Cross-Project Pooling of Defects for Handling Class Imbalance

J. M. Catherine[1†] and S Djodilatchoumy[2††],
Catherine_vinod@yahoo.co.in djodilatchoumy@hotmail.com
[†] CTTE College for Women, Chennai, Tamil Nādu, India
[††]Chellammal College for Women, Chennai, Tamil Nādu, India

Summary
Applying predictive analytics to predict software defects has improved the overall quality and decreased maintenance costs. Many supervised and unsupervised learning algorithms have been used for defect prediction on publicly available datasets. Most of these datasets suffer from an imbalance in the output classes. We study the impact of class imbalance in the defect datasets on the efficiency of the defect prediction model and propose a CPP method for handling imbalances in the dataset. The performance of the methods is evaluated using measures like Matthew's Correlation Coefficient (MCC), Recall, and Accuracy measures. The proposed sampling technique shows significant improvement in the efficiency of the classifier in predicting defects.
Keywords:
Software Defect, Random Forest, Matthews Correlation Coefficient, Accuracy, Dataset Imbalance handling

1. Introduction

In the ever-changing and ever-evolving world, the software industry has undergone rapid changes both in terms of technology and product. The software has improved from merely accepting text or image input to processing more real-world and real-time data inputs. As the complexity of the inputs increases, the code gets complex giving rise to defects. The software's confusing source code triggers software bugs, which can lead to software failure. Prediction of a software defect involves, building classifiers using machine learning algorithms to predict code segments that have flaws in them. The prediction is done by using the historical knowledge in software libraries and `changing documents to formulate the measures for identifying software defects. Prediction of software error can be accomplished by training the classifier using the modules within the project or using modules of a similar project. The key focus areas of software defect prediction include building classifiers, comparing, and analyzing the performance of various classifiers, analyzing the impact of various parameters on the efficiency of defect prediction, and so on.

One of the major issues in software defect prediction is the "Class Imbalance" [1] problem. It is a major factor that affects the quality and efficiency of defect prediction. Class distribution is described as the number of instances of each class in the training dataset. The datasets in which the distribution of classes is highly disproportionate are termed as imbalanced datasets. The efficiency of the defect prediction models is highly dependent on the class distribution of the training data. If the number of instances belonging to one class is much more than the number of instances belonging to another class, then the problem is known as the class imbalance problem.

In the defect prediction datasets, the number of records belonging to the 'buggy' class is very less compared to the 'clean' class. This imbalance leads to inaccurate training of the prediction models, which in turn leads to wrong predictions. The main objective of this study is to study the impact of feature selection on the software defect dataset whose features are not discrete components but are metrics that are semantically interrelated. This paper mainly focuses on the strategies available for alleviating the problem of class imbalance in software defect datasets. In this study, we analyze the various data level and classifier level approaches for class imbalance handling. The research questions include:

RQ1: Is there a significant impact of the sampling technique on the performance of the dataset

RQ2: Is our proposed CPP method able to improve the performance of the model significantly

The main contributions of this paper include:

- Analysis of the impact of various class imbalance handling approaches available on the efficiency of defect prediction.
- A hybrid imbalance handling method that combines data level and algorithm level approaches for effective defect prediction.

The rest of the paper is organized into five sections. Section 2 presents the background of imbalance handling methods. Section 3 presents the existing work in the area of class imbalance handling. Section 4 presents the methodology used to answer the research questions. Section 5 describes the experimental results. Section 6 provides the conclusions.

Manuscript received October 5, 2022
Manuscript revised October 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.10.2

2. Background

In most of the datasets, the actual characteristic that needs to be detected is represented by a very small number of instances, compared to the normal occurrences. When the dataset is used to train the prediction model, it leads to unfair training, which in turn results in misclassification. Standard classifiers are biased towards the majority class and thus fail to capture rare phenomena depicted by the minority class. Several data-level and algorithm-level approaches have been proposed in the literature for handling this class imbalance.

Data-level approaches include resampling the data to either increase the minority class instances or decrease the majority class instances. This first approach, where additional minority class instances are added to the dataset, is known as Oversampling. The latter, where the majority class instances are deleted to obtain balance is referred to as Under sampling. Under sampling techniques have been found to improve performance, but it also has the disadvantage of discarding potentially useful information. Whereas Oversampling does not lead to loss of information, it places a huge load on memory as well as leads to overfitting.

While Data-level approaches focus on the data, the Algorithm-level approach focuses on modifying the classifier algorithm to effectively handle imbalanced datasets. Bootstrap Aggregation or Bagging is a technique that generates 'n' bootstrap samples with replacement from the dataset and then aggregates them at the end. It reduces overfitting and creates strong learners. On the other hand, Boosting technique combines several weak learners to create a strong learner in order to make accurate predictions.

