Information and Software Technology 58 (2015) 388–402

Contents lists available at ScienceDirect

Information and Software Technology

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

Software defect prediction using ensemble learning on selected features

Issam H. Laradji, Mohammad Alshayeb *, Lahouari Ghouti

Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

ARTICLE INFO

Article history:
Received 4 February 2014
Received in revised form 22 May 2014
Accepted 8 July 2014
Available online 24 July 2014

Keywords:
Defect prediction
Ensemble learning
Software quality
Feature selection
Data imbalance
Feature redundancy/correlation

ABSTRACT

Context: Several issues hinder software defect data including redundancy, correlation, feature irrelevance and missing samples. It is also hard to ensure balanced distribution between data pertaining to defective and non-defective software. In most experimental cases, data related to the latter software class is dominantly present in the dataset.
Objective: The objectives of this paper are to demonstrate the positive effects of combining feature selection and ensemble learning on the performance of defect classification. Along with efficient feature selection, a new two-variant (with and without feature selection) ensemble learning algorithm is proposed to provide robustness to both data imbalance and feature redundancy.
Method: We carefully combine selected ensemble learning models with efficient feature selection to address these issues and mitigate their effects on the defect classification performance.
Results: Forward selection showed that only few features contribute to high area under the receiver-operating curve (AUC). On the tested datasets, greedy forward selection (GFS) method outperformed other feature selection techniques such as Pearson's correlation. This suggests that features are highly unstable. However, ensemble learners like random forests and the proposed algorithm, average probability ensemble (APE), are not as affected by poor features as in the case of weighted support vector machines (W-SVMs). Moreover, the APE model combined with greedy forward selection (enhanced APE) achieved AUC values of approximately 1.0 for the NASA datasets: PC2, PC4, and MC1.
Conclusion: This paper shows that features of a software dataset must be carefully selected for accurate classification of defective components. Furthermore, tackling the software data issues, mentioned above, with the proposed combined learning model resulted in remarkable classification performance paving the way for successful quality control.

1. Introduction

With the increasing impact of software applications on day-to-day businesses and activities, software attribute prediction such as effort estimation [1,2], maintainability [3,4], defect [5,6] and quality [7,8] classification are gaining growing interest from both academic and industry communities. Defective software components have devastating consequences on increased development and maintenance costs and declining customer satisfaction [9]. This safety and reliability fact calls for the adoption of rigorous software quality control processes. However, the scarcity of human and financial resources dictates the need for cost-efficient approaches to detect and repair defective components.

Research on software defect prediction emphasized the success of many algorithms including decision trees [10,11], Bayesian methods [12,13], and artificial neural networks multilayer perceptrons (ANN-MLPs) [14]. However, these methods are sub-optimal in the case of skewed and redundant defect datasets [15]. The prediction performance of these methods gets worse when the defect datasets contain incomplete or irrelevant features [16]. Classifiers such as support vector machines (SVMs) [17] and ANN-MLPs [18], biased towards the dominant class, tend to ignore the minority class which results in high false negative rates [19]. It is noteworthy that ensemble learning models are very adequate to address the data issues mentioned earlier. For instance, random forests [20] outperforms the aforementioned algorithms in detecting defective modules even though they are not tuned to directly address imbalanced data [21]. In addition, the voting mechanism in ensemble learning mitigates any residual effect attributed to feature irrelevance and redundancy. This mitigation is carried out

* Corresponding author.
E-mail addresses: g200790850@kfupm.edu.sa (I.H. Laradji), alshayeb@kfupm.edu.sa (M. Alshayeb), lahouari@kfupm.edu.sa (L. Ghouti).

http://dx.doi.org/10.1016/j.infsof.2014.07.005
0950-5849/

by assigning higher weights to individual classifiers that perform well on the tested datasets. The robustness to irrelevant and redundant features certainly enhances the prediction performance. In fact, voting, in its simple form of averaging, ensures the mitigation of the noise effects, which boots the overall prediction performance.

In this paper, we propose a software defect classification method using an average probability ensemble (APE) learning module. The proposed APE system incorporates seven classifiers: random forests (RF), gradient boosting (GB), stochastic gradient descent (SGD), weighted SVMs (W-SVMs), logistic regression (LR), multinomial naive Bayes (MNB) and Bernoulli naive Bayes (BNB). The base classifiers have been selected after extensive simulation validation as indicated in the sequel of the paper. It is worth mentioning that some of these classifiers, such as the RF and W-SVMs models, are considered "de-facto" classifiers in the literature [22]. In addition, the variation in classification capabilities of the base classifiers enable them to capture different statistical characteristics of the underlying data, which makes their combination an added value for the proposed ensemble learning model.

