International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Cost-Sensitive Boosting Networks for Data Defect Prediction

Liya A Asharaf[1], Vini Vijayan[2]

[1]P.G Scholar, Sarabhai Institute of Science and Technology, Vellanad, Trivandrum, (DIST)-Kerala 695543, India

[2]Assistant Professor, Dept. of CSE, Sarabhai Institute of Science and Technology, Vellanad, Trivandrum, (DIST)-Kerala 695543

Abstract: ***Software Defect prediction which classify the software modules into defect prone and not defect prone category and plays an important role in reducing the coast of software development and maintaining high quality software system. Existing defect prediction model face two challenges 1)Class Imbalance 2)High Dimensionality. In this paper a new cost sensitive boosting networks for data defect prediction is proposed. Cost sensitive learning is a method to solve class imbalance problem. Cost sensitive learning takes costs such as the misclassification coast into consideration. High dimensionality can be handled by using feature selection methods. For feature selection purpose three cost sensitive feature selection algorithms are used namely Cost-Sensitive Variance Score (CSVS), Cost-Sensitive Laplacian Score (CSLS), and Cost-Sensitive Constraint Score. The proposed techniques are evaluated on the data set taken from NASA MDP data set .The experiments shows that cost sensitive feature selection methods are more efficient than traditional one in reducing the total cost. The accuracy and class imbalance problem can be better solved by using the method like swarm and bagging techniques.***

Keywords: Cost-sensitive learning, feature selection, software defect prediction , swarm technique and bagging technique.

1. Introduction

Software defect prediction is one of the most active research areas in software engineering. Software defect predictors which classify the software modules into defect-prone and not-defect-prone classes are effective tools to maintain the high quality of software products. The early prediction of defect-proneness of the modules can allow software developers to allocate the limited resources on those defect-prone modules such that high quality software can be produced on time and within budget. Existing SDP work can be categorized into three types 1) estimating the number of defects existing in a software system, 2) mining defect associations, and 3) classifying software modules into defect-prone and not-defect-prone categories.

Software defect prediction still remains a difficult problem to be solved, and is faced with two challenges [1]: high dimensionality, and class imbalance. As modern software systems grow in both size and complexity, the number of features (i.e., software metrics) extracted from software modules becomes much larger than ever before, and these features may be redundant or irrelevant [2]. It is a great challenge for classification algorithms to deal with such superabundant features. As an important pre-processing procedure, feature selection is beneficial to facilitate data understanding, to reduce the storage requirements, and to overcome the curse of dimensionality for improved prediction performance. As shown in previous studies [2], [3] feature selection is effective to deal with the high dimensionality problem in SDP.

The second challenge for SDP is the class imbalanced data, where the majority of defects in a software system are only found in a small portion of its modules. In such cases, standard machine learning based SDP models may be inaccurate for or never predict the minority class that is the defect-prone module, because they do not explicitly consider different error costs or class distributions.

Cost-sensitive learning has attracted increasing attention in the SDP domain [4] which explicitly considers those different error costs, and aims to minimize the total expected costs rather than the classification error rates. In general, there are two types of errors in software defect prediction. Type I is defined as misclassifying a not-defect-prone module as defect-prone, while Type II misclassification is when a defect-prone module is predicted as not-defect-prone. The cost incurred by Type II misclassification is much higher than that of Type I misclassification.

In most cost-sensitive learning based SDP studies, cost information is used in the classification stage instead of in the feature selection stage. But considering the valuable cost information in the feature selection stage may further boost the performances of SDP models because features associated with the minority class that is defect-prone modules are more likely to be selected. The goal of this paper is to develop a cost-sensitive boosting network for data defect prediction is a method for SDP by using cost information in both the classification and the feature selection stages. Here the accuracy of the software can be improved by using swarm technique and bagging techniques can be used to solve the class imbalance problem. Work has been done on cost-sensitive feature selection and three cost-sensitive feature selection algorithms are also used .The experimental results on the public NASA Metrics Data Program repository [5] validate the efficacy of the proposed methods.

2. Related Work

1) A General Software Defect-Proneness Prediction Framework

Predicting defect-prone software component is an economically important activity and so has received a good

deal of attention. However, making sense of the many, and sometimes seemingly inconsistent, result is difficult. In this paper a general framework for software defect prediction that supports unbiased and comprehensive comparison between competing prediction systems. The framework is comprised of scheme and defect prediction components. The scheme evaluation analyzes the prediction performance of competing learning schemes for given historical data sets.