3. Related Work

In [5], *H. He et al*, proposed a RIPPER classifier based on Ensemble MultiBoost, to reduce the dimensionality of the dataset and to remove redundant values. The approach picks out the most effective features from the original dataset using Principal Component Analysis (PCA) to achieve this.

In [7], *K. E. Bennin et al*, use a self-organizing method in data mining to predict software defects. They consider software metric parameters as the influencing factor. A model for software defect prediction using the chromosomal theory of inheritance was proposed in this work. Two distinct sub-classes are treated as parents which generate new instances. The new instance inherits distinct traits from both parents and thus creates diversity within the dataset.

In [11], *X. Jing et al*, showed that datasets with highly imbalanced data usually result in misclassification of defects. A unified framework for predicting defects in these datasets has been proposed using Subclass discriminant analysis (SDA) for both within and cross-project defect prediction and proved it to be very effective.

In [10], *L. Gong et al* proposed a Cluster-based Over-sampling with noise filtering (KMFOS) approach to tackle the class imbalance problem in SDP. KMFOS works by dividing the defective instances into K- clusters. Interpolation between instances of each two cluster is used to generate new defective instances. This would cause these new defective instances to diversely spread in the space of defective datasets. This cluster-based over-sampling is extended through the Closest List Noise Identification (CLNI) to clean the noise instances.

In [3], Farhad *et al*, proposed a hybrid algorithm using Particle Swarm Optimization (PSO) and Sparrow Search Algorithm (SSA) for estimation of the parameters of a software defect prediction model was proposed in this work. With the support of the new fitness function, it effectively solved the problems of slow convergence speed and low accuracy of the solution. The experimental results showed that the hybrid SSA-PSO could obtain a better solution, convergence speed, and stability than a single SSA and PSO in software defections estimation and prediction.

4. Proposed Method

Our proposed Cross Project Pooling method (CPP) handles class imbalance by pooling all defective instances across projects and appending them to the dataset to balance the number of instances of each class. This method is more advantageous than synthetic sampling methods, as the semantic relationship between the various metrics is maintained.

4.1 Dataset

The datasets have been obtained from PROMISE [4] open-source repository. Three datasets have a high imbalance ratio (i.e ratio of clean to defective modules is less than or approximately equal to 15%). The dataset contains values of 20 object-oriented metrics for the software modules in the project. The output class specifies the defect proneness of each of the modules, given its attribute values. Table 1 presents the summary of the three datasets.

Table 1: Dataset summary

Dataset	Instances	Clean	Buggy	%Defective
prop-6	644	583	61	10.463
poi-2.0	314	277	37	13.357
jedit-4.2	367	319	48	15.047

4.2 Methodology

The methodology used in this study can be divided into four steps. Figure 1 depicts the methodology.

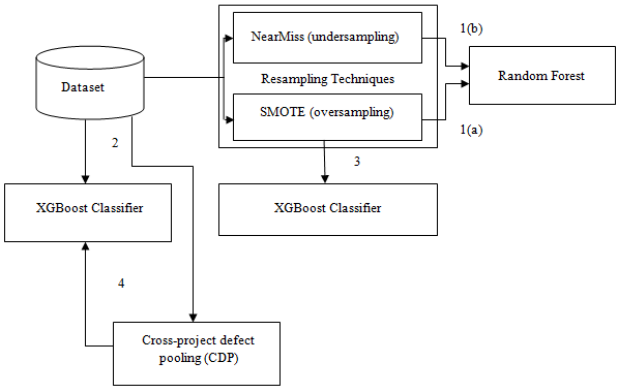


Fig 1: Methodology

The steps are as below:

Step 1: Applications of Resampling techniques

a. Apply SMOTE oversampling technique to generate new instances of the minority class, and apply step 1 to the resultant dataset.
b. Apply the NearMiss undersampling technique to remove instances of the majority class, and apply step 1 to the resultant dataset.

Step 2: Application of Boosting technique. Classify the dataset using XGBoost classifier and tabulate the values of Accuracy, Recall and MCC.

Step 3: Classify the dataset using SMOTE and XGBoost classifier and tabulate the values of Accuracy, Recall and MCC.

Step 4: Pool the minority samples from various projects to create a master dataset and append to the existing datasets for creating balance. Apply XGBoost and tabulate the values.

4.3 Performance Measures

The classifier performance is measured using various measures computed from the confusion matrix. The confusion matrix provides a breakup of the classification as below.

