Predicting Software Escalations with Maximum ROI

Charles X. Ling[1], Shengli Sheng[1], Tilmann Bruckhaus[2], Nazim H. Madhavji[1]
[1]*Department of Computer Science, The University of Western Ontario*
London, Ontario N6A 5B7, Canada
{cling, madhavji, ssheng}@ csd.uwo.ca
[2]*Customer Network Services, Sun Microsystems, Inc.*
USCA22-104, 4220 Network Circle, Santa Clara, CA 95054
Tilmann.Bruckhaus@Sun.Com

Abstract

Enterprise software venders often have to release software products before all reported defects are corrected, and a small number of these reported defects will be escalated by customers whose businesses are seriously impacted. Escalated defects must be quickly resolved at a high cost by the software vendors. The total costs can be even greater, including loss of reputation, satisfaction, loyalty, and repeat revenue. In this paper, we develop an Escalation Prediction (EP) system to mine historic defect report data and predict the escalation risk of current defect reports for maximum ROI (Return On Investment). More specifically, we first describe a simple and general framework to convert the maximum ROI problem to cost-sensitive learning. We then apply and compare several best-known cost-sensitive learning approaches for EP. The EP system has produced promising results, and has been deployed in the product group of an enterprise software vendor. Conclusions drawn from this study also provide guidelines for mining imbalanced datasets and cost-sensitive learning.

1. Introduction

Building large enterprise software is often 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) [2, 3, 6].

The objective of the Escalation Prediction (EP) system is to assist human experts in the review process of software defect reports by modeling and predicting escalation risk using data mining technology [1, 10, 7]. If the EP system can accurately predict the escalation risk of known defect reports, escalations can be greatly reduced 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 [5].

Indeed, the business goal of EP (and many industrial applications using data mining) is to maximize the ROI (Return on Investment), not the usual data-mining measures such as accuracy, AUC (area under the ROC curve), lift, or recall and precision combinations [14]. We have found little previous work that directly optimizes the ROI as the data mining effort.

In this paper, we first set up a simple framework in which the problem of maximum ROI can be converted to minimum total cost in cost-sensitive learning [8, 9] under certain conditions (see Section 3). We then apply and compare several best-known cost-sensitive learning algorithms on a defect report dataset to see how they perform, in terms of maximum ROI (Section 5). 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 [12, 13] and cost-sensitive learning.

2. The EP Solution Architecture

The diagram in Figure 1 illustrates a general architecture of the Escalation Prediction System. Data is captured periodically from the software vendors' Online Transaction Processing (OLTP) systems which collect defect report and escalation information. Data may be captured weekly and stored indefinitely in a data mart. While the information in the OLTP systems continues to change from moment to moment, the information in the data mart remain constant so as to provide the historical data required for training predictive models in EP.

The historical data may be pre-processed before being fed into data-mining algorithms (see Section 5). Derived fields, historical information, and statistics are

added to yield the set of available input fields. These fields or attributes may include data fields which come directly from the defect report tracking system and derived fields obtained from the data and other sources.

The next step is to train and validate predictive models. See Sections 3-6 for details. Once one or more satisfactory models have been found, the most appropriate model may be selected and run against the most recent snapshot of defect report data. The predicted escalations may be then reported to the product group for evaluation and proactive resolution. The product group may provide feedback to allow for ongoing improvement of the overall escalation prediction and prevention effort.

