiple samples from the original dataset. The classifiers' output is collected at prediction time. Consequently, the aggregation technique ensures that each classifier's variance is minimized while its bias is not raised. In layman's words, the bagging approach randomly resamples the original datasets, trains several base classifiers using the resampled subsets, and then creates a prediction based on the predictions of the many base learners [36]. Summarily, Table 2 depicts the investigated ensemble techniques and their parameters as they were used throughout the experimental phase of this research work.

Table 2. Experimented Ensemble Methods with parameter settings

Ensemble methods	Parameter settings
Bagging	Classifier = {NB, DT}, bagSizePercent = 100; numIteration = 10; seed = 1; calcOutOfBag = False; batchSize = 100
Boosting	Classifier = {NB, DT}, weightThreshold = 100; numIteration = 10; seed = 1; useResampling = True; batchSize = 100

3.4 Software Defect Datasets

The software defect datasets utilized in this research work were gathered from the NASA repository. In this research work, the [4] version of the NASA corpus was employed. The NASA datasets contain software features obtained from static code analysis centred on source code size and complexity [38–40]. Table 3 contains details of the datasets analysed as well as their corresponding imbalance ratios.

Table 3. Description of studied defect datasets

Datasets	Number of instances	Number of defective instances	Number of defective instances	Imbalance ratio (IR)
KC3	194	36	158	4
PC1	679	55	624	11
MW1	250	25	225	9

3.5 Evaluation Measures

According to available research, choosing performance evaluation criteria is crucial in SDP [41, 42]. This is because the datasets used to train and test the SDP models are unbalanced. Relying just on prediction accuracy values may not be sufficient. For example, in [1], an experiment was conducted to test the biasness of measurements such as Accuracy, F-Measure, and Area Under a ROC Curve (AUC). They concluded that Matthews Correlation Coefficient (MCC) is a more trustworthy statistic since it includes all confusion measures, as opposed to others that exclude the True Negative (TN). In this study, the prediction performances of the developed SDP models were evaluated using AUC, and MCC values. These chosen assessment measures have been used often and are reliable [1, 16, 21].

$$\text{AUC} = \frac{1 + TPR - FPR}{2} \tag{1}$$

$$\text{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \tag{2}$$

3.6 Experimental Procedure

This section discusses the experimental procedure employed in this research work as shown in Fig. 1.

The procedure is designed to experimentally investigate and evaluate the efficacy of the data sampling-based ensemble methods in SDP. Particularly, the original defect datasets and the newly generated datasets based on studied data sampling methods (ROS, SMOTE, ADASYN, RUS, and NearMiss) are deployed on ensembled NB and DT classifiers. That is, each of the studied data sampling methods is used to resolve the inherent class imbalance present in the defect datasets by balancing the number of the majority and minority class variables respectively, hence new datasets. The balancing of the original datasets is based on conclusions presented in previous research [21, 43]. The SDP models are created and tested using the K-fold (k = 10) Cross-Validation (CV) technique. The preference for the k-fold technique is based on its ability to develop prediction models while minimizing the influence of the class imbalance problem [27, 44]. Furthermore, the K-fold technique enables every variable to be deployed iteratively for the training and testing process. The investigated classifiers (NB and DT) and ensemble techniques (Boosting and Bagging) were chosen based on their application and performance in previous research [11, 36]. Table 1 (Sect. 3.3) and Table 2 (Sect. 3.4) indicate the parameter values for the classifiers and ensemble methods investigated in this research work. Following that, the prediction performances of the resulting models are evaluated using the chosen evaluation measures (AUC and MCC values). In addition, the prediction performances of the created model are compared to each other to determine the influence of data sampling techniques on the prediction models (NB and DT) and the effectiveness of the examined data sampling methods. Conclusively, the effectiveness of the created models is correlated with existing SDP models. The essence of the comparison is to validate the effectiveness of data sampling-based ensemble approaches in SDP

procedures. The Python-Scikit ML library was utilized to develop the data sampling techniques, while the WEKA ML platform was used to construct the prediction models. These two ML resources are often employed in SDP and ML activities.

