Efficiency of oversampling methods for enhancing software defectprediction by using imbalanced data

Dr.B.Tirimula Rao
T.Karunya

Outline

Introduction

Software defect prediction is important to identify defects in the early phases of software development life cycle. This early identification and thereby removal of software defects is crucial to yield a cost-effective and good quality software product.

Use of **machine learning techniques** for software defect prediction, yields biased results when applied on **imbalanced data sets**.

Use of imbalanced datasets leads to off-target predictions of the minority class, which is generally considered to be more important than the majority class. Thus, **handling imbalanced data** effectively is crucial for successful development of a competent defect prediction model.

Introduction - Research Investigates

1. **What is the impact of using oversampling techniquesto generate defect prediction models by using ML classi-fiers?**

2. **Do balanced datasets improve the prediction capa-bility of ML classifiers? If yes, what is the extent ofimprovement?**

3. **Which oversampling technique used provides the optimum results to develop a defect prediction model?**

Introduction - Methodology

This research evaluates the effectiveness of five machine learning classifiers for software defect prediction on nine imbalanced NASA datasets by application of sampling methods.

We investigate five existing oversampling methods, which replicate the instances of minority class.

Background - Class Imbalance Problem

Datasets with a nonuniform distribution of class values suf-fer from class imbalance problems. For example, considera dataset whose 95% class values belong to one class andonly 5% class values belong to another class; this dataset isimbalanced. Only few instances of minority class values arereported for hundreds, thousands, or millions of instances ofmajority classes or class values. Thus, when ML classifiersare implemented on these datasets, the prediction capabil-ity of models decreases, and the model provides off-targetresults.

1. **Machine Learning Techniques** :
   - Decision Tree
   - Random Forest
   - Naive Bayes
   - Adaboost
   - Bagging
2. **Imported machine learning techniques from various packages of sklearn** :

```
from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
```

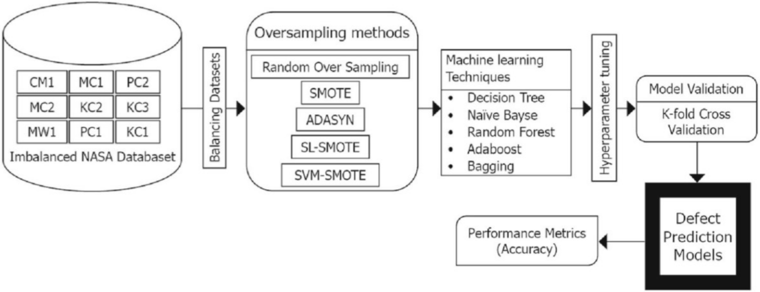
1. **OverSampling Techniques** :
    - SMOTE (Synthetic Minority Oversampling Technique)
    - SL-SM (Safe-level SMOTE)
    - ADASYN (Adaptive synthetic sampling technique)
    - SVM-SMOTE (Support Vector Smote)
    - Random OverSampler
2. **Imported OverSampling techniques from imblearn.overSampling package** :

```
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import SVMSMOTE
from imblearn.over_sampling import RandomOverSampler
from imblearn.over_sampling import SLSM
from imblearn.over_sampling import ADASYN
```

1. **Comparing OverSampling Techniques** :

OverSampling Technique	***Random Oversampler***	***SMOTE***	***ADASYN***	***SL-SM***	***SVM-SMOTE***
Minority class instances	Duplicates the values of minority classes	Between random data points and k-nearest neighbors	Using SMOTE and Borderline Smote	Done around a larger safe level	Done within the parameters of support vectors
Number of minority class instances	Equal to the number of majority class values	Proportional to the number of instances in majority class	Proportional to the Density distribution of minority class	Proportional to the number of instances in majority class	Proportional to the number of instances in majority class

Experiment Design



Datasets - Description

- This study used nine NASA datasets from the PROMISE repository. These datasets comprise Halstead metrics, McCabe metrics, size metrics, and other features that are used to determine whether a software model is defective.
- The class values of the datasets indicate whether the dataset is faulty.
- If the class value is "0" or "no," then the model is not defective; if it is "1" or "yes," the model is defective. The datasets used are incredibly biased (imbalanced) to slightly biased (balanced) with minority class samples (i.e., number of faulty modules) in a range of 0.1%-25.5%.

