Information Sciences 181 (2011) 4867–4877

Contents lists available at ScienceDirect

Information Sciences

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

Class noise detection based on software metrics and ROC curves

Cagatay Catal *, Oral Alan, Kerime Balkan

The Scientific and Technological Research Council of Turkey (TUBITAK), Center of Research for Advanced Technologies of Informatics and Information Security, Information Technologies Institute, 41470 Gebze, Kocaeli, Turkey

ARTICLE INFO

Article history:
Received 17 October 2009
Received in revised form 2 June 2011
Accepted 15 June 2011
Available online 2 July 2011

Keywords:
Receiver Operating Characteristic (ROC) curve
Noise detection
Software metrics
Metric threshold values
Software fault prediction
Software quality

ABSTRACT

Noise detection for software measurement datasets is a topic of growing interest. The presence of class and attribute noise in software measurement datasets degrades the performance of machine learning-based classifiers, and the identification of these noisy modules improves the overall performance. In this study, we propose a noise detection algorithm based on software metrics threshold values. The threshold values are obtained from the Receiver Operating Characteristic (ROC) analysis. This paper focuses on case studies of five public NASA datasets and details the construction of Naive Bayes-based software fault prediction models both before and after applying the proposed noise detection algorithm. Experimental results show that this noise detection approach is very effective for detecting the class noise and that the performance of fault predictors using a Naive Bayes algorithm with a logNum filter improves if the class labels of identified noisy modules are corrected.

1. Introduction

Examples of data quality problems in real-world datasets include incomplete, noisy, and inconsistent data [7], and this low-quality data lowers the performance of machine learning-based models. Preprocessing in data mining, which is an important step in the data mining process, improves the data quality. Data cleaning, data integration, data transformation, and data reduction routines are the major tasks of data preprocessing. These terms are defined as follows:

- *Data cleaning* (*or data cleansing*): Missing values are filled in; outliers are identified; noisy data is removed; and inconsistent data is corrected.
- *Data integration:* Data from multiple databases or data cubes is combined into a data store.
- *Data transformation:* Data is transformed into appropriate forms using normalization or aggregation techniques.
- *Data reduction:* Data is reduced to a smaller volume, but the new dataset provides similar analytical results. Attribute subset selection, dimensionality reduction, numerosity reduction, and data cube aggregation are some strategies used for data reduction [7].

The following five types of data can be described for a dataset: mislabeled cases, redundant data, outliers, borderlines, and safe cases [16]. Mislabeled cases are noisy data objects that introduce *class noise.* Redundant data form clusters and those entries can be removed from the dataset since some of the clusters can be represented by other clusters. Outliers are significantly dissimilar to the other data objects in the dataset, and they may be noisy instances or very extreme cases. Borderline

* Corresponding author.
E-mail address: cagatay.catal@bte.tubitak.gov.tr (C. Catal).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.06.017

instances are very near to decision borders, and they are not considered as reliable as safe cases. Safe cases represent the remaining data points that should be used in the learning phase [16].

Although many researchers use the keywords "outlier" and "noisy instances" interchangeably, outlier and noisy instances actually refer to two different concepts that are used in data mining. While outliers are interesting to the analyst, noise that is present in data does not provide any significance by itself, yet the interest in outliers exempts it from noise removal [3]. For credit card fraud detection, these kinds of instances are very valuable because such instances may point to malicious activities [3]. Therefore, these instances are outliers for the credit card fraud detection domain. For the software engineering domain, these kinds of instances do not provide any interesting information because they are mostly caused by human error during the labeling process of modules. Therefore, these approaches to software engineering should be evaluated from the perspective of noise detection rather than outlier detection. Numerous algorithms have been developed to cleanse software measurement datasets within the last decade [29,6,24].

Data noise can be categorized into the following two groups: class and attribute noise. "Class noise occurs when an error exists in the class label of an instance, while attribute noise occurs when the values of one or more attributes of an instance are corrupted or incorrect" [11]. Most data mining studies have focused on detecting class noise, and more recent studies have shown that class noise impacts classifiers more severely than attribute noise [33]. Developers may not report each software fault that occurs during software tests, since a larger number of faults may create the perception of poor performance on the part of the developers [29]. In addition, developers may consider some faults as not significant for the project; thus, they may not report this kind of fault. As such, some software modules that should be labeled as faulty are labeled as non-faulty during the labeling process due to the inaccurate reporting of faults. In addition, data entry and data collection errors may also cause class and attribute noise [10].

Many empirical studies have concluded that the presence of class noise in the dataset adversely impacts the performance of machine learning-based classifiers [33,1,6,29]. In addition, the presence of attribute noise decreases the performance of classifiers as well [33]. Therefore, the preprocessing task of identifying noisy instances prior to the training phase of classifiers is a significant issue that should be addressed in a data mining initiative. Another solution used to prevent the degradation of classifier performance is the development of classifiers that are robust and not sensitive to noise. Robustness to noise is directly related to the machine learning algorithm. Van Hulse and Khoshgoftaar [29] reported that simple algorithms, such as the Naive Bayes algorithm, were more resistant to the impact of class noise than complex classifiers, such as the Random Forests algorithm. In addition, they showed that the noise resulting from the minority class is more severe than the noise resulting from the majority class. Therefore, faulty modules that are incorrectly labeled as non-faulty modules cause the most severe type of noise in software measurement datasets.

In this study, we developed a noise detection approach that uses threshold values for software metrics in order to capture these noisy instances. These thresholds were calculated using Shatnawi et al.'s [26] threshold calculation technique. We empirically validated the proposed noise detection technique on five public NASA datasets and built Naive Bayes-based software fault prediction models both before and after applying the proposed noise detection algorithm. After class labels of noisy instances were corrected, the performance of the fault predictors increased significantly. This observation and other results of experiments show that this technique is very effective for detecting class noise in software measurement datasets.

Twenty-one metrics are shown in NASA datasets, but we used thirteen software metrics for experiments since the remaining metrics are derived from four Halstead primitive metrics. Folleco et al. [6] used the same thirteen metrics to investigate the impact of increasing levels of class noise on a software quality classification problem. Khoshgoftaar and Van Hulse [11] applied the same metrics set to validate their attribute noise detection technique with NASA datasets.The threshold calculation approach used in this study is slightly different from Shatnawi et al.'s technique. They stated that they chose the candidate threshold value that had maximum values of both sensitivity and specificity. However, such a candidate threshold may not always exist. Therefore, we calculated the area under the ROC curve (AUC) that passes through three points (0,0), (1,1), and (PD,PF) where PD is the probability of detection and PF is the probability of false alarm. We chose the candidate threshold value that provides the maximum AUC. Details of the proposed approach are given in Section 3.1.

A two-stage algorithm is used for noise detection approach. Noisy instances were identified with noise detection approach and their labels were corrected before building the fault prediction model.The motivation behind experiments comes from the recently published related studies that are presented in Section 2. Recently, a number of techniques were developed to identify attribute and class noise. However, none of these noise detection techniques considered the use of threshold values for software metrics, even though software metrics are widely used in software quality prediction studies. Software quality prediction is a challenging research area in the field of software engineering [22]. In this study, we showed that the threshold values for software metrics can be identified using ROC analysis and that the performance of fault predictors using the Naive Bayes algorithm and logNum filter significantly improves if the class labels of identified noisy modules are corrected. The remainder of this paper is structured as follows:

- Section 2 presents related work;
- Section 3 introduces the proposed noise detection approach;
- Section 4 presents empirical case studies using real-world data from NASA projects; and
- Section 5 contains the conclusion and suggestions for future work.