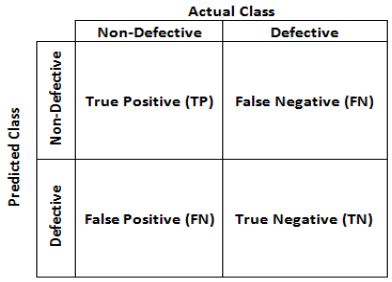


Fig 2: Confusion Matrix

Accuracy: Number of correct predictions to the total number of predictions

Recall or Sensitivity: Number of positive cases classified correctly.

Matthew's Correlation Coefficient (MCC): It produces a high score only if the prediction obtained good results in all the four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements in the dataset [2].

5. Experimental Results

The above-mentioned algorithms were executed using Python 3 Google Compute Engine backend (GPU) with 1.05 GB/ 12.68 GB RAM and 37.35 GB/78.5 GB Hard disk. The confusion matrix for each of the methods is shown in figures 3 to 17. The values of the performance measures namely MCC, accuracy, and recall obtained are tabulated in Tables 2, 3 and 4 respectively.

The performance scores obtained were analysed using one-way ANOVA [10] between columns. The hypothesis is stated as below:

H_0: There is no significant impact of imbalance handling techniques on the performance of the classifier.

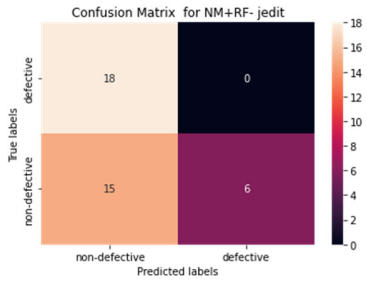


Fig 3: Confusion Matrix for NearMiss (jedit)

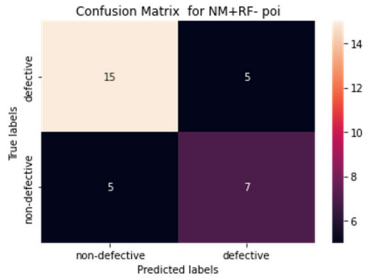


Fig 4: Confusion Matrix for NearMiss (poi)

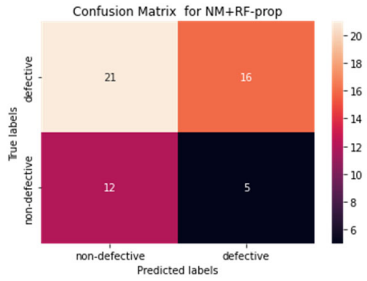


Fig 5: Confusion Matrix for NearMiss (prop)

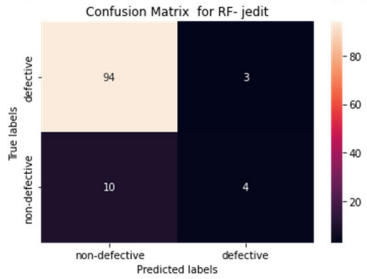


Fig 6: Confusion Matrix for RF(jedit)

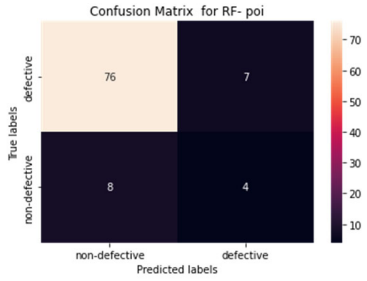


Fig 7: Confusion Matrix for RF (poi)

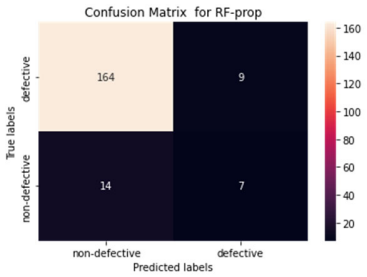


Fig 8: Confusion Matrix for RF (prop)

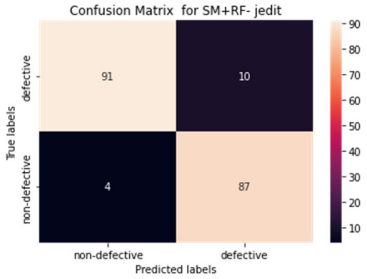


Fig 9: Confusion Matrix for SMOTE (jedit)

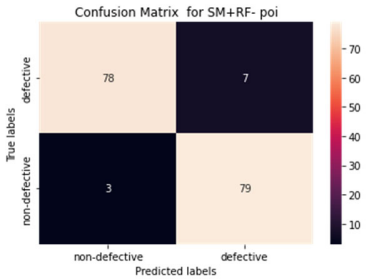


Fig 10: Confusion Matrix for SMOTE (poi)