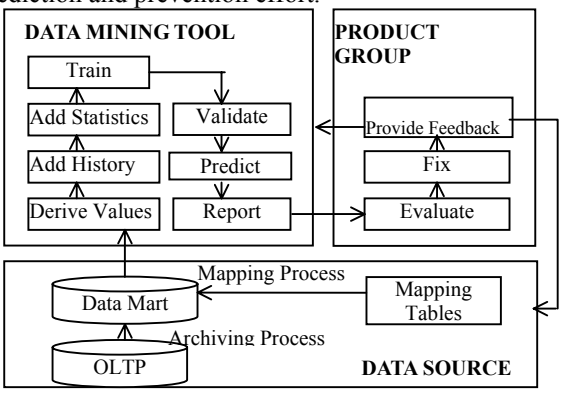

Figure 1. **The EP solution architecture.**

3. Maximum ROI 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 EP (Escalation Prediction) system (and many other real-world data mining applications) is to maximize the Return on Investment (ROI) after data mining is deployed. However, ROI is not equivalent to the usual measures such AUC, which is often used as a general measure when the cost metric and classification distributions are unknown. Indeed, ROI is rarely used as an optimization goal in building data-mining models[1].

Let us first establish the formula for ROI. In our study, the cost of correcting an escalated defect is estimated to be as much as 7 times the cost of correcting a (non-escalated or would-be escalated) defect (we assume that after a defect is corrected it will never be escalated in the future). That is, $FN = 7$, and $FP = 1$ (we did not use the actual numbers here for confidentiality reasons but the cost figures used are the same modulo to a constant factor). When EP makes a prediction on a dataset, it will produce a number of true positive (*tp*), true negative (*tn*), false positive (*fp*), and false negative (*fn*) cases. Thus, the ROI is:

$$ROI = tp \times (FN - FP) - fp \times FP = 6 \times tp - fp. \quad (1)$$

As long as the ROI, such as (1), can be expressed as a linear formula of *tp*, *fp*, *tn*, and *fn*, we can negate its coefficients in the linear formula, and re-assign the cost metric by the negated coefficients. For example, the cost metric for ROI of (1) can be converted and represented in Table 1. In the rest of the paper, we will study EP under this cost metric.

Through this simple conversion, the maximum ROI problem can be indeed solved by cost-sensitive learning algorithms.

Table 1: **Cost metric converted from (1).**

	Actual 0	Actual 1
Predict 0	0	0
Predict 1	1	-6

4. The Datasets

Our dataset consists of historical defect reports from industry software projects of an enterprise software vendor. Defect reports change over time and so there is an opportunity to learn from multiple different versions of a single defect report. Therefore, multiple data records in the dataset may belong to only one defect report. The data was collected in late 2004, and contains a total of about 400,000 records (defect report observations). The total number of attributes is 53, most of which are numerical attributes (including dates), and a few are nominal (such as product name). The system uses three types of inputs: 1) raw inputs which correspond to field values which are reported by users and stored in the defect tracking database, 2) row-level transformations of raw values, such as concatenating two string-valued raw inputs into a derived new string-valued input, and 3) statistical inputs which are derived from statistical analyses of all rows which fall into a given time period, such as a fiscal quarter.

[1] There are two common definitions of ROI: the amount of the benefit divided by, or subtracted by, the cost of investment. In this paper we take subtraction in the ROI calculation, and the investment is the cost of deploying data mining.

Statistical inputs were particularly useful for string valued raw inputs because string-valued data is typically difficult to use for machine learning. For example, a raw input for "CUSTOMER_NAME" may have 20,000 possible customer name values. Therefore, the system calculates statistics for string-valued inputs. These statistics include counts (number of defects submitted by customer "abc", and number of escalations raised by customer "abc"), means (number of escalations over number of defects submitted for hardware platform "xyz"), and the probability of an escalation, and so on. Further analysis of these inputs and their utility for prediction is of great interest but beyond the scope of this paper.

The target attribute is binary (escalation or no escalation). The whole dataset is split up into training and test sets according to the defect report date. The training set contains about 165,000 records before a certain date, and the test set contains 232,000 records after that date. The dataset is very imbalanced, with slightly less than 1% of the positive examples (escalated defects). In the training set, there are only about 1,000 positive examples (escalated defects).