2) The Foundations of Cost-Sensitive Learning

This paper revisits the problem of optimal learning and decision-making when different misclassification errors incur different penalties. It characterizes precisely but intuitively when a cost matrix is reasonable, and shows how to avoid the mistake of defining a cost matrix that is economically incoherent. For the two-class case, prove a theorem that shows how to change the proportion of negative examples in a training set in order to make optimal cost-sensitive classification decisions using a classifier learned by a standard non-cost sensitive learning method. It argue that changing the balance of negative and positive training examples has little effect on the classifiers produced by standard Bayesian and decision tree learning methods. Accordingly, the recommended way of applying one of these methods in a domain with differing misclassification costs is to learn a classifier from the training set as given, and then to compute optimal decisions explicitly using the probability estimates given by the classifier.

3) Reflections on the NASA MDP data sets

The NASA metrics data program (MDP) data sets have been heavily used in software defect prediction research. To highlight the data quality issues present in these data sets, and the problems that can arise when they are used in a binary classification context. A thorough exploration of all 13 original NASA data sets, followed by various experiments demonstrating the potential impact of duplicate data points when data mining. Firstly researchers need to analyze the data that forms the basis of their findings in the context of how it will be used. Secondly, the bulk of defect prediction experiments based on the NASA MDP data sets may have led to erroneous findings. This is mainly because of repeated/duplicate data points potentially causing substantial amounts of training and testing data to be identical.

3. Proposed Methods

In most real-world applications, different misclassifications are usually associated with different costs. Here cost matrix is used and where the element indicates the cost value of classifying a sample from the*i* th class a *j* th class. The diagonal elements in the cost matrix are zero because a correct classification will cause no cost.

Table 1: Cost Matrix

	I_1	…	I_{c-1}	O
I_1	0	…	C_{II}	C_{IO}
…	…	…	…	…
I_{c-1}	C_{II}	…	0	C_{IO}
O	C_{OI}	…	C_{OI}	0

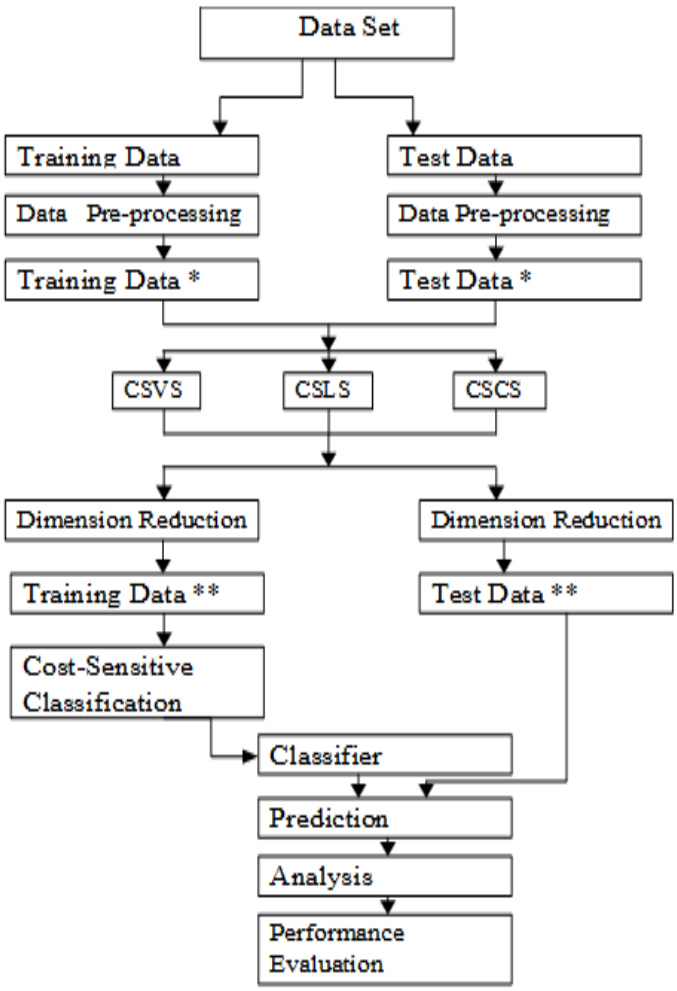

Figure: Architectural Diagram of the proposed system

Here the training data is used to train the system and data preprocessing is performed before feature selection process. In feature selection the three algorithms are performed to find the minimized attributes. In this dimension reduced data set classification is performed and ranked list of data are obtained and range of values are calculated. Test data values are compared with this range of values and predict whether the data is defect prone or not.