Datasets - Description

Dataset	Num_attributes	Num_modules	% Of defective modules	Class_values	
				Defective (true)	Non-defective (false)
CM1	22	344	12.21	49	449
JM1	22	7782	21.48	326	1783
KC2	22	522	20.5	107	415
KC3	40	194	18.55	36	158
MC1	39	1988	2.31	46	1942
MC2	40	125	35.2	44	81
MW1	38	253	10.67	27	226
PC1	22	705	8.03	77	1032
PC2	37	1585	1	16	729

Environment setting

- The proposed methods are implemented on a laptop with anIntel(R) Core(TM) i7-8565U CPU@ 1.80 GHz 1992 MHz,4 core(s), eight logical processor(s), and the Microsoft Win-dows 10 Pro 64-bit operating system.
- A $\times$ 64-based processor with Jupyter notebook 6.1.4, python 3.8.5., sklearn 1.0.2 (MLclassifiers are imported from sklearn lib), imblearn 0.8.1(oversampling methods are obtained from imblearn), mat-plotlib 3.2.2, and seaborn 0.11.2 are the libraries used forplotting graphs.

Methodology

1. **HyperParameters For 5 Machine Learning Techniques over 9 datasets are calculated using GridSearch CV**
2. **Compare performance of ML models after Oversampling:**
   - SMOTE (Synthetic Minority Oversampling Technique)
   - SL-SM (Safe-level SMOTE)
   - ADASYN (Adaptive synthetic sampling technique)
   - SVM-SMOTE (Support Vector Smote)
   - Random OverSampler
3. **Comparative analysis of Oversampling Techniques to find out best oversampling method to improve the performance of ML techniques for software defect pre-diction in this study**

Methodology - GridSearchCV

Hyperparameter Tuning with GridSearchCV

It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. As the performance of a model significantly depends on the value of hyperparameters.
GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.

Methodology - Hyperparameters

HYPER PARAMETER TUNING															
Datasets	Machine Learning Techniques														
	Decision Tree				**Random Forest**					**Adaboost**			**Bagging**		
	criteria	*Max depth*	*Min Samples leaf*	*Min Samples split*	*Max depth*	*Max features*	*Min Samples leaf*	*Min Samples split*	*N-estimator*	*Base estimator*	*N-estimator*	*Algorithm*	*Base estimator*	*Max depth*	*Max samples*
CM1	gini	2	2	4	80	2	3	10	100	decision tree	200	SAMME.R	decision tree	1	0.1
KC1	gini	3	1	2	90	3	3	10	100	Decision tree	100	SAMME.R	Decision tree	5	0.5
KC2	gini	5	2	2	80	3	4	8	100	Decision tree	300	SAMME.R	Decision tree	1	0.1
KC3	gini	2	2	4	80	2	3	10	100	Decision tree	200	SAMME.R	Decision tree	5	0.2
MC1	gini	2	1	4	80	2	3	8	100	Decision tree	300	SAMME.R	Decision tree	5	0.5
MC2	entrop y	3	1	3	80	3	3	10	100	decision tree	300	SAMMER	decision tree	1	0.2
MW1	gini	3	1	4	80	2	3	8	100	decision tree	200	SAMMER	decision tree	5	0.1
PC1	entrop y	5	4	4	80	3	4	8	100	decision tree	100	SAMMER	decision tree	5	0.5
PC2	gini	1	1	2	80	2	3	8	100	decision tree	100	SAMMER	decision tree	1	0.05

Performance measure

- The confusion matrix is used to assess the accuracy of a model.Matrix comprises four variables: true-positive(TP), true-negative(TN), false-positive(FP), and false-negative(FN).
- Each variable in the matrixhas its significance; true-positive indicates the number ofinstances that are defective and predicted inappropriately.True-negative indicates the cases are nondefective and areexpected to be nondefective. False-positive value shows thenumber of instances that are nondefective but are predicteddefective. False-negative indicates the number of defectivecases that are predicted nondefective

