ARTICLE IN PRESS

Journal of King Saud University – Computer and Information Sciences (2016) **xxx**, xxx–xxx

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

An empirical evaluation of classification algorithms for fault prediction in open source projects

Arvinder Kaur, Inderpreet Kaur *

Dept: CSE/IT, University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University, Dwarka, New Delhi 110078, India

Received 23 September 2015; revised 17 March 2016; accepted 8 April 2016

KEYWORDS

Metrics;
Fault prediction;
Receiver Operating Characteristics Analysis;
Machine learning;
Nimenyi test

Abstract Creating software with high quality has become difficult these days with the fact that size and complexity of the developed software is high. Predicting the quality of software in early phases helps to reduce testing resources. Various statistical and machine learning techniques are used for prediction of the quality of the software. In this paper, six machine learning models have been used for software quality prediction on five open source software. Varieties of metrics have been evaluated for the software including C & K, Henderson & Sellers, McCabe etc. Results show that Random Forest and Bagging produce good results while Naïve Bayes is least preferable for prediction.

© 2016 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Building high quality software with limited quality assurance budgets has become difficult. Various Software prediction models are used these days to learn fault predictors from software metrics. Software fault prediction, prior to the release of software helps in verification and validation activity and allocate the limited resources to modules which are predicted to be fault prone. Early and accurate fault prediction is a better approach for reducing testing efforts. Study (Mahanti and Antony, 2005) shows that software companies spend 50–80 percent of their software development effort on testing.

If fault prone modules are known in advance, review, analysis and testing efforts can be concentrated on those modules. Early detection of fault prone modules in the software life cycle has become one of the important goals of fault prediction because, earlier the detection of fault, the cheaper it is to correct it. Boehm and Papaccio advised fixing the fault early in the life cycle can make it cheaper by a factor of 50–200 (Boehm and Papaccio, 1988). Reliability of delivered products can be ensured using software quality models.

Software quality estimation using various classifiers is performed where input is some metrics and output is quality attributes. Empirical Study of these classifiers, aids in judgment of the quality of software being developed.

Various techniques have been suggested to deal with defect prediction which include categorizing modules, represented by a set of software metrics or code attributes into fault prone and non-fault-prone by means of classification model derived from data as per the previously developed projects (Schneidewind,

* Corresponding author.
E-mail addresses: arvinderkaurtakkar@yahoo.com (A. Kaur), kaur.inderpreet19@gmail.com (I. Kaur).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2016.04.002
1319-1578 © 2016 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1992).This also includes code metrics (e.g., lines of code, complexity) (Li and Henry, 1993; Chidamber and Kemerer, 1994; Lorenz and Kidd, 1994; McCabe and Associates, 1994; Basili et al., 1996; Henderson-Sellers, 1996; Ohlsson and Alberg, 1996; Briand et al., 1999; El Emam et al., 2001a,b,c; Gyimothy et al., 2005; Aggarwal et al., 2006; Nagappan and Ball, 2005; Nagappan et al., 2006), Process metrics (e.g., number of changes, recent activity) (Hassan, 2009; Moser et al., 2008; Bernstien et al., 2007), or previous defects (Kim et al., 2007; Ostrand et al., 2005; Hassan and Holt, 2005). The decision is still out on the relative performance of these approaches. Most of them have been judged were compared to only few other approaches. In addition, a significant portion of the evaluations cannot be reproduced since the data used for their evaluation came from commercial systems and are not available for public consumption. In some cases, researchers concluded differently: For example, in the case of size metrics, Gyimothy et al. reported good results (2005), as opposed to the findings of Fenton and Ohlsson (2000).

Various types of classifiers have been applied to this task, including statistical procedures (Basili et al., 1996; Khoshgoftaar and Seliya, 2004), tree-based methods (Selby and Porter, 1988; Porter and Selby, 1990; Khoshgoftaar et al., 2000; Guo et al., 2004; Menzies et al., 2004), neural networks (Khoshgoftaar et al., 1995; Khoshgoftaar et al., 1997), and analogy-based approaches (Khoshgoftaar et al., 2000; El Emam et al., 2001a,b,c; Khoshgoftaar and Seliya, 2003). However, as noted in (Myrtveit and Stensrud, 1999; Shepperd and Kadoda, 2001; Myrtveit et al., 2005) results regarding the superiority of one method over another or the usefulness of metric-based classification in general are not always consistent across different studies. Therefore, "There is a need to develop more reliable research procedures before we can have confidence in the conclusion of comparative studies of software prediction models" (Myrtveit et al., 2005).

