Chapter I
Software Quality Modeling with Limited Apriori Defect Data

Naeem Seliya
University of Michigan, USA

Taghi M. Khoshgoftaar
Florida Atlantic University, USA

ABSTRACT

In machine learning the problem of limited data for supervised learning is a challenging problem with practical applications. We address a similar problem in the context of software quality modeling. Knowledge-based software engineering includes the use of quantitative software quality estimation models. Such models are trained using apriori software quality knowledge in the form of software metrics and defect data of previously developed software projects. However, various practical issues limit the availability of defect data for all modules in the training data. We present two solutions to the problem of software quality modeling when a limited number of training modules have known defect data. The proposed solutions are a semisupervised clustering with expert input scheme and a semisupervised classification approach with the expectation-maximization algorithm. Software measurement datasets obtained from multiple NASA software projects are used in our empirical investigation. The software quality knowledge learnt during the semisupervised learning processes provided good generalization performances for multiple test datasets. In addition, both solutions provided better predictions compared to a supervised learner trained on the initial labeled dataset.

INTRODUCTION

Data mining and machine learning have numerous practical applications across several domains, especially for classification and prediction problems. This chapter involves a data mining and machine learning problem in the context of software quality modeling and estimation. Software measurements and software fault (defect) data have been used in the development of models that predict

software quality, for example, a software quality classification model (Imam, Benlarbi, Goel, & Rai, 2001; Khoshgoftaar & Seliya, 2004; Ohlsson & Runeson, 2002) predicts the fault-proneness membership of program modules. A software quality model allows the software development team to track and detect potential software defects relatively early-on during development.

Software quality estimation models exploit the software engineering hypothesis that software measurements encapsulate the underlying quality of the software system. This assumption has been verified in numerous studies (Fenton & Pfleeger, 1997). A software quality model is typically built or trained using software measurement and defect data from a similar project or system release previously developed. The model is then applied to the currently under-development system to estimate the quality or presence of defects in its program modules. Subsequently, the limited resources allocated for software quality inspection and improvement can be targeted toward low-quality modules, achieving cost-effective resource utilization (Khoshgoftaar & Seliya, 2003).

An important assumption made during typical software quality classification modeling is that fault-proneness labels are available for all program modules (instances) of training data, that is, supervised learning is facilitated because all instances in the training data have been assigned a quality-based label such as fault-prone (*fp*) or not fault-prone (*nfp*). In software engineering practice, however, there are various practical scenarios that can limit availability of quality-based labels or defect data for all the modules in the training data, for example:

- The cost of running data collection tools may limit for which subsystems software quality data is collected.
- Only some project components in a distributed software system may collect software quality data, while others may not be equipped for collecting similar data.
- The software defect data collected for some program modules may be error-prone due to data collection and recording problems.
- In a multiple release software project, a given release may collect software quality data for only a portion of the modules, either due to limited funds or other practical issues.

In the training software measurement dataset the fault-proneness labels may only be known for some of the modules, that is, labeled instances, while for the remaining modules, that is, unlabeled instances, only software attributes are available. Under such a situation following the typical supervised learning approach to software quality modeling may be inappropriate. This is because a model trained using the small portion of labeled modules may not yield good software quality analysis, that is, the few labeled modules are not sufficient to adequately represent quality trends of the given system. Toward this problem, perhaps the solution lies in extracting the knowledge (in addition to the labeled instances) stored in the software metrics of the unlabeled modules.

The above described problem represents the labeled-unlabeled learning problem in data mining and machine learning (Seeger, 2001). We present two solutions to the problem of software quality modeling with limited prior fault-proneness defect data. The first solution is a semisupervised clustering with expert input scheme based on the *k-means* algorithm (Seliya, Khoshgoftaar, & Zhong, 2005), while the other solution is a semisupervised classification approach based on the expectation maximization (EM) algorithm (Seliya, Khoshgoftaar, & Zhong, 2004).

The semisupervised clustering with expert input approach is based on implementing constraint-based clustering, in which the constraint maintains a strict membership of modules to clusters that are already labeled as *nfp* or *fp*. At the end of a constraint-based clustering run a domain expert is allowed to label the unlabeled clusters, and the semisupervised clustering process is iter-