Knowledge-Based Systems 90 (2015) 1–13

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Improving Recall of software defect prediction models using association mining

Zeeshan Ali Rana [a,b,*], M. Awais Mian [a], Shafay Shamail [a]

[a] CS Department, SBA SSE, Lahore University of Management Sciences (LUMS), Lahore, Pakistan
[b] Faculty of IT, University of Central Punjab (UCP), Lahore, Pakistan

ARTICLE INFO

Article history:
Received 10 January 2015
Revised 4 October 2015
Accepted 6 October 2015
Available online 23 October 2015

Keywords:
Software defect prediction
Naive Bayes
PROMISE repository
Imbalanced data
Improving Recall
Association mining

ABSTRACT

Use of software product metrics in defect prediction studies highlights the utility of these metrics. Public availability of software defect data based on the product metrics has resulted in the development of defect prediction models. These models experience a limitation in learning Defect-prone (*D*) modules because the available datasets are imbalanced. Most of the datasets are dominated by Not Defect-prone (*ND*) modules as compared to *D* modules. This affects the ability of classification models to learn the *D* modules more accurately. This paper presents an association mining based approach that allows the defect prediction models to learn *D* modules in imbalanced datasets. The proposed algorithm preprocesses data by setting specific metric values as missing and improves the prediction of *D* modules. The proposed algorithm has been evaluated using 5 public datasets. A Naive Bayes (NB) classifier has been developed before and after the proposed preprocessing. It has been shown that *Recall* of the classifier after the proposed preprocessing has improved. Stability of the approach has been tested by experimenting the algorithm with different number of bins. The results show that the algorithm has resulted in up to 40% performance gain.

1. Introduction

Early identification of defect-prone modules helps in improving software process control, achieving reduced defect correction effort and hence, reduced cost and high software reliability [1–4]. Managing resources during testing is considered a non-trivial task [5] and the identification of defect-prone modules helps in planning resources during testing [5–7]. This identification is done using defect prediction techniques that also help in controlling software projects and developing resource and test plans [6,8].

Software defect prediction techniques either classify a software module as Defect-prone (*D*) or Not Defect-prone (*ND*) or predict the number of defects in a software. Both types of models take software metrics as input. Significant number of software defect prediction models are based on product metrics [9,10]. Use of product metrics for defect prediction has been criticized for being unable to find causal relationship between the metrics and software defects [2,11]. Despite the critique metrics collected from static analysis have been made publicly available [12] and have encouraged the development of numerous prediction models. These metrics are available in the form of datasets which are considered to be the benchmarks in the domain of software quality. Availability of alternate benchmarks is not well known.

Although the available datasets have been useful to develop models with good *Recall* (proportion of correctly predicted *D* modules from all *D* modules) and Area Under the Curve (*AUC* can be considered as probability that a model will give higher score to a randomly chosen *D* module than to a randomly chosen *ND* module) [13–17], these datasets have the following limitations: (1) they are imbalanced, (2) the static code attributes available in the datasets have limited information content [13]. Most of the datasets have significantly larger number of *ND* modules as compared to *D* modules. Smaller number of *D* modules (as training examples) affects the ability of classification models to learn the *D* modules more accurately. Class imbalance is considered a problem in the domain of software defect prediction [18–20] as well as in other domains [21–24]. The factor of limited information content suggests that simple learners (like Naive Bayes) can perform as good as any complex learners (such as J48) [13].

Classifying a *D* module correctly is very important in software defect prediction. Software development organizations cannot afford to ship defective modules to customers, therefore, they strive to detect as many defective modules as possible before the release of software. Significance of correctly identifying defect prone modules can be seen through Pareto principle as applied in software engineering. The principle says that 80% of the defects are located in 20% of code [25].

* Corresponding author at: CS Department, SBA SSE, Lahore University of Management Sciences (LUMS), Lahore, Pakistan. +92 3004239327.
E-mail addresses: zeeshanr@lums.edu.pk (Z.A. Rana), awais@lums.edu.pk (M.A. Mian), sshamail@lums.edu.pk (S. Shamail).

http://dx.doi.org/10.1016/j.knosys.2015.10.009
0950-7051/

Table 1
Two major views in software defect prediction.

	View 1	View 2
Focus	Use of all metrics. Classification models based on correlation etc.	Causes of defects. Relationship between software metrics and defects.
Use of static code metrics	Use as they are	Suggest to incorporate expert opinion
Nature	Empirical	Expert opinion based
Availability of data	Most of the times data is publicly available	Most of the times data not publicly available
Examples of studies in each group	[15,31–33,42–46]	[2,11,47–49]

It also gives an insight regarding high cost of misclassifying *D* modules. In the domain of software defect prediction, mostly standard machine learning algorithms are used which do not directly address the issue of class imbalance [20]. The standard learning algorithms are biased towards the dominant class and may not perform at their best in imbalanced datasets [21]. The standard algorithms also have the tendency to discard the scarce class by identifying it as noise [21]. Therefore, researchers perform over sampling in scarce class or under sampling in dominant class before using the learning algorithm [21]. This is done to get a balanced distribution of classes such that standard algorithms designed for learning through balanced training set can work in the same manner for both the classes.