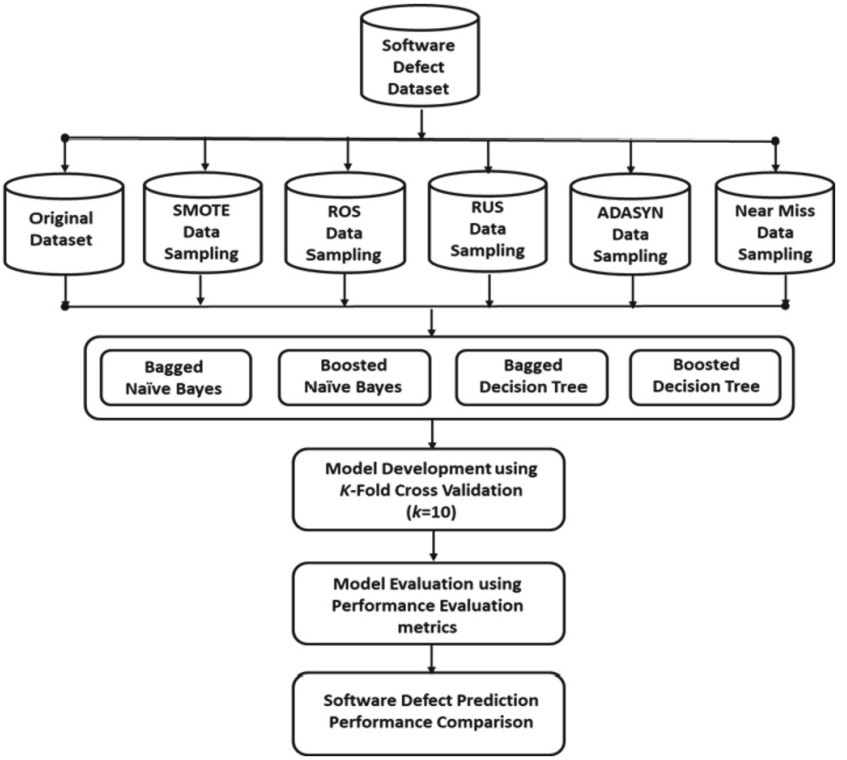


Fig. 1. Experimental framework

4 Results and Discussion

This section displays and analyses the results of evaluating the various constructed SDP models. It is crucial to show how the data sampling approach affects the effectiveness of SDP models. Furthermore, the performance of the studied data sampling-based ensemble models is one of the most important aspects of this study. As a result, the findings for investigated data sampling approaches, ensemble methods, and software defect datasets will be provided to reflect these impacts.

Table 4 and Table 5 display the AUC values of experimented prediction models using original and balanced NASA defect datasets.

From Table 4, NB and DT models developed using the balanced NASA datasets had better AUC values than when original NASA datasets are used. Models based on the studied datasets recorded significant increments in their respective AUC values except in the case of the RUS-balanced KC3 dataset. NB and DT models trained with SMOTE (NB: +9.52%; DT: +24.65%), ADASYN (NB: +8.16%; DT: +29.71%), ROS (NB: +4.23%; DT: +36.75%) and NearMiss (NB: +22.21%; DT: +39.36%)-balanced KC3 datasets had increments in AUC values when compared with the NB and DT models developed with the original KC3 dataset. On PC1 dataset, NB and DT models developed