5. Comparing cost-sensitive learning approaches for EP

As we discussed in Section 3, maximum ROI is the ultimate measure to compare different data-mining approaches on the EP dataset. However, the original dataset is very large with many different learning algorithms used, some with bagging [4] and boosting [11, 16], it would be difficult to use exactly the same training and test datasets in all comparisons. To make the ROI result comparable, we use "unit" ROI, defined as the ROI divided by the number of records in our comparison. In the comparisons below, we will report the values of the unit maximum ROI.

Note that even if a data mining method can obtain the unit maximum ROI of 1, the saving to the enterprise software vendors can be quite significant. If we assume that 1 represents $5,000 (a reasonable figure), then with 1,000 software defect reports against a software product to be released, a total saving with EP's prediction is 1000×5000=5 million dollars.

The first approach is to rebalance the data (to an equal portion between positive and negative examples) using under-sampling, applies regular (cost-insensitive) learning algorithms (such as J48 [19], which is similar to the decision tree algorithm C4.5 [17, 18]), and finds the best threshold for classification that produces the maximum ROI on the validation set. The best threshold is then applied to the test set to obtain the unit maximum ROI. This approach is simple to implement as any regular learning algorithms can be used, and is intuitive as maximum ROI is sought directly.

The second approach, called costing, achieves true cost-sensitivity via an advanced sampling (rejection sampling) with a provable performance guarantee [20]. As the method is truly cost-sensitive, the classification results can be used directly to calculate the minimal total cost, which is equivalent to the (negated) maximum ROI. We apply costing (rejection sampling and bagging) with J48 used in the first approach, to see if it would outperform the first one. As the training set after rejection sampling contains about 30,000 examples, much larger than the one in the first approach (only about 2,000), we only run bagging 1, 10, and 100 times. Table 2 shows the unit maximum ROI for rebalance approach and costing.

Table 2: **Unit maximum ROI.**

	Bagging Iterations		
	1	10	100
Rebalance	0	0	1.743
Costing	1.262	1.955	1.779

Comparing the two different sampling approaches, we can draw several interesting conclusions. First of all, rejection sampling with bagging is just a little better than the simple rebalancing with bagging when the bagging iterations reach 100. Second, similar to the conclusion drawn earlier, the results do not always improve with more bagging iterations.

The third approach is the decision tree learning algorithm which uses directly the minimal total cost or maximum ROI as its split criterion [15]. Several improvements have been made, including post pruning.

The unit maximum ROI for this approach is obtained as 1.67, comparable to the results with J48 in rebalancing and costing (Table 2). The tree directly predicts labels of the test examples without the need to search the best threshold, as this approach is also true cost-sensitive. The rules produced from this approach have been shown to the software vendor where the data come from, and are regarded as reasonable, useful, and interesting. Further analysis of the decision tree shows that the statistical inputs (see Section 4) were generally highly useful for modeling and that statistics about the customer (e.g., customer name and the role of the submitter) and the customer environment (such as the hardware platform of the OS version) were also highly useful.

To see if bagging can be used to improve ROI further, we have also applied bagging to this approach. Indeed, the result is even better: the unit maximum ROI reaches over 2.4, the highest of all approaches we have tried. This model is used in the EP system for evaluation (see next Section).

6. Deployment

Our EP system has been in deployment for several months in the product group of an enterprise software vendor where the dataset comes from. It has been used to make suggestions on current defect reports with high

risks of escalation. As the software vendor must wait to see which defect reports are actually escalated after the product is released, and the would-be escalations will not happen after they are accurately predicted by EP and corrected before software release, the final evaluation cannot be obtained until the software has been released and has been in use by customers for some period of time (in the order of one or more quarters).

We have evaluated EP using the defect reports submitted or updated during the most recent three weeks in the test set. Any records corresponding to defect reports which had already been escalated at the time of preparing the data set were also removed. After EP makes its predictions, the results are compared to the actual escalations happened up to date. The EP prediction performance is quite good. The lift chart of EP's prediction is well above the random diagonal line: at top 10 percentile (among top 10% of the most likely predicted escalations), about 70% of the actual escalations are predicted; at 20 percentile about 85%; and at 30 percentile about 90%. These results are of significant business value to the software vendor. In addition, the product group has provided positive feedback on the performance of the EP system, citing that it "catches" defect reports that, after thorough evaluation by specialist, are considered likely candidates for future escalations. For instance, some defect reports have been found to have been assigned a lower than appropriate priority. After a prediction of high escalation risk becomes available such a defect report can be upgraded to a higher priority which will lead to expedited evaluation and resolution.