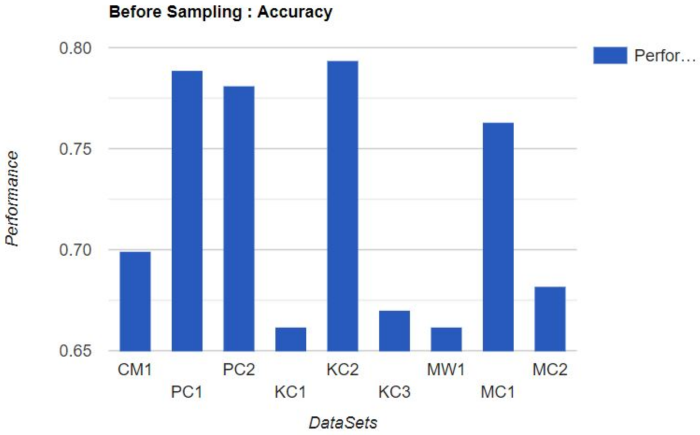
accuracy

It is the most intuitive performance measure. It is simply a ratio of correctly predicted observations to the total words.
Accuracy $=$ TP $+$ TN$/($TP$+$TN$+$FP$+$FN$)$

1. **Performance Accuracy of Machine Learning Techniques Before OverSampling :**

NO_SAMPLING									
ML Technique	***CM1***	***PC1***	***PC2***	***KC1***	***KC2***	***KC3***	***MW1***	***MC1***	***MC2***
Decision tree	0.59	0.719	0.448	0.616	0.704	0.653	0.449	0.521	0.698
Naïve Bayes	0.69	0.844	0.836	0.701	0.825	0.736	0.716	0.883	0.717
Random Forest	0.76	0.768	0.877	0.633	0.832	0.661	0.728	0.708	0.702
Adaboost	0.71	0.793	0.914	0.669	0.784	0.573	0.711	0.842	0.616
Bagging	0.72	0.820	0.828	0.689	0.825	0.729	0.705	0.860	0.676
Avg	*0.699*	*0.789*	*0.781*	*0.662*	*0.794*	*0.670*	*0.662*	*0.763*	*0.682*

Results - NoSampling : Performance



Results - Datasets: After OverSampling

Dataset	Class Values	Before OverSampling	After OverSampling				
			SMOTE	*ADASYN*	*SL-SM*	*SVM-SMOTE*	*Random Oversampler*
CM1	*True*	39	359	373	359	141	359
	False	359	359	359	359	359	359
PC1	*True*	824	824	824	824	824	824
	False	63	824	803	824	824	824
PC2	*True*	584	584	584	584	584	584
	False	12	584	584	584	584	584
KC1	*True*	256	1431	1415	1431	1431	1431
	False	1431	1431	1431	1431	143	1431
KC2	*True*	94	323	322	323	323	323
	False	323	323	323	323	323	323
KC3	*True*	33	33	122	122	122	122
	False	122	122	125	122	122	122
MC1	*True*	38	1552	1550	1552	826	1552
	False	1552	1552	1552	1552	1552	1552
MC2	*True*	67	67	67	67	67	67
	False	33	67	61	67	67	67
MW1	*True*	178	178	178	178	178	178
	False	24	178	173	178	109	178

1. **Performance Accuracy of datasets after application of Oversampling Techniques** :

CM1	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.68	0.63	0.77	0.76	0.84
Nb	0.87	0.84	0.83	0.85	0.85
Rf	0.87	0.85	0.87	0.86	0.89
Ada	0.85	0.84	0.87	0.80	0.86
bg	0.71	0.72	0.66	0.74	0.87

Results - Performance:After OverSampling

MC2	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.60	0.64	0.64	0.56	0.68
Nb	0.68	0.68	0.68	0.68	0.68
Rf	0.64	0.64	0.60	0.60	0.68
Ada	0.56	0.56	0.64	0.64	0.64
bg	0.56	0.72	0.68	0.64	0.56

PC1	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.72	0.84	0.85	0.72	0.73
Nb	0.90	0.88	0.89	0.90	0.89
Rf	0.95	0.94	0.94	0.95	0.95
Ada	0.89	0.89	0.92	0.92	0.95
bg	0.74	0.76	0.77	0.77	0.91

MW1	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.84	0.82	0.86	0.70	0.88
Nb	0.80	0.80	0.80	0.80	0.80
Rf	0.86	0.90	0.86	0.88	0.88
Ada	0.88	0.84	0.86	0.86	0.88
bg	0.80	0.78	0.78	0.82	0.82

PC2	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.79	0.83	0.83	0.97	0.79
Nb	0.68	0.61	0.59	0.80	0.68
Rf	0.97	0.96	0.97	0.97	0.97
Ada	0.97	0.95	0.94	0.96	0.97
bg	0.79	0.77	0.78	0.97	0.79

