Software Defect Prediction Using Hybrid Distribution Base Balance Instance Selection and Radial Basis Function Classifier

Mrutyunjaya Panda, Utkal University, Odisha, India

ABSTRACT

Software is an important part of human life and with the rapid development of software engineering the demands for software to be reliable with low defects is increasingly pressing. The building of a software defect prediction model is proposed in this article by using various software metrics with publicly available historical software defect datasets collected from several projects. Such a prediction model can enable the software engineers to take proactive actions in enhancing software quality from the early stages of the software development cycle. This article introduces a hybrid classification method (DBBRBF) by combining distribution base balance (DBB) based instance selection and radial basis function (RBF) neural network classifier to obtain the best prediction compared to the existing research. The experimental results with post-hoc statistical significance tests shows the effectiveness of the proposed approach.

KEYWORDS

Accuracy, distribution base balance, Kruskal-Wallis test, Mann-Whitney test, radial basis function, Software defect, Win-draw-loss

1. INTRODUCTION

Software defect is envisioned as an issue, the presence of which makes the software product to perform abnormally. At present, software defect prediction is thought of as an application area in big data analysis, as it could be possible to collect large amounts of unlabeled software defect metric data at low cost. However, the challenge lies on how to exploit the unlabeled data to predict software defect attracts many researchers during the past few years.

Knowing that Software defect may cause serious consequences in terms of huge financial and human losses in today’s software-intensive system, early detection of defect-prone modules before the release of any new software attracts lots of attention (Ryu et al., 2015; Kamei, and Shihab, 2016). In spite of a lot of research to obtain the quality software is ON, still, the poor performance issues with software reliability become a major concern for the researchers, due to the inherent problems in any one or all of the following: (1) no clear understanding in the requirement. For example: the poor understanding of which subset of attributes are generally responsible for the cause of the software defect, poses difficulty in choosing the right measures for the analysis (Emam et al., 2001; Sandhu et al., 2010) (2) coding errors (Menzies et al., 2007), (3) economics of software defect prediction
that may cost heavily when a non-defect module is classified as defect ones (Jiang et al., 2008), (4) insufficient software testing before release (Keiller and Miller, 1991), (5) class imbalance problem where the presence of one class is dominant over the other in the dataset (Batista et al., 2004) and (6) issues with cross-project defect prediction, where the model prediction is done in one project while examining it is done with a different company’s project (Turhan et al., 2009; Watanabe et al., 2008). It is also envisioned that the software defect prediction shall be carried out in entirety rather than investigating each individual component in isolation and further, the design choice shall be made judiciously in order to avoid the loss of generality and/or to avoid the useless results.

There are two ways through which software defect prediction can be modelled: static model and dynamic model. The static model predicts the number of software defect instances based on various characteristics and metrics of the software product in a subsequent software project. In contrast, dynamic model tries to predict the future defect prone software modules looking into the present and past defects in the subsequent time interval.

Since there is scarcity in getting quality data (e.g. which are usually not only noisy but also suffers from class imbalance problem), data pre-processing followed by the machine learning application are being proposed by many researchers to have better software defect prediction model (Rathore and Kumar, 2017; Khoshoftaar et al., 2010). Researchers have also opined for a good machine learning model for the classification of defective and non-defective software modules with a consideration that the cost involved in mis-classification of defective ones as non-defective is more than the other and also takes more time for testing the classifier (Khemchandani and Chandra, 2007; Tian et al., 2014; Tomar and Agarwal, 2015).

1.1. Motivation

As a member of the software testing team, proper planning is sought to meet the deadlines for the release of the new software for the end users. In the process of execution of the software testing, the testing team is supposed to ensure that all the software defects are found and fixed by the software developer in the early phase of system testing. After all these processes are over, then only it is decided whether to release the software or not. Hence, it is important to find some ways and means to predict the number of defects in the software at an early stage of system testing, to address this issue.

While solving the software defect prediction problem, the incorporation of both labelled and unlabeled data in the machine learning process leads to the best possible classification results. To this end, many researchers have used Graph-based learning with application of sparse theory on the dataset for pairwise relationship (Li and Fu, 2013); collaborative representation by the authors (Jing et al., 2014); metrics-based (Khoshoftaar and Gao, 2010); class imbalance (Wang and Yao, 2013); Dictionary learning (Wan et al., 2017), traditional methods like: Support Vector Machine (SVM) in (Elish and Elish, 2008), Naïve Bayesian (NB) (Wang and Li, 2010), Neural Network (Zheng, 2010) and the list goes on. It is observed that performance of the traditional methods severely limited with respect to lack of common feature representation and selection of a good feature selection algorithm in order to deal with sparse nature of the software prediction dataset.

It is also observed that the bulk of software defect prediction experiments carried out using the NASA Metrics Data Program data sets may have led to erroneous findings. This is mainly due to repeated data points potentially causing substantial amounts of training and testing data to be identical in these datasets (Gray et al., 2017). Thus, it is suggested to use feature selection techniques, to deal with the above issue, so that one can build a good software defect prediction model with less time & space complexity and better prediction accuracy.

Finally, some researchers opined to perform double pre-processing of the dataset by applying instance filtering along with attribute selection as a part of the future scope of work (Kakkar and Jain, 2016).
Designing for Changing Work and Business Practices
www.igi-global.com/chapter/designing-changing-work-business-practices/4218?camid=4v1a