Table 4. AUC values of NB and DT models on original and balanced datasets

		NB	DT
KC3	SMOTE	0.725	0.814
	ADASYN	0.716	0.847
	ROS	0.690	0.893
	RUS	0.584	0.595
	NearMiss	0.809	0.910
	No Sampling	0.662	0.653
PC1	SMOTE	0.831	0.919
	ADASYN	0.829	0.929
	ROS	0.818	0.964
	RUS	0.826	0.653
	NearMiss	0.858	0.840
	No Sampling	0.790	0.598
MW1	SMOTE	0.803	0.888
	ADASYN	0.812	0.875
	ROS	0.767	0.942
	RUS	0.734	0.588
	NearMiss	0.862	0.803
	No Sampling	0.314	0.503

with SMOTE (NB: +5.19%;DT: +53.68%), ADASYN (NB: +4.94%; DT: + 55.35%), ROS (NB: +3.54%; DT: +61.20%), RUS (NB: +4.56; DT: + 9.20%) and NearMiss (NB: +8.61%; DT: +40.47%)-balanced PC1 datasets had enhanced AUC values when compared with the NB and DT models. Also, a similar occurrence was observed in the MWI dataset. NB and DT models trained with the balanced MW1 dataset had superior AUC values than when the original MW1 dataset is deployed in most cases. NB models developed with SMOTE, ADASYN, ROS, NM, and RUS-balanced MW1 datasets had more than a +100% increment in AUC values while DT models developed with SMOTE (+76.54%), ADASYN (+73.96%), ROS (+87.28%), NearMiss (+59.64%), and RUS (+16.89%) had a significant increment in AUC values. These results proved that the experimented data sampling methods can enhance the performance of NB and DT in the presence of class imbalance.

Based on this observation, the performance of ensembled NB and DT models trained with balanced and original NASA datasets are further analyzed. Table 6 presents the AUC values of Ensemble NB and DT models on original and balanced NASA datasets. Specifically, ensembled NB models developed with SMOTE (BaggedNB: +5.93%; BoostedNB: +27.08%), ADASYN (BaggedNB: +7.42%; BoostedNB: +23.38%), ROS (BaggedNB: +5.64%; BoostedNB: +16.46%) and NearMiss (BaggedNB: +

24.04%; BoostedNB: +16.92%)-balanced KC3 datasets had increments in AUC values when compared with the ensemble NB model developed with the original KC3 dataset. A similar occurrence was observed with ensemble DT models developed with SMOTE (BaggedDT: +16.62%; BoostedDT: +30.06%), ADASYN (BaggedDT: +19.87%; BoostedDT: +33.15%), ROS (BaggedDT: +25.71%; BoostedDT: +39.88%) and NearMiss (BaggedDT: +14.68%; BoostedDT: +22.19%)-balanced KC3 datasets and original KC3 dataset. In the case of the RUS-balanced KC3 dataset, ensembled NB and DT models had poor AUC values that are lower than other experimented models.

Table 5. AUC values of ensemble NB and DT models on original and balanced datasets

		BaggedNB	BaggedDT	BoostedNB	BoostedDT
KC3	SMOTE	0.714	0.898	0.826	0.926
	ADASYN	0.724	0.923	0.802	0.948
	ROS	0.712	0.968	0.757	0.996
	RUS	0.591	0.546	0.545	0.584
	NearMiss	0.836	0.883	0.760	0.870
	No sampling	0.674	0.77	0.650	0.712
PC1	SMOTE	0.826	0.981	0.852	0.985
	ADASYN	0.829	0.988	0.821	0.991
	ROS	0.817	0.998	0.884	0.999
	RUS	0.812	0.785	0.882	0.756
	NearMiss	0.883	0.916	0.855	0.912
	No sampling	0.785	0.834	0.817	0.780
MW1	SMOTE	0.805	0.956	0.863	0.955
	ADASYN	0.813	0.971	0.897	0.980
	ROS	0.767	0.997	0.857	0.999
	RUS	0.722	0.666	0.690	0.533
	NearMiss	0.811	0.863	0.844	0.949
	No sampling	0.772	0.749	0.774	0.715

For the PC1 dataset, ensemble NB and DT models trained with the balanced PC1 dataset had superior AUC values than when the original PC1 dataset is utilized. Ensemble NB models developed with SMOTE (BaggedNB: +5.22%; BoostedNB: +4.28%), ADASYN (BaggedNB: +5.61%; BoostedNB: +0.49%), ROS (BaggedNB: +4.08%; BoostedNB: +8.20%), NM (BaggedNB: +12.48%; BoostedNB: +4.65%) and RUS (BaggedNB: +3.44%; BoostedNB: +7.96%)-balanced PC1 datasets had improved AUC values. Also, the ensemble DT model with a balanced PC1 dataset had better AUC values than when the original PC1 dataset is used.