The challenge of class imbalance is also faced in the domain of direct marketing where profit needs to be maximized even when significantly large number of customers are not likely to respond as compared to number of customers who are likely to respond [22]. Similarly charity organizations need to contact potential donors from a list of people who have pledged to donate and there is a small proportion of pledges that is fulfilled [22]. In this domain the problem of imbalance has been addressed through association mining (AM). Association Rule Mining (ARM) is an important data mining technique and is employed for discovering interesting relationships between variables in large databases [26]. ARM is used to find *interesting correlations, frequent patterns, associations* or *causal structures* among items of large datasets [27]. Fuzzy logic has also been applied in other domains to address the issue of class imbalance [23].

Due to the factor of limited information content the defect prediction models have reached a performance ceiling which can be crossed if information content is improved. This information content can be improved by collecting more insightful data or accessing and combining relevant features available at the time of model development [13]. In the case of most of the public datasets where the additional data required to improve the information is not available, use of data in an insightful and different way can be useful to improve quality of defect prediction.

This paper uses the available information in the public datasets in an effective manner, applies association mining (AM) to find association between software metrics and software defects, and improves performance of classification model in imbalanced datasets. The datasets are preprocessed using the proposed approach, a defect prediction model is developed using the preprocessed data and performance analysis of the model is performed in terms of *Recall*. The preprocessing step partitions data and finds important itemsets. The important itemsets are relabeled in one partition of the data and the prediction of *D* modules improves as a result. Afterwards, the preprocessed datasets are used for model development and performance of the model is also analyzed. Naive Bayes (NB) classifier (one of the best techniques along with Random Forests in the field of defect prediction [14,17]) has been used as a test case to evaluate the proposed preprocessing. Significance of *Recall* for performance analysis is highlighted through a questionnaire distributed in the software industry. The results show that the proposed approach has improved *Recall* of the NB classifier up to 40%. Stability of the approach has been tested by experimenting with different number of bins.

When lack of detailed information content is reported in literature, it becomes a non-trivial task to use the available information in an effective manner. Preprocessing suggested in this paper is one such attempt that gives a method to use the available data in a new and insightful way. Average *Recall* reported for the datasets used has been below 75% [15,21,28] whereas *Recall* values with the proposed approach vary from 78.6% to 85% with different number of bins. It is pertinent to mention that unlike other studies [29], no additional information has been used with the publicly available static code attributes.

Rest of the paper is organized as follows: Section 2 discusses the related work, Section 3 presents our research methodology and our approach to find focused itemsets. Section 4 presents the results of the experiments whereas Section 5 discusses the results. Section 6 concludes the paper and presents future directions.

2. Related work

Numerous techniques for defect prediction have been reported in literature and comparative studies to evaluate their performance have also been conducted [9,10,13–17,29]. These techniques include models like neural networks, decision trees, Naive Bayes, case based reasoning, fuzzy inference systems, regression trees, association rule mining based models as well as ensemble models like RIPPER, WHICH, DTNB, and FURIA. Most of these techniques are based on data mining methods and dominantly use public datasets available at PROMISE repository [12]. The datasets used contain product metrics as attributes of software modules along with defectiveness information of the software modules. Software attributes, specifically code attributes, have been ascertained to be significant in making defect predictions [15]. Performance achieved using number of data mining techniques suffer from ceiling effect [17] meaning that the proposal of new defect prediction models is not significantly improving quality of defect prediction. Menzies et al. [13,17] suggest that instead of proposing new techniques for defect prediction, focus of research should be on how to use the available data to improve performance of prediction.

Though use of the public datasets and product metrics has been frequent, they have been criticized for the reasons including lack of ability to find causal relationship between software metrics and defect [11] and quality issues related to available datasets [30]. Proponents of using these data have developed numerous models based on these data.

Wang et al. [31] have used feature subset selection algorithms to select the attributes to be used by classifiers including Logistic regression, *K*-nearest neighbor clustering and Multi-layer perceptron. The study has empirically evaluated the minimum number of metrics that can be used for defect prediction to be three. Various other models, for example Quad Tree-Based *K*-Means Algorithm [32] and Analytical Hierarchical Processing (AHP) based ensemble models have been used for defect prediction [33]. WHICH framework [17] is capable of serving customized objectives and has been reported to outperform the data mining schemes employed for predicting defects in software.

Association Rule Mining (ARM) has been useful to predict software defect correction effort and determine association among software defects [34–36]. An association rule based classifier, CBA2, has been empirically evaluated to predict software defects [37].