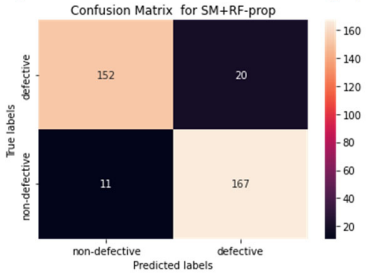


Fig 11: Confusion Matrix for SMOTE (prop)

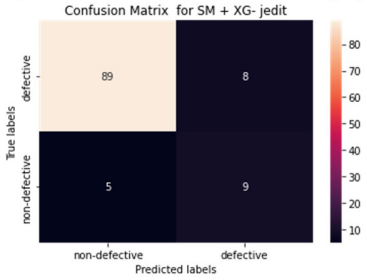


Fig 12: Confusion Matrix for SMOTE +XGBoost (jedit)

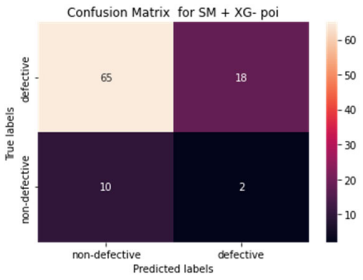


Fig 13: Confusion Matrix for SMOTE+XGBoost (poi)

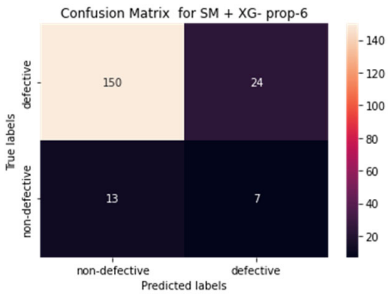


Fig 14: Confusion Matrix for SMOTE+XGBoost(prop)

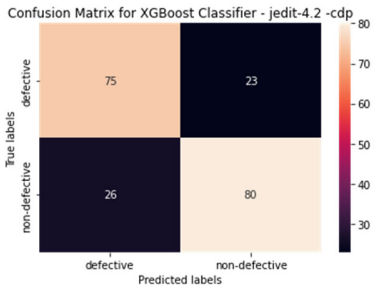


Fig 15: Confusion Matrix for XGBoost(jedit)

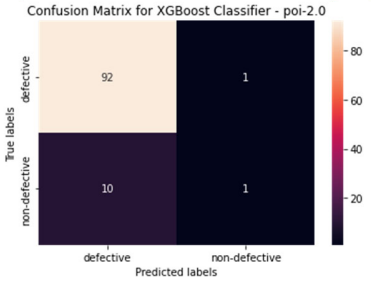


Fig 16: Confusion Matrix for XGBoost(poi)

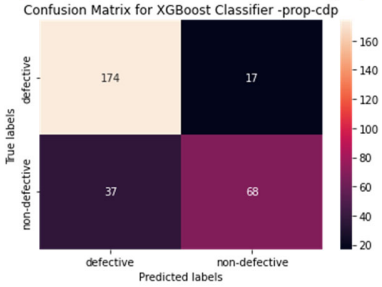


Fig 17: Confusion Matrix for XGBoost(prop)

Table 2: MCC values in %

Dataset	RF	SM	NM	XG	SMX	CPP
prop-6	32	21	17	30	18	59
poi-2.0	39	8	8	18	39	63
jedit-4.2	27	50	52	14	52	52

Table 3: Accuracy values in %

Dataset	RF	SM	NM	XG	SMX	CPP
prop-6	87	86	86	89	81	82
poi-2.0	86	86	89	89	87	82
jedit-4.2	89	87	88	84	86	76

Table 4: Recall values in %

Dataset	RF	SM	NM	XG	SMX	CPP
prop-6	25	35	35	37	35	64
poi-2.0	25	42	50	9	41	76
jedit-4.2	29	50	57	20	64	76

Table 5: ANOVA table summary

Performance Measure	Calculated Value	Table Value (@5%)	Table Value (@1%)	Remarks
Accuracy	4.4	3.105	5.06	H_0 rejected @1% significance while accepted @5%significance
Recall	8.56	3.105	5.06	H_0 rejected @1% and @5%significance
MCC	2.23	3.105	5.06	H_0 accepted @1% and @5%significance