In addition, on the MWI dataset, ensemble NB and DT models trained with the balanced-MW1 dataset had better AUC values than when the original MW1 dataset is deployed except in the case of the RUS-balanced MW1 dataset. Ensembled NB models developed with SMOTE (BaggedNB: +4.27%; BoostedNB: +11.49%), ADASYN (BaggedNB: +5.31%; BoostedNB: +15.89%), and NearMiss (BaggedNB: + 5.05%; BoostedNB: +9.04%)-balanced MW1 datasets had increments in AUC values when compared with the ensemble NB model developed with the original MW1 dataset. A similar occurrence was observed with ensemble DT models developed with SMOTE (BaggedDT: +29.63%; BoostedDT: +33.56%), ADASYN (BaggedDT: +29.64%; BoostedDT: +37.06%), ROS (BaggedDT: +33.11%; BoostedDT: +39.72%) and NearMiss (BaggedDT: +15.22%; BoostedDT: +32.73%)-balanced MW1 datasets and original KC3 dataset.

Furthermore, as recommended by [1], the performance of the developed models was analyzed using the MCC value. Table 6 and Table 7 present the MCC values of the developed models using both balanced and original NASA datasets. According to Table 6, NB and DT models created utilizing balanced datasets showed higher MCC values than original NASA datasets. NB and DT models based on the balanced datasets showed a considerable increase in their respective MCC values. This observation further supports the notion that data sampling methods can improve the prediction performance of SDP models in the presence of class imbalance.

Table 6. MCC values of NB and DT models on original and balanced datasets

		NB	DT
KC3	SMOTE	0.312	0.626
	ADASYN	0.308	0.694
	ROS	0.317	0.753
	RUS	0.321	0.396
	NearMiss	0.589	0.727
	No sampling	0.278	0.257
PC1	SMOTE	0.396	0.808
	ADASYN	0.386	0.839
	ROS	0.402	0.930
	RUS	0.417	0.273
	NearMiss	0.644	0.600
	No sampling	0.314	0.271

(*continued*)

Table 6. (*continued*)

		NB	DT
MW1	SMOTE	0.479	0.760
	ADASYN	0.469	0.743
	ROS	0.435	0.890
	RUS	0.500	0.280
	NearMiss	0.750	0.600
	No sampling	0.328	0.142

Table 7. MCC values of ensemble NB and DT models on original and balanced datasets

		BaggedNB	BaggedDT	BoostedNB	BoostedDT
KC3	SMOTE	0.328	0.708	0.504	0.728
	ADASYN	0.314	0.701	0.459	0.771
	ROS	0.338	0.817	0.407	0.875
	RUS	0.268	0.000	0.060	0.167
	NearMiss	0.626	0.701	0.589	0.556
	No sampling	0.233	0.137	0.298	0.223
PC1	SMOTE	0.396	0.861	0.541	0.909
	ADASYN	0.384	0.878	0.477	0.905
	ROS	0.406	0.944	0.402	0.972
	RUS	0.400	0.511	0.401	0.456
	NearMiss	0.644	0.711	0.644	0.691
	No sampling	0.287	0.194	0.300	0.315
MW1	SMOTE	0.466	0.810	0.609	0.840
	ADASYN	0.046	0.852	0.650	0.848
	ROS	0.431	0.898	0.536	0.923
	RUS	0.446	0.281	0.360	0.000
	NearMiss	0.718	0.725	0.750	0.600
	No sampling	0.333	0.267	0.279	0.332

Table 7 presents the MCC values of ensemble NB and DT models on both original and balanced datasets. Except in the case of the RUS-balanced dataset, the MCC values of ensemble NB and DT models developed using balanced KC3 datasets showed more than a +40% increase in MCC values. A similar phenomenon was observed with the

ensemble NB and DT models using balanced-PC1 and balanced-MW1 datasets. Specifically, ensemble DT models on balanced-PC1 and balanced-MW1 datasets had a +100% increment in their MCC values in most cases.

