Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

An empirical framework for defect prediction using machine learning techniques with Android software

Q1 Ruchika Malhotra

Department of Software Engineering, Delhi Technological University, Bawana Road, Delhi, India

ARTICLE INFO

Article history:
Received 30 November 2015
Received in revised form 20 March 2016
Accepted 26 April 2016
Available online xxx

Keywords:
Object-oriented metrics
Machine-learning
Software defect proneness
Statistical tests
Inter-release validation

ABSTRACT

Context: Software defect prediction is important for identification of defect-prone parts of a software. Defect prediction models can be developed using software metrics in combination with defect data for predicting defective classes. Various studies have been conducted to find the relationship between software metrics and defect proneness, but there are few studies that statistically determine the effectiveness of the results.

Objective: The main objectives of the study are (i) comparison of the machine-learning techniques using data sets obtained from popular open source software (ii) use of appropriate performance measures for measuring the performance of defect prediction models (iii) use of statistical tests for effective comparison of machine-learning techniques and (iv) validation of models over different releases of data sets.

Method: In this study we use object-oriented metrics for predicting defective classes using 18 machine-learning techniques. The proposed framework has been applied to seven application packages of well known, widely used Android operating system viz. Contact, MMS, Bluetooth, Email, Calendar, Gallery2 and Telephony. The results are validated using 10-fold and inter-release validation methods. The reliability and significance of the results are evaluated using statistical test and post-hoc analysis.

Results: The results show that the area under the curve measure for Naïve Bayes, LogitBoost and Multilayer Perceptron is above 0.7 in most of the cases. The results also depict that the difference between the ML techniques is statistically significant. However, it is also proved that the Support Vector Machines based techniques such as Support Vector Machines and voted perceptron do not possess the predictive capability for predicting defects.

Conclusion: The results confirm the predictive capability of various ML techniques for developing defect prediction models. The results also confirm the superiority of one ML technique over the other ML techniques. Thus, the software engineers can use the results obtained from this study in the early phases of the software development for identifying defect-prone classes of given software.

1. Introduction

In software engineering, early detection of defective portions of the software can help the software developers and engineers in proper allocation of limited resources in testing and maintenance phases of the software development. The cost of correcting the defects increases exponentially if the defects are encountered later in the software development. The software defect prediction models can be used in the early phases of software development life cycle. Further, their use reduces the testing and maintenance time, cost and effort of the project and thus improves the quality of the software [1].

In order to increase the level of automation while developing software, models are effective and are gaining importance [2]. Moreover, defect prediction models can be developed by using software metrics in conjunction with defect data obtained from historical repositories. The models can be trained using the historical releases of the same software and validated either on the same release or the subsequent releases of the software.

There are several machine-learning (ML) techniques proposed in the literature including neural networks, Support Vector Machines, ensemble learners and decision trees. But, it is difficult to establish the superiority of one ML technique over the other techniques using multiple data sets. Hence, more and more stud-

E-mail address: ruchikamalhotra2004@yahoo.com

http://dx.doi.org/10.1016/j.asoc.2016.04.032
1568-4946/

ies should be performed in order to draw well-formed, widely acceptable and generalized conclusions based on the experimental evidence gathered from the obtained results [1]. The results from the empirical studies will help to improve, refute and validate the results obtained from the past studies.

An effective empirical framework for the development of the prediction models should focus on the following issues: (i) use of appropriate and large number data sets (ii) performance of the predicted models assessed using appropriate performance measures (iii) reliability of the results using statistical tests (iv) validating the predicted models on data different from which they are trained. Lessman et al. [3] and Malhotra [4,5] observed that, the size of the study, performance measures used to assess the predicted model performance and the statistical tests to confirm the reliability of the results are three very important factors that need to be considered while conducting an empirical study. Also, the data sets available in software engineering research are scarce. There are only few studies that use statistical tests to analyze the suitability and validity of the results in literature. The evaluation of effectiveness of the performance of the predicted models is very crucial for the assessment of practical application of any defect prediction models. The reliability of empirical experiments can only be confirmed using the statistical tests [6]. Certain studies have pointed out that the statistical significance of the obtained results is rarely examined (Menzies et al. [7], Myrtviet et al. [8]). Further, previous studies have validated the developed models using the same data, on which they were trained.

