Knowledge-Based Systems 66 (2014) 146–155

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Cost-sensitive learning for defect escalation

Victor S. Sheng [a,c,*], Bin Gu [b], Wei Fang [c], Jian Wu [d]

[a] Jiangsu Engineering Center of Network Monitoring, NanJing University of Information Science & Technology, NanJing, China
[b] Department of Medical Biophysics, Western University, London, Ontario, Canada
[c] Computer Science Department, University of Central Arkansas, Conway, AR, USA
[d] The Institute of Intelligent Information Processing and Application, Soochow University, Suzhou, China

ARTICLE INFO

Article history:
Received 2 September 2013
Received in revised form 12 April 2014
Accepted 21 April 2014
Available online 30 April 2014

Keywords:
Software defect escalation prediction
Cost-sensitive learning
Data mining
Defect escalation
Machine learning

ABSTRACT

While most software defects (i.e., bugs) are corrected and tested as part of the prolonged software development cycle, enterprise software venders often have to release software products before all reported defects are corrected, due to deadlines and limited resources. A small number of these reported defects will be escalated by customers whose businesses are seriously impacted. Escalated defects must be resolved immediately and individually by the software vendors at a very high cost. The total costs can be even greater, including loss of reputation, satisfaction, loyalty, and repeat revenue. In this paper, we develop a *Software defecT Escalation Prediction* (*STEP*) system to mine historical defect report data and predict the escalation risk of current defect reports for maximum net profit. More specifically, we first describe a simple and general framework to convert the maximum net profit problem to cost-sensitive learning. We then apply and compare four well-known cost-sensitive learning approaches for STEP. Our experiments suggest that *cost-sensitive decision trees* (*CSTree*) is the best methods for producing the highest positive net profit.

1. Introduction

Building large enterprise software is generally a highly complex and lengthy process, during which numerous software defect reports can exist and some of them may not be resolved when the software products are released (usually against a tight deadline) [14]. For example, it may be difficult to reproduce a reported error condition; there may be conflicts between desired product behavior and applicable standards; there may be uncertainty as whether a requested change is related to a defect or a request for enhancement; or it may be difficult to assess which of several products in a given environment may cause a reported error condition. Enterprise software vendors often have in place sophisticated processes for evaluating defect reports before release. This process entails a careful human expert review of each known bug, evaluation of trade-offs and delicate judgment. Still, after product release a small number of defects become "escalated" by customers, whose businesses are seriously impacted. Escalations of software defects require software vendors' immediate management attention and senior software engineers' immediate and continuous effort to reduce the business or financial loss to the customers. Therefore, software defect escalations are costly to the software vendors, with the associated costs amounting to millions of dollars each year. In addition, software defect escalations result in loss of reputation, satisfaction, loyalty and repeat revenue of customers, incurring extremely high costs in the long run for the enterprise software vendors [3,5].

In this paper, we further investigate a possible solution of developing a *Software defecT Escalation Prediction* (*STEP*) system. It is an extension of our previous work [28]. The objective of the STEP system is to assist human experts in the review process of software defect reports by modeling and predicting escalation risk using data mining technologies [4,15,20]. If the STEP system can accurately predict the escalation risk of known defect reports, then some escalations can be prevented by correcting those high-risk defect reports with a much lower cost within the software development and testing cycle before release. This would save a huge amount of money for the enterprise software vendors [9].

Indeed, the business goal of STEP (and many industrial applications using data mining) is to maximize the net profit, that is, the difference in the cost of defect resolution before and after introducing the data mining solution, as opposed to the usual data-mining measures such as accuracy, AUC (area under the ROC curve), misclassification cost [41], lift, or recall and precision combinations

* Corresponding author at: Jiangsu Engineering Center of Network Monitoring, NanJing University of Information Science & Technology, NanJing, China. Tel.: +1 5014505839.
E-mail addresses: ssheng@uca.edu (V.S. Sheng), jsgubin@nuist.edu.cn (B. Gu), hsfangwei@sina.com (W. Fang), jianwu@suda.edu.cn (J. Wu).

http://dx.doi.org/10.1016/j.knosys.2014.04.033
0950-7051/

[26]. However, it is clear that the net profit is not equivalent to any of these standard machine learning measures, and we have found little previous work that directly optimizes the net profit as the data mining effort.

In this paper, we first set up a simple framework in which the problem of maximum net profit can be converted to minimum total cost in cost-sensitive learning under certain conditions (see Section 2). We then apply and compare four well-known cost-sensitive learning algorithms on a defect report dataset to see how they perform, in terms of maximum net profit (Section 5). Our results (see Mini-Summary in Sections 5.5 and 6) suggest that cost-sensitive decision tree is best for producing the highest positive net profit. Conclusions drawn in this study not only help enterprise software vendors to improve profit in software production by reducing the cost of escalations, but also provide some general guidelines for mining imbalanced datasets [25,36,46] and cost-sensitive learning.

To the best of our knowledge, applying data mining for predicting software defect escalations is novel in software business. Software development is an extremely complex process, and hundreds or even thousands of defect reports may exist within a large enterprise software product. Predicting and prioritizing defect reports for evaluation and resolution is crucial in software engineering development. Our data-mining based STEP is the first and important step toward improving the effectiveness and efficiency of this process through automated analysis. As we will show in Sections 5–7, our STEP performs quite well. The system is currently deployed with product groups of a software vendor, and the system has quickly become a popular tool for prioritization.