Consequently, these findings indicate that the deployment of balanced datasets further enhances the prediction performances of the experimented ensemble NB and DT models. Table 8 shows the performance comparison of some of the developed models (ROS-BoostedDT, ADASYN-BoostedDT and SMOTE-BoostedDT) with existing methods on PC1. Specifically, the experimental results from El-Shorbagy, El-Gammal and Abdelmoez [16], Li, Zhou, Zhang, Liu, Huang and Sun [45], and Alsaeedi and Khan [7] are compared with ROS-BoostedDT, ADASYN-BoostedDT and SMOTE-BoostedDT. The developed methods had superior AUC and MCC values to existing SDP models.

Table 8. Comparison of some developed models with existing SDP results

Methods		AUC	MCC
PC1	Stacking (NB, MLP, J48) [16]	0.876	0.443
	Adaboost[45]	0.861	-
	AdaboostSVM[7]	0.760	-
	BaggedLR[7]	0.770	-
	*ROS-BoostedDT	0.999	0.972
	*ADASYN-BoostedDT	0.991	0.905
	*SMOTE-BoostedDT	0.985	0.909

*Indicates models from this study.

In summary, the analyses of the experimental results show that the investigated data sampling methods can ameliorate the class imbalance problem in SDP datasets while also improving the prediction performances of the SDP models. Furthermore, it was discovered that the analyzed data oversampling techniques (SMOTE, ADASYN, and ROS) outperformed the data undersampling approaches (NearMiss and RUS). There are considerable disparities in performance amongst the explored oversampling strategies since this changes throughout the explored datasets and chosen prediction models. However, it is worth mentioning that the RUS technique had the least influence on the prediction models and, in some instances, performed worse than when the original datasets were used. This finding may be ascribed to the random elimination of key data that might be critical for the SDP process. Although the NearMiss technique is likewise a data undersampling method, it eliminates instances based on their closest neighbour characteristics.

5 Conclusion and Future Works

Addressing SDP concepts and the class imbalance problem as described in research work is critical for successful SDP model development. Data sampling methods are utilized on

software defect datasets in this study to alleviate the latent class imbalance problem by levelling the number of minority and majority class instances observed, resulting in new defect datasets with no class imbalance problem. Particularly, three data oversampling methods (SMOTE, ADASYN, and ROS) and two data undersampling methods (RUS and NM) are deployed on defect datasets from the NASA repository, while ensembled (Bagging and Boosting) NB and DT classifiers are employed on the original and newly developed software defect datasets.

Overall, the experimental findings showed that the data sampling methods investigated can address the class imbalance problem in SDP datasets. Furthermore, in most of the experimental scenarios, the studied data sampling method improved the prediction performances of the deployed ensembled NB and DT models. However, it should be noted that when combining ensemble models with data sampling methods, the choice of the data sampling method, as well as the base classifier, is critical if any significant result is to be achieved. In terms of the effectiveness of the data sampling methods, the oversampling approaches (ROS, SMOTE, ADASYN) had a greater (positive) impact on the prediction models than their undersampling counterparts (RUS and NearMiss).

As a result, it is recommended that data sampling operations, particularly oversampling approaches, be carried out during SDP activities. Implementing data sampling procedures may help to ease the underlying class imbalance issue and ensure the effectiveness of SDP models.

Following this, further study on the hybrid technique should be carried out utilizing other ensemble methods to investigate the data sampling method that best suits them as well as the classification algorithm that works well with those ensemble methods.