Various classification algorithms have been applied to a variety of data sets. Different experimental setup results limit the ability to understand classifier's pros and cons. A modeling technique is good if it is able to perform well on all or at least most of them. To simplify model comparison, appropriate and consistent performance measures should be considered.

Evaluating the performance of various approaches is subject to discussion. While some use Binary classification (i.e. predicting if a given entity is a buggy or not) others predict by prioritizing components with most defects. In this paper, the binary classification technique is used which has been evaluated on the basis of the ROC, lift chart and other statistical parameters.

The datasets used in this work are open source java projects: PMD, EMMA, Find Bugs, Trove and Dr Java. Open source projects are different from Industrial projects. Open source software, is preferred for research as results of these can be compared and repetition of validation can be performed. Open source projects foster more creativity and have fewer defects as defects are found and fixed rapidly. (Paulson et al., 2004).

In this paper, the six well-known classification algorithms have been used Classifiers selected are Random Forest, Naive Bayes, Bagging, J48, logistic regression and IB1. These six classifiers have been chosen for the current study as previous studies indicate that these classifiers provide better than average performance in software fault prediction (Menzies et al., 2007a,b; Jiang et al., 2008). Many studies have used insufficient performance metrics which do not show enough level of details for future comparison. Therefore, the objectives of this paper include:

1. To compare models for fault prediction on open source software on the basis of
   (i) Performance measures including accuracy, sensitivity, specificity, Precision, G-mean, F-measure, J_coefficient.
   (ii) Graphical methods which include ROC curve, Precision Recall curve, Cost curves and Lift charts.
   (iii) Non parametric Freidman test followed by the post hoc Nimenyi test.
2. To compare the results of open source software projects with industrial data sets.

The paper is organized as follows: Section 2 presents the overview of Related Work. Section 3 describes the Research methodology along with datasets and metrics used for the classifier selection. Section 4 represents the model evaluation techniques, which includes comparison of numerical performance indices, Graphical evaluation techniques and statistical test performed. Section 5 consists of the analysis performed on each of the data sets taken in this work. Section 6 discusses the guidelines for selecting the best model followed by conclusions in section 7.

2. Related work

Basili et al. (1996) found that several of the (Chidamber and Kemerer, 1994) metrics were associated with fault proneness based on a study of eight medium-sized systems, developed by students.

Tang et al. (1999) analysed (Chidamber and Kemerer, 1994) OO metrics suite on three industrial applications developed in C++. They found none of the metrics examined to be significant except RFC and WMC.

Briand et al. (2000) have extracted 49 metrics to identify a suitable model for predicting fault proneness of classes. The system under investigation was a medium sized C++ software system developed by undergraduate/graduate students. There were eight systems under study, consisting a total of 180 classes. They used univariate and multivariate analyses to find the individual and combined impact of OO metrics and fault proneness. The results showed all metrics except NOC (which was found related to fault proneness in an inverse manner) to be significant predictors of fault proneness.

El Emam et al., 2001a,b,c examined a large telecommunication application developed in C++ and found that class size i.e. SLOC has a confused effect of most OO metrics on faults.

Another study by Briand and Wust (2001) used a commercial system consisting of 83 classes. They found the DIT metric related to fault proneness in an inverse manner and NOC metric to be an insignificant predictor of fault proneness.

Yu et al. (2002) choose eight metrics and they examined the relationship between these metrics and the fault proneness. The subject system was the client side of a large network service management system developed by three professional software engineers. It was written in Java, and consisted of

Download English Version:

https://daneshyari.com/en/article/6899071

Download Persian Version:

https://daneshyari.com/article/6899071

Daneshyari.com