Results - Performance:After OverSampling

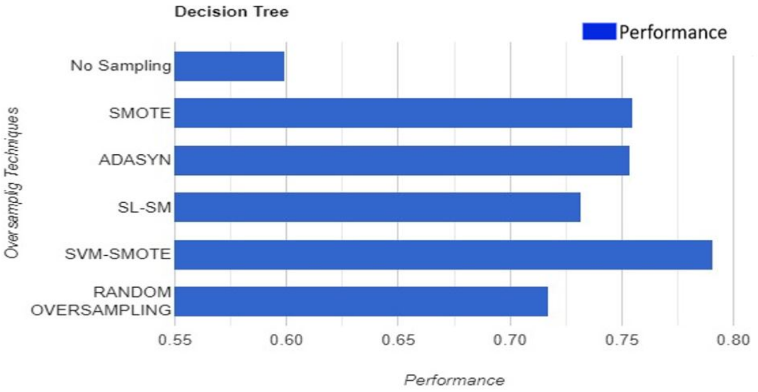
KC2	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.68	0.80	0.80	0.81	0.80
Nb	0.89	0.89	0.89	0.89	0.89
Rf	0.79	0.82	0.80	0.82	0.82
Ada	0.80	0.77	0.82	0.79	0.82
bg	0.79	0.79	0.80	0.78	0.78

KC3	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.64	0.87	0.64	0.64	0.87
Nb	0.69	0.69	0.69	0.69	0.79
Rf	0.79	0.76	0.79	0.74	0.79
Ada	0.79	0.82	0.79	0.71	0.79
bg	0.61	0.64	0.56	0.64	0.64

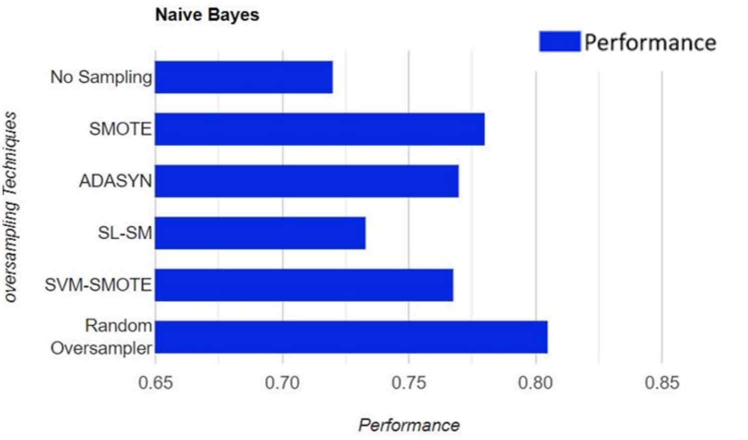
MC1	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.86	0.79	0.77	0.78	0.85
Nb	0.93	0.88	0.85	0.22	0.47
Rf	0.97	0.97	0.97	0.97	0.97
Ada	0.93	0.95	0.95	0.96	0.97
bg	0.77	0.74	0.75	0.81	0.85

KC1	Random Sampler	SMOTE	ADASYN	SL-SM	SVM-SMOTE
Dt	0.65	0.58	0.63	0.65	0.68
Nb	0.81	0.80	0.79	0.77	0.81
Rf	0.80	0.80	0.81	0.81	0.80
Ada	0.72	0.79	0.77	0.79	0.80
bg	0.65	0.67	0.63	0.66	0.68

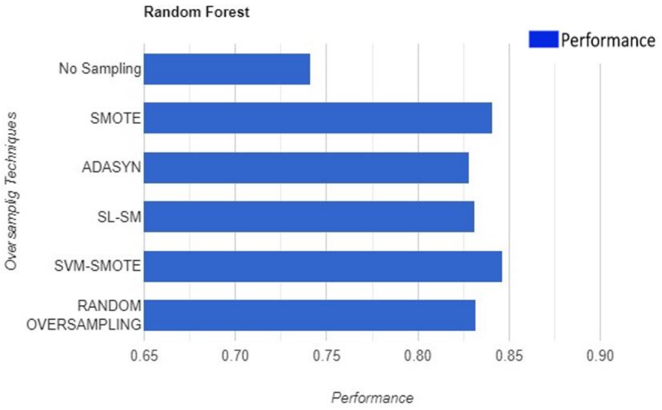
Results - Comparison:Performance before and after OverSampling



