Software Protection Using Association Rule Mining

N.Ravi
M.Tech (CSE) student
Akula Sriramulu College of engg , Tanuku, AP, India

SVD. Venugopal
Assistant Professor
Akula Sriramulu College of engg, Tanuku, AP, India

M.Sateesh Kumar
H.O.D of CSE
Akula Sriramulu College of engg, Tanuku, AP,India

Abstract

Attribute selection is an important activity in data preprocessing for software quality modeling and other data mining problems. The software quality models have been used to improve the fault detection process. Finding faulty components in a software system during early stages of software development process can lead to a more reliable final product and can reduce development and maintenance costs. Ithas been shown in some studies that prediction accuracy of the models improves when irrelevant and redundant features are removed from the original data set. In this study, we investigated four filter attribute selection techniques, Automatic Hybrid Search (AHS), Rough Sets (RS), Kolmogorov-Smirnov (KS) and Probabilistic Search (PS) and conducted the experiments by using them on a very large telecommunications software system. In order to evaluate their classification performance on the smaller subsets of attributes selected using different approaches, we built several classification models using five different classifiers. The empirical results demonstrated that by applying an attribution selection approach we can build classification models with accuracy comparable to that built with a complete set of attributes.

Keywords: software quality, probabilistic search, attributes selection approach, and classification models.

INTRODUCTION

Software quality is an important attribute of software product especially for high-assurance and mission-critical systems. Predicting the quality of software modules in the early stages of software development process is very critical, so that software quality assurance efforts can be prioritized for targeting those modules that are either high-risk, or likely to have a high number of faults. Software quality models are the tools to implement such predictions. A software quality model can use software metrics that are collected prior to software testing and operations to estimate the quality factor of software modules such as number of faults or quality based classes, fault-prone and not fault-prone.

Over the last two decades, significant research has been dedicated towards developing methods for improving the predictive accuracy of software quality models. It has been shown in some studies that the performance of these models improves when irrelevant and redundant features are eliminated from the original data set. In addition, attribute selection can reduce the time for the metrics collection, model calibration, model validation, and model evaluation of future software development efforts of similar systems.

In this paper, we investigated four different attribute selection techniques, AHS, PS, KS and RS and applied them to a data set for a very large telecommunications software system. In order to evaluate their classification performance on the smaller subsets of attributes selected using various approaches, we built several classification models using five different classifiers. They are Instance-based learning (IBK), Multilayer perception (MLP), Support vector machine (SVM), Naive Bakes (NB), and Logistic Regression (LR). The experimental results demonstrate that the classification accuracy of the models built with some smaller subsets of attributes is comparable to that built with the complete set of attributes. Moreover, the smaller subsets of attributes include less than 15 percent of the complete set of attributes. In addition, among the four attribute selection

approaches, our recently proposed KS method performed better than the other three techniques in terms of two performance metrics (AUC and BGM) for four out of five learners.

II. METHODOLOGY

A. Attribute Selection Techniques

Attribute selection is a process of reducing data dimension. It is one of the frequently used techniques in data preprocessing for data mining. Attribute selection process consists of four basic steps [6]:

1) **Subset generation**. It produces candidate attribute subset based on a certain search strategy. In this step, two problems need to be solved.
2) First, where to start. According to the different strategies, we can divide them into two categories, *forward* (search starts with an empty set and inserts attributes subsequently) and *backward* (search starts with a full set and deletes attributes subsequently).

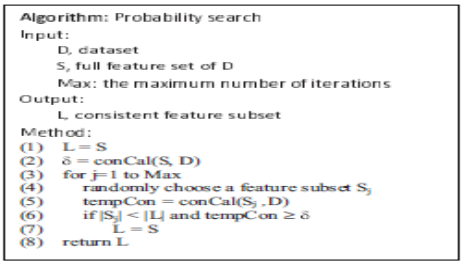

Fig. 1. PS Algorithm

The **Automatic Hybrid Search** (AHS), an attribute subset selection

Starting with size 1 of any attribute, attribute subsets that have the locally highest consistency rate are selected. These selected attribute subsets will be used to generate superset. Repeat the process until finding the attribute subsets that have the same consistency rate or the complete attribute set is reached. If more than one attribute subsets are generated, a classifier called C4.5 [10] will be used to decide which attribute subset is selected based on an error rate. C4.5 is an algorithm for inducing classification rules in the form of a decision tree from a given data set. For this case study, one attribute subset (with six attributes) was produced without using C4.5. The AHS algorithm is illustrated in Figure 2.

Fig. 2. AHS Algorithm

The **K-S Method** is an attribute selection method recently proposed by our research group [11]. It utilizes the Kolmogorov-Smirnov (KS) statistic to measure the maximum differences between the empirical distribution function of the posterior probabilities of instances in each class. The larger the distance between the distribution functions, the better the attribute is able to distinguish between the two classes. The attributes can be ranked based on their K-S scores and be selected according to their K-S scores and the number of attributes needed.

3 PROPOSED SOFTWARE DEFECT PREDICTION FRAMEWORK

3.1 Overview of the framework

Generally, before building defect prediction model(s) and using them for prediction purposes, we first need to decide which learning scheme should be used to construct the model. Thus the predictive performance of the learning scheme(s) should be determined, especially for future data. However, this step is often neglected and so the resultant prediction model may not be trustworthy. Consequently we propose a new software defect prediction framework that provides guidance to address these potential shortcomings. The framework consists of two components: (i) scheme evaluation and (ii) defect prediction. Fig. 1 contains the details.