To further improve the classification performance of the proposed ensemble classifier, efficient feature selection is combined with the proposed ensemble model yielding an enhanced ensemble classifier. This enhancement resulted in efficient handling of redundant and irrelevant features in software defect datasets.

Therefore, the objectives of the paper are to demonstrate the positive effect of feature selection on the performance of defect classification and to propose a two-variant ensemble learning algorithm which is robust to both data imbalance and feature redundancy. In addition, the proposed two-variant ensemble algorithm has exhibited stronger robustness to redundant and irrelevant features, which constitutes a major contribution attributed to this paper.

The paper is organized as follows: Section 2 summarizes the related work. Section 3 gives a detailed description of the proposed learning model for software defect prediction. In Section 4, we describe the software defect datasets, the experimental setup and results. Detailed analysis and discussion on the reported resulted are given in Section 5. In Section 6, we present the threats to validity and finally, conclusions, along with suggestions for future work, are given in Section 7.

2. Literature review

A detailed account of current related work is given below. First, general prediction techniques are reviewed followed by a summary of sampling techniques that are used to properly handle data imbalance. Then, an overview of commonly used cost-effective classification techniques is given. The use of ensemble learning models to remedy data imbalance is discussed afterwards. Finally, approaches for feature selection are outlined before the section concludes with a review of other defect classification methods.

2.1. General techniques

Decision trees [10,11], Bayesian methods [12], and ANNs [14] have paved the way for machine learning-based methods in the field of defect classification. These methods use software metrics to properly classify defective software modules. However, it should be noted that these methods often ignore the skewness and other statistical characteristics of the defect datasets. Omitting such characteristics substantially affects, in a negative way, the classification performance [19]. Conventional methods, such as SVMs [17] and Bayesian networks [12], generate models that tend to ignore the minority class (defective modules usually) [19]. For instance, for a given dataset with 0.5% defective components, a classification accuracy of 99.5% can be achieved by simply classifying all components as non-defective. However, the AUC measure of the receiver operating characteristics (ROC) curve is 0.5 indicating that the classifier is simply tossing the coin to classify the dataset.

2.2. Sampling techniques

Oversampling and under sampling are two well-known standard techniques commonly used to deal with imbalanced datasets where the majority class is highly over-represented compared to the minority class [21]. The former technique adds data duplicates or synthetic samples to the minority class and data samples are removed from the majority class in the latter sampling technique. Moreover, results reported by Seiffert et al. [23] clearly indicate that the use of data sampling techniques improves the classification performance in the case of software defect prediction applications. However, it was reported that ensemble learning models, combined with boosting, always outperform data sampling-based defect classifiers in terms of classification accuracy [23].

In their experimental evaluation, Seiffert et al. [23] assessed the performance of 50,000 classification models using 15 datasets with five different sampling algorithms. While the best data sampling technique yielded an AUC of 0.744, boosting-based ensemble learning models achieved an AUC measure of 0.798. Pelayo and Dick [24] investigated the use of two data stratification approaches: (1) under-sampling and (2) over-sampling methods for software defect prediction. Data pertaining to the minority class was over-sampled to generate synthetic samples based on the synthetic minority oversampling technique (SMOTE) method [25]. Fewer samples were selected from the majority class by random under-sampling. Their approach resulted in approximately 23% mean classification accuracy improvement. It is noteworthy that over-sampling and under-sampling may lead to over-fitting and removal of relevant samples, respectively.

2.3. Cost-sensitive methods

Although sampling techniques tend to balance the data distribution properly, misclassifying different software defect classes might have aggravated costs [9]. Errors occurring in data classification can be cast into two types, namely, "Type-I" and "Type-II" [26]. The former quantifies the misclassification rates of defect-free software components and the latter is concerned with the misclassification of defective ones. Needless to mention that Type-II errors are more costly and, therefore, should be carefully looked at. This consideration stems from the devastating costs that could result from accepting a defective software component as defect-free. Given these facts, Khoshgoftaar et al. [26] proposed a cost-sensitive boosting technique that combines boosting ensemble learning algorithm and cost-sensitivity feature. Cost-sensitivity allows for the incorporation of a cost matrix that measures the penalty that is incurred by misclassifying data samples. In addition, the cost-sensitivity is reflected on the weight update of the classifier, which takes place more aggressively when Type-II errors are detected. In this way, higher penalty costs are assigned to the misclassification of defective software components.

Using the C4.5 decision tree as a base classifier, Quinlan improved the classification performance compared to the original boosting technique [27]. Another study is attributed to Zheng [28] where three types of cost-sensitive prediction models were compared. In all three models, boosted ANN techniques were used consisting of 10 basic back-propagation ANN blocks with 11 hidden neurons each. Unlike fixed weight update schemes, the smallest expected cost of misclassification (ECM) was achieved using the