7. Conclusions

In this paper, we have made a case for predicting and preventing escalations from known product defect reports for enterprise software vendors. By applying data mining technologies that have been used successfully in many other fields, enterprise software vendors can proactively predict and resolve known product defect reports with the greatest risk of escalation. This can save software vendors an enormous amount of software maintenance cost.

An escalation prediction solution based on data-mining for the maximum ROI has been proposed and tested, and is currently deployed at an enterprise software vendor. Preliminary results provide evidence that we can indeed make useful predictions about the escalation risk of product defects. In our future work, we plan to continue to improve the effectiveness of the EP system and track its results.

8. References

[1] Berry, M.J.A., and Linoff, G. 1997. *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley & Sons.

[2] Beohm, B.W., and Basili, V. 2001. Software Defect Reduction Top 10 List. *Computer* 34(1): 135-137.

[3] Boehm, B.W. 1981. *Software Engineering Economics*. Prentice-Hall Advances in Computing Science & Technology Series.

[4] Breiman, L. 1996. Bagging Predictors. Machine Learning 24(2): 123-140.

[5] Bruckhaus, T., Ling, C.X., Madhavji, N.H., and Sheng, S. 2004. Software Escalation Prediction with Data Mining. *Workshop on Predictive Software Models (PSM 2004), A STEP Software Technology & Engineering Practice.*

[6] Chulani, S., and Boehm, B.W. 1997. Modeling Software Defect Introduction. *California Software Symposium*, Nov. 1997.

[7] Dai, H. (editor). 2003. *Proceedings of The International Workshop on Data Mining for Software Engineering and Knowledge Engineering*.

[8] Drummond, C., and Holte, R.C. 2003. C4.5, Class Imbalance, and Cost Sensitivity: Why under-sampling beats over-sampling. *Workshop on Learning from Imbalanced Datasets II.*

[9] Elkan, C. 2001. The Foundations of Cost-Sensitive Learning. *International Joint Conference of Artificial Intelligence (IJCAI 2001),* 973-978.

[10] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R.(editors). 1996. *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press.

[11] Freund Y. and Schapire, R.E. 1996. Experiments with a New Boosting Algorithm. *Proceeding of International Conference on Machine Learning (ICML),* 148-156.

[12] Japkowicz. N. 2001. Concept-Learning in the Presence of Between-Class and Within-Class Imbalances, *Proceedings of the Fourteenth Conference of the Canadian Society for Computational Studies of Intelligence (AI'2001).*

[13] Joshi, M.V., Agarwal, R.C., and Kumar, V. 2001. Mining needles in a haystack: classifying rare classes via two-phase rule induction. *In SIGMOD'01 Conference on Management of Data*.

[14] Ling, C.X., and Li, C. 1998. Data Mining for Direct Marketing: Specific Problems and Solutions. *Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, 73-79.

[15] Ling, C.X., Yang, Q., Wang, J., and Zhang, S. 2004. Decision trees with minimal costs. *Proceedings of International Conference on Machine Learning (ICML).*

[16] Niculescu-Mizil, A., and Caruana, R. 2001. Obtaining Calibrated Probabilities from Boosting. *AI Stats*.

[17] Quinlan, J.R. 1986. Induction of decision trees. *Machine Learning*, 1(1): 81-106.

[18] Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[19] Witten, I.H., and Frank, E. 2000. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco.

[20] Zadrozny, B., Langford, J., and Abe, N. 2003. Cost-Sensitive Learning by Cost-Proportionate Example Weighting. *Proceedings of International Conference of Data Mining (ICDM).*