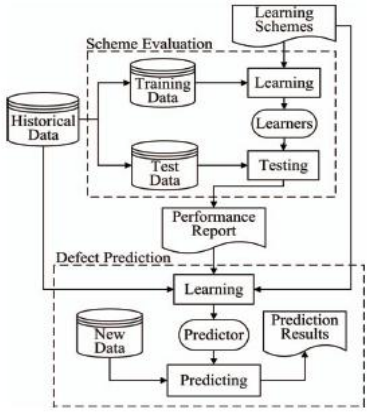

Fig. 1. Proposed software defect prediction framework

At the scheme evaluation stage, the performances of the different learning schemes are evaluated with historical data to

determine whether a certain learning scheme performs sufficiently well for prediction purposes or to select the best from a set of competing schemes.

This is very different from the first stage; it is very useful for improving the generalization ability of the predictor.

After the predictor is built, it can be used to predict the defect-proneness of new software components. MGF proposed a baseline experiment and reported the performance of the Naiıve Bayes data miner with log filtering as well as attribute selection, which performed the scheme evaluation but with inappropriate data. This is because they used both the training (which can be viewed as historical data) and test (which can be viewed as new data) data to rank attributes, while the labels of the new data are unavailable when choosing attributes in practice.

3.2 Scheme evaluation

After the training-test splitting is done each round, both the training data and learning scheme(s) are used to build a learner. A learning scheme consists of a data preprocessing method, an attribute selection method, and a learning algorithm. The detailed learner construction procedure is as follows:

1) Data preprocessing

This is an important part of building a practical learner. In this step, the training data are preprocessed, such as removing outliers, handling missing values, discrediting or transforming numeric attributes. In our experiment, we use a log-filtering preprocessor which replaces all numeric's n with their logarithms ln(n) such as used in MGF.

2) Attribute selection

The data sets may not have originally been intended for defect prediction, thus even if all the attributes are useful for its original task, not all may be helpful for defect prediction. Therefore attribute selection has to be performed on the training data.

Attribute selection methods can be categorized as either filters or wrappers [36]. It should be noted that both 'filter' and 'wrapper' methods only operate on the

training data. A 'filter' uses general characteristics of the data to evaluate attributes and operates independently of any learning algorithm. In contrast, a 'wrapper' method, exists as a wrapper around the learning algorithm searching for a good subset using the learning algorithm itself as part of the function evaluating attribute subsets .Wrappers generally give better results than filters but are more computationally intensive. In our proposed framework, the 'wrapper' attribute selection method is employed. To make most use of the data, we use a M×N-way cross-validation to evaluate the performance of different attribute subsets.

3) Learner construction

Once attribute selection is finished, the preprocessed training data are reduced to the best attribute subset. Then the reduced training data and the learning algorithm are used to build the learner.

IV. CONCLUSION

Attribute selection plays an important role in data preprocessing. By removing irrelevant and redundant features from a training data set, software quality estimation based on some classification models may improve.

In this paper, we present four different attribute selection techniques and their applications to a very large telecommunications software system. The classification accuracy was evaluated in terms of two performance metrics AUC and BGM. The experimental results demonstrate that the KS method is better than the other three techniques, PS, AHS and RS. Also, the classification model built on the smaller subset of attributes via the KS method has a comparable (no significant difference) performance to that built with a complete set of attributes.

REFERENCES

[1] T. M. Khoshgoftaar and N. Seliya, "Comparitive assessment of software quality classification technique," *Empirical Sofware Engineering Journal*, vol. 9, no. 3, pp. 229–257, 2004

[2] T. M. Khoshgoftaar, L. A. Bullard, and K. Gao, "Attribute selection using rough sets in software quality classification,"

International Journal of Reliability, Quality and Safty Engineering, vol. 16, no. 1, pp. 73–89, 2009.

[3] T. M. Khoshgoftaar, Y. Xiao, and K. Gao, "Assessment of a multistrategy classifier for an embedded software system," in *Proceedings of 18th IEEE International Conference on Tools with Artificial Intelligence*, Washing D. C., November 13-15 2006, pp. 651–658.

[4] D. Rodriguez, R.Ruiz, J. Cuadrado-Gallego, and J. Aguilar-Ruiz, "Detecting fault modules applying feature selection to classifiers," in *Proceedings of 8th IEEE International Conference on Information Reuse and Integration*, Las Vegas, IL, August 13-15 2007, pp. 667–672.

[5] M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437 – 1447, Nov/Dec 2003.

[6] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005

[7] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Boston, MA: Springer, 1998.

[8] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artificial Intelligence*, vol. 151, no. 1-2, pp. 151–176, 2003

[9] P. Lin, H. Wang, and T. M. Khoshgoftaar, "A novel hybrid search algorithm for feature selection," in *Proceedings of 21st International Conference on Software Engineering and Knowledge Engineering*, Boston, MA, July 1-3 2009, in press.

[10] J. R. Quinlan, *C4.5: programs for machine learning*. Los Altos, California: Morgan Kaufmann, 1993.