Cost-Sensitive Feature Selection

CSVS: Similar to Variance score, we assume that the variance of a good feature in the out-group class should be larger than that of the in-group classes. Thus, the Cost-Sensitive Variance Score (CSVS) of the r th feature denoted, which should be maximized, is defined as

$$f(k)\begin{cases}(c-2)C_{II}+C_{IO} & if\ k=1,2,...c-1\\ (c-1)C_{OI} & otherwise\end{cases} \quad (1)$$

Cost-Sensitive Laplacian Score

Let y*i* denote the class label of sample x*i*. Similar to Laplacian Score, the cost-sensitive laplacian score Lr of the *r*-th feature, which should be minimized, is defined as follows:

$$L_r = \sum_{i,j}[-cost(y_i, y_j)](f_{ri} - f_{rj})^2 S_{ij} - \lambda \sum_i f(i)(f_{ri} - \mu_r)^2 D_{ii} \quad (3)$$

Where cost(y*i*, y*j*) can be easily obtained from the cost matrix. D is a diagonal matrix and Sij, which reflect the neighborhood relationship between sample X*i* (*i*=1,…,m), is defined as follows :

$$S_{ij} = \begin{cases} e^{-\frac{\|X_i - X_j\|^2}{t}}, & \text{if } X_i \text{ and } X_j \text{ are neighbors} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Where t is a constant to be set, and „Xi and Xj are neighbors" means that either Xi is among k nearest neighbors of Xi, or Xj is among k nearest neighbors of Xi. There is a regularization coefficient λ , which is used to balance the contribution of the two terms in Eq.(3). Since f(*i*) is usually larger than cost(y*i*, y*j*), set λ=1 in this paper.

Cost-Sensitive Constraint Score

Given a set of samples X=[x1, x2, … , xm], we can utilize its pairwise must-link constraints M={(xi, xj)|xi and xj belong to the same class} and pairwise cannot-link constraints C={(xi, xj)| xi and xj belong to the different class} as the supervision information. The cost-sensitive constraint score of Cr, which should be minimized, is defined as follows:

$$C_r = \sum_{(x_i - x_j) \in M} cost(y_i, y_j)(f_{ri} - f_{rj})^2 - \lambda \sum_{(x_i - x_j) \in C} f(y_i)(f_{ri} - f_{rj})^2 \quad (5)$$

Where C is sets of pair wise cannot-link constraints and M is sets of pair wise must-link constraints. λ is a regularization coefficient, which is used to balance the two terms in Eq.(5).

4. Data Sets

The data sets used in this study come from the public NASA Metrics Data Program (MDP) repository [5], making our proposed methods repeatable and verifiable.

5. Performance Measurements

For better evaluating the performances in the cost-sensitive learning scenarios,the *Total-cost* of misclassification, which is a general measurement for cost-sensitive learning [6], is used as one primary evaluation criterion in our experiments. On the other hand, as shown in Table I, the classification results can be represented by the confusion matrix with two rows and two columns reporting the number of true positives (*TP*), false positives (*FP*), false negatives (*FN*), and true negatives (*TN*).

Table: Defect Prediction Confusion Matrix

		Actual	
		Defect-prone	Not-defect-prone
Predicted	Defect-prone	*TP*	*FP*
	Not-defect-prone	*FN*	*TN*

From the confusion matrix, sensitivity and accuracy can be defined

$$Sensitivity = \frac{TP}{(TP + FN)}$$

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

where sensitivity measures the proportion of defect-prone modules correctly classified, and accuracy measures the proportion of samples correctly classified among the whole population. In addition to the *Total-cost*, we also adopt the sensitivity and accuracy of the classification results as evaluation measures.

6. Conclusion

To address the class-imbalance and high-dimensional data problems of software defect prediction a cost-sensitive learning method is proposed where the cost information is utilized not only in the classification stage but also in the feature selection stage. Here three cost-sensitive feature selection methods, called CSVS, CSLS, and CSCS, are used Experimental results demonstrate that the proposed cost-sensitive boosting networks for data defect prediction methods outperform single-stage cost-sensitive learning methods, while the proposed cost-sensitive feature selection methods perform better than conventional feature selection methods. Two new techniques called swarm and bagging are used to increase accuracy and reduce class imbalance problem.

References

[1] L. Pelayo and S. Dick, "Evaluating stratification alternatives to improve software defect prediction," *IEEE Trans. Rel.*, vol. 61, pp. 516–525, 2012.
[2] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: An investigation on feature selection techniques," *Software Pract. Exper.*, vol. 41, pp. 579–606, 2011.
[3] T. M. Khoshgoftaar, K. Gao, and A. Napolitano, "An empirical study of feature ranking techniques for software quality prediction," *Int. J. Softw. Eng. Know.*, vol. 22, pp. 161–183, 2012.
[4] T. M. Khoshgoftaar, E. Geleyn, L. Nguyen, and L. Bullard, "Costsensitive boosting in software quality modeling," in *Proc. 7th IEEE Int. Symp. High Assurance Syst. Eng.*, Tokyo, Japan, 2002, pp. 51–60.
[5] M. Chapman, P. Callis, and W. Jackson, 2004, Metrics Data Program [Online]. Available: http://mdp.ivv.nasa.gov

[6] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, pp. 63–77, 2006.