References

1. Song, Q., Guo, Y., Shepperd, M.: A comprehensive investigation of the role of imbalanced learning for software defect prediction. IIEEE Trans. Software Eng. **45**, 1253–1269 (2019)
2. Laradji, I.H., Alshayeb, M., Ghouti, L.: Software defect prediction using ensemble learning on selected features. Inf. Softw. Technol. **58**, 388–402 (2015)
3. El-Sharkawy, S., Yamagishi-Eichler, N., Schmid, K.: Metrics for analyzing variability and its implementation in software product lines: a systematic literature review. Inf. Softw. Technol. **106**, 1–30 (2019)
4. Shepperd, M., Song, Q., Sun, Z., Mair, C.: Data quality: some comments on the NASA software defect datasets. IIEEE Trans. Softw. Eng. **39**, 1208–1215 (2013)
5. Tiwari, S., Rathore, S.S.: Coupling and cohesion metrics for object-oriented software: a systematic mapping study. In: Proceedings of the 11th Innovations in Software Engineering Conference, pp. 1–11 (2018)
6. 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. Comp. ICT **12**, 30–42 (2019)
7. Alsaeedi, A., Khan, M.Z.: Software defect prediction using supervised machine learning and ensemble techniques: a comparative study. JSEA **12**, 85–100 (2019)
8. Kumar, L., Dastidar, T.G., Goyal, A., Murthy, L.B., Misra, S., Kocher, V., Padmanabhuni, S.: Predicting software defect severity level using deep-learning approach with various hidden layers. In: Mantoro, T., Lee, M., Ayu, M.A., Wong, K.W., Hidayanto, A.N. (eds.) ICONIP 2021. CCIS, vol. 1517, pp. 744–751. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92310-5_86

9. Kumar, L., et al.: Deep-learning approach with Deepxplore for software defect severity level prediction. In: Gervasi, O., et al. (eds.) ICCSA 2021. LNCS, vol. 12955, pp. 398–410. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87007-2_28
10. Balogun, A., Bajeh, A., Mojeed, H., Akintola, A.: Software defect prediction: a multi-criteria decision-making approach. Niger. J. Technol. Res. **15**, 35–42 (2020)
11. Alsawalqah, H., Faris, H., Aljarah, I., Alnemer, L., Alhindawi, N.: Hybrid SMOTE-ensemble approach for software defect prediction. In: Silhavy, R., Silhavy, P., Prokopova, Z., Senkerik, R., Kominkova Oplatkova, Z. (eds.) CSOC 2017. AISC, vol. 575, pp. 355–366. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57141-6_39
12. 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)
13. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Expl. Newsl. **6**, 20–29 (2004)
14. Balogun, A.O., et al.: Data sampling-based feature selection framework for software defect prediction. In: The International Conference on Emerging Applications and Technologies for Industry 4.0, pp. 39–52. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-80216-5
15. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
16. El-Shorbagy, S.A., El-Gammal, W.M., Abdelmoez, W.M.: Using SMOTE and heterogeneous stacking in ensemble learning for software defect prediction. In: The 7th International Conference, pp. 44–47. ACM Press (2018)
17. Tantithamthavorn, C., Hassan, A.E., Matsumoto, K.: The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. IIEEE Trans. Softw. Eng. **46**, 1200–1219 (2020)
18. Xie, Z., Jiang, L., Ye, T., Li, X.: A synthetic minority oversampling method based on local densities in low-dimensional space for imbalanced learning. In: International Conference on Database Systems for Advanced Applications, pp. 3–18. Springer, Cham (2015). https://doi.org/10.1007/978-3-030-73200-4
19. Kamalov, F., Elnagar, A., Leung, H.H.: Ensemble learning with resampling for imbalanced data. In: Huang, D.-S., Jo, K.-H., Li, J., Gribova, V., Hussain, A. (eds.) ICIC 2021. LNCS, vol. 12837, pp. 564–578. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84529-2_48
20. Cai, X., et al.: An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search. Concurr. Comput. Pract. Exp. **32**, e5478 (2020)
21. 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**, 15 (2019)
22. Goyal, S.: Handling class-imbalance with KNN (Neighbourhood) under-sampling for software defect prediction. Artif. Intell. Rev. **55**, 1–42 (2021)
23. Cao, Y., Ding, Z., Xue, F., Rong, X.: An improved twin support vector machine based on multi-objective cuckoo search for software defect prediction. Int. J. Bio-Insp. Comput. **11**, 282–291 (2018)
24. 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)
25. Yu, Q., Jiang, S., Zhang, Y.: The performance stability of defect prediction models with class imbalance: an empirical study. IEICE Trans Info Sys. **100**, 265–272 (2017)
26. Menzies, T., Greenwald, J., Frank, A.: Data mining static code attributes to learn defect predictors. IIEEE Trans. Softw. Eng. **33**, 2–13 (2007)

