International Journal of Applied Metaheuristic Computing
Volume 10 • Issue 1 • January-March 2019

Cross-Project Change Prediction Using Meta-Heuristic Techniques

Ankita Bansal, Netaji Subhas Institute of Technology, Delhi, India
Sourabh Jajoria, Netaji Subhas Institute of Technology, Delhi, India

ABSTRACT

Changes in software systems are inevitable. Identification of change-prone modules can help developers to focus efforts and resources on them. In this article, the authors conduct various intra-project and cross-project change predictions. The authors use distributional characteristics of dataset to generate rules which can be used for successful change prediction. The authors analyze the effectiveness of meta-heuristic decision trees in generating rules for successful cross-project change prediction. The employed meta-heuristic algorithms are hybrid decision tree genetic algorithms and oblique decision trees with evolutionary learning. The authors compare the performance of these meta-heuristic algorithms with C4.5 decision tree model. The authors observe that the accuracy of C4.5 decision tree is 73.33%, whereas the accuracy of the hybrid decision tree genetic algorithm and oblique decision tree are 75.00% and 75.56%, respectively. These values indicate that distributional characteristics are helpful in identifying suitable training set for cross-project change prediction.

KEYWORDS

Change-Prone, Cross-Validation, Decision Tree, Distribution Characteristics, Ensemble Learners, Evolutionary Algorithms, Open Source, Receiver Operating Characteristics

1. INTRODUCTION

Change-proneness is the likelihood that a particular class will be changed in future versions of software (Tsantalis, 2005). Due to the increasing size, complexity, changing demands of customer etc., the changes in software are unavoidable. Incorporating these changes is essential but at the same time, requires huge amount of resources. Additionally, changes in one class may also further lead to changes in the other classes. The largest percentage of the software development effort is spent on rework and maintenance (Brooks, 1974). Due the availability of limited resources, it is useful to identify some classes which may be more prone to changes in future as compared to the other classes. During the initial phases of software development life cycle, we identify the change prone classes with the help of software metrics. Identification of such classes will allow the developers and testers to pay focused attention on them, thus, leading to judicious allocation of limited resources in terms of time, money and manpower.

Building a prediction model requires training data. In case of change prediction, the training data is the large amount of historical data of the project. Intra-project change prediction models use historical data of the same project to predict change-prone classes. But sometimes, the historical data might not be available like in the case of first version of software. Cross-project change prediction refers to predicting change-prone classes from training data of other projects. It is important to find

DOI: 10.4018/IJAMC.2019010103

suitable training data for predicting change-prone classes. Once the training data is identified, the models can be constructed and those models can be used to predict change prone classes of some other data/project. This has motivated the authors to conduct intra-project and cross-project prediction, followed by generating rules for successful selection of training data. Although work has been done in cross-project defect prediction, but very less work has been done on cross-project change prediction.

In this work, the authors have performed intra-project and cross-project change prediction using ensemble learners, viz. Adaboost, Bagging, Logitboost and Random Forest. For selecting suitable training data, the authors have used the similar methodology as that used by He (2012). He (2012) used distributional characteristics of training dataset and testing dataset to create rules for cross-project defect prediction. For generating rules, the authors have constructed models using evolutionary algorithms.

Evolutionary algorithms are a type of meta-heuristic optimization algorithms that search space of candidate solutions. Evolutionary algorithms need very little domain specific knowledge. Such algorithms are inspired by the principle of evolution and natural selection. They make use of a quality function to create a set of candidate solutions. The quality or fitness function represents a heuristic estimate of solution quality. A population of these candidate solutions is chosen to seed the next generation by applying various operators like reproduction, recombination and mutation. These operators optimize the fitness of solution over generations until a fixed number of iterations or until a good enough solution has been found (Harman, 2010). They are best suited to be applied on the project with varying characteristics as they are robust, flexible and can handle imbalanced and noisy data. Despite the stated advantages of evolutionary algorithms, they have not been applied in the domain of cross-project change prediction to the best of authors' knowledge. Thus, the authors have used evolutionary algorithms for generating rules for successful cross-project change prediction because of the following reasons: 1) In cross-project change prediction, the prediction model created from a project are applied to different projects having different characteristics. 2) The data gathered for the experiments from the open source software repositories contains noisy and imbalanced data. 3) Due to the self-optimizing nature of evolutionary algorithms, they may produce good classification rules for cross-project change prediction.

In this work, the authors have used decision tree techniques to generate rules. The decision tree techniques are used as the reproduced or discovered knowledge needs to be expressed in the form of rules. This way of representing the knowledge is easily comprehensible by the readers and thus, is preferred over other knowledge representation methods. The rules can be easily read and analyzed in a tree-structured model. In addition to this, the construction of decision tree classifiers does not require any domain knowledge, thus it is appropriate for exploratory knowledge discovery. Once the rules (If-then type) are formed, then suitable training data can be selected to create a model that predicts the change prone classes of the unseen instances of some other data.

The evolutionary tree based algorithms the authors have used are Hybrid decision tree genetic algorithm (Carvalho, 2004) and Oblique decision tree with evolutionary learning (Cantu-Paz, 2003). In addition to evolutionary algorithms, authors have also used a Machine Learning (ML) decision tree technique, C4.5 which was used by He (2012) for generating rules in defect prediction studies. Thus, this will allow us to explore and compare the performance of traditional decision tree technique with evolutionary decision tree techniques.

The research questions addressed in this paper are:

1. Does intra-project change prediction work better than cross-project change prediction?
2. Can distributional characteristics be used for selecting suitable training set for predicting change-prone classes?
3. Can evolutionary decision tree algorithms give comparable or better results than traditional decision tree for selecting suitable training set?