In this work, we develop defect prediction models using object-oriented (OO) metrics over multiple application packages of Android operating system, open source software. Specifically, we address the following research issues in this work: (i) low repeatability of empirical studies, (ii) less usage of statistical tests for comparing the effectiveness of different models, and (iii) non-assessment of results on different releases of the software. This study will present an empirical framework of defect prediction models using 18 ML techniques, which will yield unbiased, accurate and repeatable results. The outcome of this research is assessed over various releases of seven application packages of Android software available in the Google code repository—Contact, MMS, Bluetooth, Email, Calendar, Gallery2 and Telephony.

As there is less use of statistical tests in the literature for statistically determining the comparative difference between predictive performances of developed models. Hence, after the models are generated, we will apply statistical techniques (such as Friedman test) to statistically determine whether there is a statistical difference between the performances of different ML techniques. We will also perform post-hoc analysis (using Nemenyi test) to evaluate the pairwise comparison amongst the results of different techniques. The results are evaluated using area under the curve (AUC) obtained from Receiver Operating Characteristics (ROC) analysis. Thus, the following research questions are addressed in this work:

- RQ1: What is the overall predictive capability of various ML techniques on seven application packages of Android software using 10-fold validation?

In this question we validate the results of predicted models using 10-fold validation with the help of various performance measures. The overall capability of the 18 ML techniques is assessed based on the results obtained using seven application packages of Android software.

- RQ2: What is the performance of defect prediction models when inter-release validation is carried out?

The performance of defect prediction models is validated using inter-release validation in this question. The results are evaluated using the AUC measure obtained using ROC analysis.

- RQ3: Is the performance of defect prediction models validated using inter-release validation comparable to 10-fold validation and is it statistically different than 10-fold validation?

We compare and assess the performance of models validated using inter-release with models validated using 10-fold validation. We determine the statistical difference between the results of inter-release validation and 10-fold validation for defect prediction using Wilcoxon test.

- RQ4: Which are the best and worst ML techniques for defect prediction using OO metrics?

The best and worst ML techniques are determined using the results of both 10-fold and inter-release validation over the seven application packages of Android software. These results are based on AUC measure and derived using the statistical test, Friedman.

- RQ5: Which pairs of ML techniques are statistically different from each other for defect prediction?

In this research question, we determine the pairs of ML techniques that are statistically different than each other. The results are based on post-hoc analysis using Nemenyi test.

The initial results carried out using one application package of Android software following the proposed approach are reported in Malhotra et al. [9]. Now, we present a major extension of the preliminary results presented in our previous study by evaluating the ML techniques over six additional application packages of Android software. The results of the previous study were not generalizable as they were only based on one application package of Android software. Also, we carry out post-hoc analysis using Nemenyi test to determine the effectiveness of the results. We also determine the statistical significance of inter-release validation and compare the ML techniques on the basis of inter-release validation. There is no study to the best of the authors' knowledge that extensively compares and assesses the performance of ML techniques to analyze the relationship between OO metrics and defect prediction using statistical tests. Hence, the main contributions of this paper are summarized below:

(1) An extensive comparison of 18 popular ML techniques in the context of defect prediction.
(2) The use of data collected from seven application packages over multiple releases of widely used Android software.
(3) Statistical analysis of the obtained results for comparison of ML techniques.
(4) An inter-release validation of models developed in order to obtain unbiased and generalized results.

The rest of the paper is organized as follows: Section 2 summarizes the related work and Section 3 describes the empirical research framework followed in this paper. Section 4 presents the research methodology and Section 5 provides the answers to the research questions. The threats to validity in the current research are summarized in Section 6 and the conclusions of the work are presented in Section 7.

Please cite this article in press as: R. Malhotra, An empirical framework for defect prediction using machine learning techniques with Android software, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.04.032