In summary, this is a real-world application paper. It has four main contributions as follows. (1) It proposes a software defect escalation prediction system. (2) It converts a maximum net profit problem in software engineering to cost-sensitive learning. (3) It introduces negative values in the cost matrix, which are corresponding to the benefit obtained from correct classification. This is seldom discussed in existing cost-sensitive learning algorithms, which focus on the cost of misclassification. (4) The comparison studies of different approaches shed light for data mining practitioners on algorithm selection and for data mining researchers to see the performance of different techniques on real-world applications.

The paper is organized as follows: in the next section, we describe the maximum net profit and its relationship with cost-sensitive learning. Section 3 reviews several popular approaches of cost-sensitive learning. Then Section 4 describes an STEP dataset, and Section 5 compares different cost-sensitive learning approaches for maximum net profit. In Section 6, we further investigate the performance of different approaches on five real-world datasets. Section 7 discusses the deployment of our work, and Section 8 concludes the paper.

2. Maximum net profit and cost-sensitive learning

As we have discussed in the Introduction section, correcting defects after an escalation occurs is much more expensive than correcting defects before they become escalated. If we treat defect escalations as positive examples, then the cost of false negative *FN* (correcting an escalated defect) should cost many times more than the cost of false positive *FP* (correcting a non-escalated defect). If the cost of *FN* and *FP* is known, like in our study, then this would seem to be a straightforward cost-sensitive learning problem, in which the weighted misclassification costs should be minimized.

The problem is not that simple. The goal of the STEP (software defect escalation prediction) system (and many other real-world data mining applications) is to maximize the net profit after data mining is deployed. That is, we want to maximize the gain (or difference) with the data-mining effort compared to the previous practice. That is, we want to compare the cost (or profit) after data-mining based STEP is deployed to some default policy before STEP is deployed [28].

Let us first establish the formula for the net profit. Again we need to calculate the total cost before and after deploying STEP, and take the difference of the two to get the net profit. Let us assume that the cost of correcting an escalated defect after product release is 7 times as the cost of correcting a defect (non-escalated or would-be escalated) before product release. We assume that $FN = \$7{,}000$, and $FP = \$1{,}000$ (we did not use the actual numbers here for confidentiality reasons but the actual cost figures are some constant factor of the numbers used here). Let us first calculate the cost after deploying STEP ("gross profit"). When STEP makes a prediction on a dataset, it will make a number of positive predictions (predicted escalation) and negative predictions (predicted non-escalations). These predictions are often not 100% correct, so there will be certain numbers of true positive (tp), true negative (tn), false positive (fp), and false negative (fn) cases. If the software vender follows faithfully the STEP's predictions, it will fix all defects predicted positively, and ignore all defects predicted negatively. The cost of correcting all defects predicted positively by STEP is thus the multiplication of the number of defects predicated positively and the cost of correcting such a defect; that is, $(tp + fp) \times FP$. After the software is released, the would-be escalated defects predicted negatively by STEP (i.e., false negatives fn) will escalate and must be fixed at a much higher cost of $fn \times FN$. Thus, the total cost after deploying STEP is the sum of the two costs described above, plus the cost of the data mining effort (such as the cost of the tool, and computer and human cost of using the tool). If we ignore the cost of the data mining effort, then the total cost of deploying STEP is:

$$(tp + fp) \times FP + fn \times FN \quad (1)$$

Assume that the default policy (before deploying STEP) is to ignore all defect reports (or predict everything negatively) before software release, and then correct all escalated defects after release. Then using the same notation, the cost of this default policy is simply the cost of correcting all escalated defects. That is:

$$(tp + fn) \times FN. \quad (2)$$

Thus, the net profit is to subtract (1) from (2). That is:

$$Net\ profit = tp \times (FN - FP) - fp \times FP = 6000 \times tp - 1000 \times fp \quad (3)$$

Again, we obtained the above net profit equation under the assumption ($FN = \$7000$, and $FP = \$1000$). *FN* and *FP* could have different costs under different real-world applications. For example, if $FN = \$6000$, and $FP = \$1000$, then net profit can be calculated using $5000 \times tp - 1000 \times fp$; if $FN = \$5000$, and $FP = \$1000$, then net profit can be calculated using $4000 \times tp - 1000 \times fp$.

On the other hand, if the default policy is to correct all of the defects (predicting everything positively), the cost would be $(tp + fp + tn + fn) \times FP$. Subtracting (1) from the cost of the correcting-all policy above, the net profit would be: $1000 \times tn - 6000 \times fn$. Again, this net profit formula is based on the assumption ($FN = \$7000$, and $FP = \$1000$). *FN* and *FP* could have different costs under different applications. For example, if $FN = \$6000$, and $FP = \$1000$, then net profit can be calculated using $1000 \times tn - 5000 \times fn$; if $FN = \$5000$, and $FP = \$1000$, then net profit can be calculated using $1000 \times tn - 4000 \times fn$.

In general, the net profit varies with the default policy, and thus, maximizing net profit may not simply be equivalent to cost-sensitive learning given a cost metric.

However, we will show here that there is a very simple approach to convert a formula of maximum net profit to cost-sensitive learning under certain conditions. Cost sensitive