27. Balogun, A.O., et al.: SMOTE-based homogeneous ensemble methods for software defect prediction. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12254, pp. 615–631. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58817-5_45
28. Mockus, A., Weiss, D.M.: Predicting risk of software changes. Bell Labs Tech. J. **5**, 169–180 (2000)
29. Bowes, D., Hall, T., Petrić, J.: Software defect prediction: do different classifiers find the same defects? Softw. Qual. J. **26**(2), 525–552 (2017). https://doi.org/10.1007/s11219-016-9353-3
30. Japkowicz, N.: The class imbalance problem: Significance and strategies. In: Proceedings of the International Conference on Artificial Intelligence, vol. 56, pp. 111–117. Citeseer (2000)
31. Peng, M., et al.: Trainable undersampling for class-imbalance learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 4707–4714 (2019)
32. Han, H., Wang, W.Y., Mao, B.H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005). https://doi.org/10.1007/11538059_91
33. Elhassan, T., Aljurf, M.: Classification of imbalance data using tomek link (t-link) combined with random under-sampling (rus) as a data reduction method. Global J. Technol. Optim. S **1** (2016)
34. Malhotra, R., Kamal, S.: An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data. Neurocomputing **343**, 120–140 (2019)
35. Rodriguez, D., Herraiz, I., Harrison, R., Dolado, J., Riquelme, J.C.: Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In: The 18th International Conference, pp. 1–10. ACM Press (2014)
36. Suresh Kumar, P., Behera, H.S., Nayak, J., Naik, B.: Bootstrap aggregation ensemble learning-based reliable approach for software defect prediction by using characterized code feature. Innov. Syst. Softw. Eng. **17**(4), 355–379 (2021). https://doi.org/10.1007/s11334-021-00399-2
37. Berrar, D.: Bayes' theorem and naive Bayes classifier. Encyclop. Bioinform. Comput. Biol. ABC Bioinform. **403** (2018)
38. Balogun, A.O., et al.: Empirical analysis of rank aggregation-based multi-filter feature selection methods in software defect prediction. Electronics **10**, 179 (2021)
39. 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)
40. 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)
41. Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K.: An empirical comparison of model validation techniques for defect prediction models. IIEEE Trans. Softw. Eng. **43**, 1–18 (2016)
42. Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K.: Comments on "researcher bias: the use of machine learning in software defect prediction." IIEEE Trans. Softw. Eng. **42**, 1092–1094 (2016)
43. Yu, Q., Jiang, S., Zhang, Y.: The performance stability of defect prediction models with class imbalance: an empirical study. IEICE Trans **E100.D**. Inf. Syst., 265–272 (2017)

44. Balogun, A.O., Akande, N.O., Usman-Hamza, F.E., Adeyemo, V.E., Mabayoje, M.A., Ameen, A.O.: Rotation forest-based logistic model tree for website phishing detection. In: Gervasi, O., et al. (eds.) ICCSA 2021. LNCS, vol. 12957, pp. 154–169. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87013-3_12
45. Li, R., Zhou, L., Zhang, S., Liu, H., Huang, X., Sun, Z.: Software defect prediction based on ensemble learning. In: DSIT 2019: 2019 2nd International Conference on Data Science and Information Technology, pp. 1–6. ACM (2019)

View publication stats