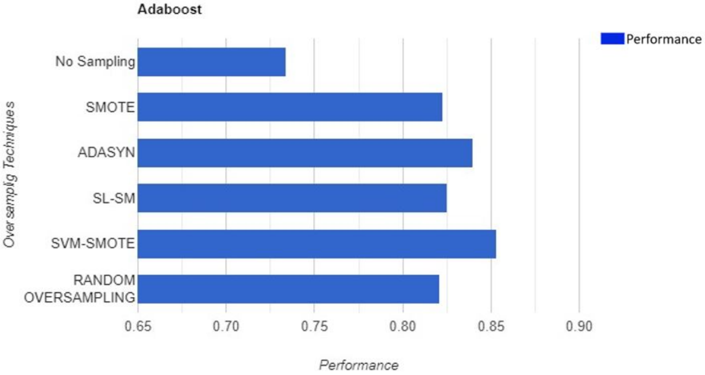
Results - Comparison:Performance before and after OverSampling



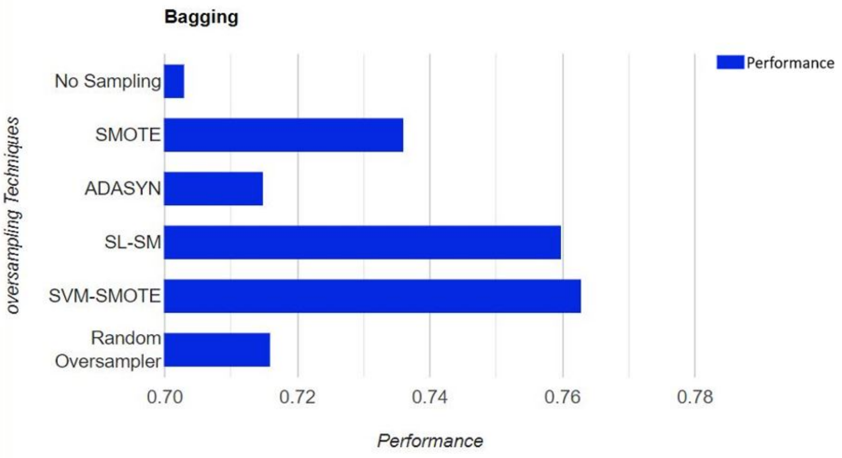
Results - Comparison:Performance before and after OverSampling



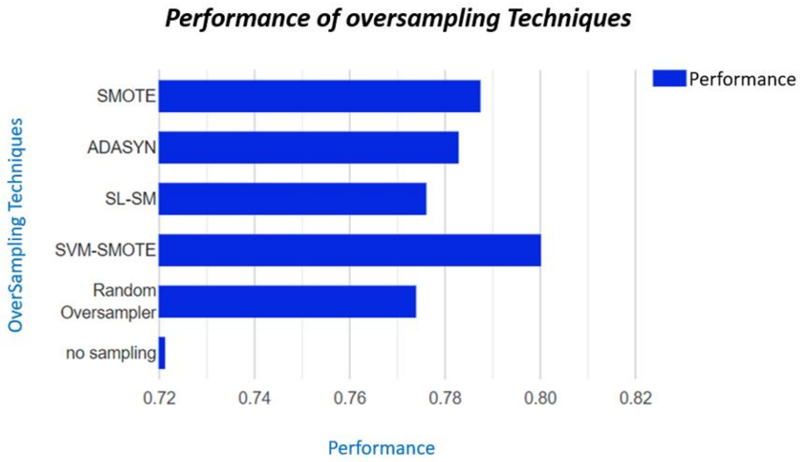
Results - Comparison:Performance before and after OverSampling



Results - Comparison:Performance before and after OverSampling



Results - Comparison: OverSampling Techniques



Research Conclusion

1. **What is the impact of using oversampling techniquesto generate defect prediction models by using ML classifiers?**
   - Before application of Oversampling Techniques (that is NoSampling)the performance of machine learning techniques for defect prediction models is around 60-70 percent and when oversampling techniques are applied the performance increased to 85-90 percent.
2. **Do balanced datasets improve the prediction capa-bility of ML classifiers? If yes, what is the extent ofimprovement?**
   - Yes,balancing of datasets using oversampling methods improves the performance of ML techniques. the extent of improvement in the performance of ML techniques is 15-20 percent.

Research Conclusion

1. **Which oversampling technique used provides the optimum results to develop a defect prediction model?**
   - From the results from performance analysis, it can be noted that SVM-SMOTE both SMOTE are perhaps the best oversampling method to improve the performance of ML techniques for software defect prediction in this study.