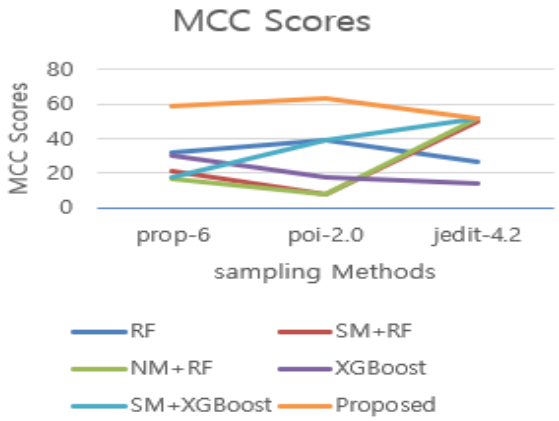
6. Conclusions

To evaluate the RQ1, the results of the one-way ANOVA presented in table 5 are considered. Our hypothesis is accepted when MCC is considered as the performance evaluation method. The accuracy and Recall measure shows a significant difference at 5% significance while rejecting the hypothesis at 1%. Since MCC is proven to be a better measure than accuracy and recall [2], we conclude that sampling significantly impacts the performance of the classifier model.

To answer RQ2, the graph in figure 18 is analyzed. It is evident from the graph that our proposed CPP method

has better MCC scores than the other sampling techniques. It is because our method does not randomly synthesize instances, but contains real-time metric values for the defect-prone datasets.

Fig 18: MCC Scores



References

[1] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. Newsl. 6, 1 (June 2004), 20–29. https://doi.org/10.1145/1007730.1007735

[2] D. Chicco, V. Starovoitov and G. Jurman, "The Benefits of the Matthews Correlation Coefficient (MCC) Over the Diagnostic Odds Ratio (DOR) in Binary Classification Assessment," in IEEE Access, vol. 9, pp. 47112-47124, 2021, DOI: 10.1109/ACCESS.2021.3068614.

[3] Farhad Soleimanian Gharehchopogh, Mohammad Namazi, Laya Ebrahimi, Benyamin Abdollahzadeh. "Advances in Sparrow Search Algorithm: A Comprehensive Survey" , Archives of Computational Methods in Engineering, 2022

[4] G. Boetticher, T. Menzies, and T. J. Ostrand, (2007) Promise repository of empirical software engineering data. [Online]. Available: http://promise.site.uottawa.ca/SERepository.

[5] H. He; X. Zhang; Q. Wang; J. Ren; J. Liu; X. Zhao; Y. Cheng, Ensemble MultiBoost Based on RIPPER Classifier for Prediction of Imbalanced Software Defect Data", IEEE Access, 2019, Vol 7 pp 110333-110343.

[6] Ishani Aroraa, Vivek Tetarwala, Anju Saha, "Open Issues in Software Defect Prediction", Procedia Computer Science 46 (2015) 906 – 912

[7] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden and S. Mensah, "MAHAKIL: Diversity Based Oversampling Approach to Alleviate the Class Imbalance Issue in Software Defect Prediction," *2018 IEEE* Transactions on Software Engineering, Vol 44, Issue 6, pp. 534-550.

[8] Lee, W., Jun, C.-H., Lee, J.-S.: Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification. Inf. Sci. 381, 92–103 (2017)

[9] L. Gong; S. Jiang; L. Jiang, "Tackling Class Imbalance Problem in Software Defect Prediction Through Cluster-Based Oversampling with Filtering", IEEE Access, 2019, Vol 7, pp 145725-145737.

[10] Ostertagova, Eva & Ostertag, Oskar. (2013). Methodology and Application of One-way ANOVA. American Journal of Mechanical Engineering. 1. 256-261. 10.12691/ajme-1-7-21.

[11] X. Jing; F. Wu; X. Dong; B. Xu, "An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project Class-Imbalance Problems," IEEE Transactions on Software Engineering, Vol 43, Issue 4, pp 321-339.

Mrs. J. Mary Catherine pursed Bachelor of Science and Master of Science from Osmania University in year 2001 and 2003 respectively. She is currently pursuing Ph.D in Periyar University, Salem, Tamilnadu and currently working as Assistant Professor in Department of Co mputer Science, Chevalier T. Thomas Elizabeth College for Women, Chennai, Tamilnadu, since 2008. Her main research work focuses on Software Testing, Neural Networks and Algorithms. He has 13 years of teaching experience.

Dr.S.Djodilatchoumy pursed Bachelor of Engineering(Computer Science and Engineering) from Anna University, Chennai, Tamilnadu in 1988, Master of Technology from PunjabUniversity in 2003 and Ph.D. from Mother Terasa University in 2011.Currently working as Head of the Department of Computer Science, Pachaiyappa's College,Chennai, India since 2008. She has published more than 15 research papers in international journals/Conferences and it's also available in online. Her main research work focuses on Neural Networks in Medical diagnosis using Spectral data. She has 11+ years of research experience.