Information and Software Technology 55 (2013) 1810–1822

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

A study of subgroup discovery approaches for defect prediction

Daniel Rodriguez [a,*], Roberto Ruiz [b], Jose C. Riquelme [c], Rachel Harrison [d]

[a] Department of Computer Science, University of Alcalá, Ctra. Barcelona, Km. 33.6, 28871 Alcalá de Henares, Madrid, Spain
[b] School of Engineering, Pablo de Olavide University, Ctra. Utrera, Km. 1, 41013 Seville, Spain
[c] Department of Computer Science, University of Seville, Avda. Reina Mercedes s/n, 41012 Seville, Spain
[d] School of Technology, Oxford Brookes University, Wheatley Campus, Oxford OX33 1HX, UK

ARTICLE INFO

Article history:
Received 26 September 2012
Received in revised form 5 May 2013
Accepted 5 May 2013
Available online 16 May 2013

Keywords:
Subgroup discovery
Rules
Defect prediction
Imbalanced datasets

ABSTRACT

Context: Although many papers have been published on software defect prediction techniques, machine learning approaches have yet to be fully explored.
Objective: In this paper we suggest using a descriptive approach for defect prediction rather than the precise classification techniques that are usually adopted. This allows us to characterise defective modules with simple rules that can easily be applied by practitioners and deliver a practical (or engineering) approach rather than a highly accurate result.
Method: We describe two well-known subgroup discovery algorithms, the SD algorithm and the CN2-SD algorithm to obtain rules that identify defect prone modules. The empirical work is performed with publicly available datasets from the Promise repository and object-oriented metrics from an Eclipse repository related to defect prediction. Subgroup discovery algorithms mitigate against characteristics of datasets that hinder the applicability of classification algorithms and so remove the need for preprocessing techniques.
Results: The results show that the generated rules can be used to guide testing effort in order to improve the quality of software development projects. Such rules can indicate metrics, their threshold values and relationships between metrics of defective modules.
Conclusions: The induced rules are simple to use and easy to understand as they provide a description rather than a complete classification of the whole dataset. Thus this paper represents an *engineering approach* to defect prediction, i.e., an approach which is useful in practice, easily understandable and can be applied by practitioners.

1. Introduction

In the recent past, the application of data mining techniques in software engineering has received a lot of attention. Problems such as planning and decision making, defect prediction, effort estimation, testing and test case generation, knowledge extraction, etc. can be reformulated using a set of techniques under the umbrella of data mining [15,73,27]. The extracted patterns of knowledge can assist software engineers in predicting, planning, and understanding various aspects of a project so that they can more efficiently support future development and project management activities.

Data mining provides techniques to analyse and extract novel, interesting patterns from data. Formally, it has been defined as the process of inducing previously unknown and potentially useful information from data collections [23]: "*The two high-level primary goals of data mining in practice tend to be prediction and description*". The former is related to the prediction of unknown or future values (e.g. classification tree models, regression models, etc.), the latter, involves finding interesting patterns that can be easily understood by humans (e.g. association and clustering algorithms). It is worth noting that Fayyad et al. also state that: "*the boundaries between prediction and description are not sharp (some of the predictive models can be descriptive, to the degree that they are understandable, and vice versa)*". For example, clustering algorithms can be used as classifiers or supervised feature selection methods can be considered descriptive. There are also techniques that hybridise prediction and description as in the case of *supervised descriptive techniques* [43]. The aim of these techniques is to understand the underlying phenomena rather than to classify new instances; i.e., to find interesting information about a specific value. The information should be useful to the domain experts and easily interpretable by them.

In this work, we tackle the defect prediction problem through a descriptive induction process using Subgroup Discovery (SD) tech-

* Corresponding author. Tel.: +34 918856933; fax: +34 918856646.
E-mail addresses: daniel.rodriguezg@uah.es (D. Rodriguez), robertoruiz@upo.es (R. Ruiz), riquelme@lsi.us.es (J.C. Riquelme), rachel.harrison@brookes.ac.uk (R. Harrison).

0950-5849/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2013.05.002

niques. These kinds of algorithms are designed to find subgroups of data that are statistically different given a property of interest [39,71,72,32]. SD algorithms can be both predictive, finding rules given historical data and a property of interest; and descriptive, discovering interesting patterns in data. For the same purpose, there are other types of supervised descriptive techniques, Contrast Set Mining (CSM) [6] and Emerging Pattern Mining (EPM) [19]. CSM finds contrast sets which are defined as conjunctions of attribute-value pairs that differ significantly in their distributions across groups (class variable). These contrast sets may have a very low support but they must clearly differentiate the different groups. EPM captures emerging patterns (EPs) in time-stamped databases or useful contrasts in classification datasets (with a class attribute). EPs are defined as itemsets (using association rule terminology) whose support increases significantly from one dataset, D_1, to another dataset, D_2. EPM searches for characteristics that differentiate two itemsets, D_1 and D_2, based on the *growthRate*[1] (ratio between both supports) as quality measure. These techniques have similarities: they all use rules as representation techniques and have been proved to be equivalent [43]. However, the SD approaches have better tool support (including the Orange toolkit) and the quality measures used as objective function focus on finding statistically different subgroups (this is explained in Section 3).

The main contributions of this paper are as follows. Firstly, to propose a descriptive approach based on subgroup discovery for defect prediction which allows us to characterise defective modules with simple rules that can easily be applied by practitioners. Such rules can describe thresholds and relationships between metrics. In this paper, we show how SD algorithms induce rules that can indicate defective software modules with a fairly high probability. To do this, we rely on the fact that SD algorithms mitigate against some of the characteristics of datasets that hinder the applicability of many classical classification algorithms such as (i) imbalanced datasets in which the number of non-defective modules is much larger than the number of defective modules, (ii) duplicated instances and contradictory cases and (iii) redundant and irrelevant attributes. In the literature these problems have mainly been tackled with preprocessing techniques such as sampling and feature selection. SD algorithms can be an alternative to classical classification algorithms without the need to apply preprocessing techniques. Modifying the original data or using preprocessing techniques does not always guarantee better results and can make it more difficult to extract knowledge from the data. SD algorithms, on the other hand, can be applied to the original data without the need for sampling or feature selection techniques and the representation of the rules makes them easy to apply.

In summary, we search for simple models represented as rules capable of detecting defective modules rather than highly accurate models. Thus our research question is: *can subgroup discovery be used to detect the most defective modules in a system?*

We describe and compare two well-known SD algorithms, the Subgroup Discovery (SD) algorithm [25] and the CN2-SD algorithm [44], by applying them to several datasets from the publicly available Promise repository [52], as well as the Bug Prediction Dataset (BPD) created by D'Ambros et al. [16,17].

The organization of the paper is as follows. Section 2 covers the related work in defect prediction followed by background related to subgroup discovery concepts in Section 3. Next, Section 4 describes the experimental work, including datasets, rule induction, study of the generalisation of the rule induced and discussion of the results. Section 5 covers the threats to the validity. Finally, Section 6 concludes the paper and outlines future research work.

[1] $growthRate(itemset) = \frac{support_{D_2}(itemset)}{support_{D_1}(itemset)}$.

2. Related work

Defect prediction has been an important research topic for more than a decade with an increasing number of papers including two recent and comprehensive systematic literature reviews [12,29]. Many studies in defect prediction have been reported using techniques which originated from the field of statistics and machine learning. Such techniques include regression [8], logistic regression [18,75], Support Vector Machines [20], etc. Others have their origin in machine learning techniques such as classification trees [34], neural networks [35], probabilistic techniques (such as Naïve Bayes [53] and Bayesian networks [24]), Case Based Reasoning [36], ensembles of different techniques and meta-heuristic techniques such as ant colony optimisation [33,5,67].

Work has also been done on using rules as a representation model or decision trees such as C4.5 [59] which can be easily transformed to rules. For example, Koru and Liu [41] used C4.5 for defect prediction with the NASA datasets to analyse the relationships between defects and module size. Also descriptive rules such as association rules [1] have been applied by Song et al. [66] to predict the defect associations and defect correction effort. In general, rules are easier to understand and apply than many other classification techniques such as neural networks or ensembles of multiple classifiers which behave as black boxes and are difficult to generalise across different datasets even when the same attributes are used. Hierarchical rules, such as chained *if ... then ... else* rules are harder to interpret and use by domain experts than independent rules such as the ones obtained by SD approaches. For example, Vandecruys et al. [67] reported the use of ant colonies as optimisation technique for generating rules. A drawback of their approach is that they cannot handle imbalanced datasets appropriately and hierarchical rules can become hard to understand and apply. Azar and Vybihal [5] have also used metaheuristic optimisation to induce rules capable of predicting defective modules from a number of static metrics that measure size, cohesion, coupling and inheritance. In this case, the authors recognise and deal with the imbalance by reporting Younden's J_{index} per class. In other work, Azar et al. [4] also combine rules from different algorithms using metaheuristics. Their approach could be used to combine and select rules induced from different SD algorithms (as shown in this work) or as a postprocessing step if a large number of rules are generated.

Several papers have compared multiple techniques with single datasets (e.g. [37]) or multiple datasets with multiple evaluation measures. Peng et al. [57] evaluated 13 classification algorithms with 11 measures over 11 software defect datasets. Although Support Vector Machines, nearest neighbour and the C4.5 algorithm were ranked as the top three classifiers, the authors indicated that a classifier which obtains the best result for a given dataset according to a given measure may perform poorly with a different measure. Also in another work, Peng et al. [58] used 10 NASA datasets to rank classification algorithms, showing that a CART boosting algorithm and the C4.5 decision-tree algorithm with boosting are ranked as the optimum algorithms for defect prediction. Another extensive study, Lessmann et al. [45] compared 22 classifiers grouped into statistical, nearest neighbour, neural networks, support vector machine, decision trees and ensemble methods over 10 datasets from the NASA repository. The authors discuss several performance metrics such as TP_{rate} and FP_{rate} but advocate the use of Area Under the ROC (AUC) [21] as the best indicator for comparing the different classifiers.

However, there are discrepancies among the outcomes of these works where (i) no classifier is consistently better than the others; (ii) there is no optimum metric to evaluate and compare classifiers as highlighted in [49,56,74,53]; and (iii) there are quality issues regarding the data such as imbalanced datasets, class overlaps,