International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Improving Software Quality Using Two Stage Cost Sensitive Learning

Ann Joshua

M. Tech Computer Science, Mount Zion College of Engineering, India

Abstract: ***The quality of software depends heavily on how accurately it works. The accuracy is determined by the fact that the less the software modules are defect prone, the more accurate the software will be. So software defect prediction which classifies software modules into defect prone and non-defect prone categories is an important area where a lot of research works are being done. Cost sensitive learning that has been adopted in software defect prediction aims to minimize total expected cost. In this paper a two-stage cost sensitive learning is proposed where the cost information is used in the feature selection stage and in the classification stage. Three cost sensitive algorithms, Cost-Sensitive Variance Score, Cost-Sensitive Laplacian Score, and Cost-Sensitive Constraint Score are proposed. The results of the proposed methods are analyzed with datasets from NASA.***

Keywords: software defect prediction, two-stage cost sensitive learning, variance score, laplacian score, constraint score

1. Introduction

The competition in software industry is increasing day by day. As technology advances, lots of industries are developing high-end software with similar functionality. So the major criterion for the customers to buy software is its accuracy rather than the functionality it provides. A software defect is an error or failure in a system that prevents the software from generating the intended outcome. Predicting the defects in an earlier stage of software development helps in reducing the development time. Software defects incur costs in terms of quality and time. The results of software defect prediction provide a list of defect-prone and non defect-prone modules. The larger the software, the more relevant the defect prediction will be in the software industry.

The number of features to be extracted increases as the size of the software increases and many of these features may be redundant or irrelevant. The two challenges faced in software defect prediction are high dimensionality and class imbalance problem. High dimensionality results when classification algorithms have to deal with superabundant features. Feature selection which is an important preprocessing procedure is capable of dealing with the high dimensionality problem. The class imbalance problem occurs when the majority of defects in a software system are found only in a small portion of the modules. The two approaches drawn from machine learning for addressing the class imbalance problem are stratification and cost sensitive learning. Stratification is done by creating a balanced data set through adding more samples to the minority class or reducing the sample number of the majority class. Cost sensitive learning takes the misclassification cost along with other types of cost in data mining and aims to minimize the total cost.

The goal of this paper is to develop a two-stage cost-sensitive (TSCS) learning method for SDP by using cost information in both the classification and the feature selection stages. Three novel cost-sensitive feature selection algorithms has been developed by emphasizing samples with higher misclassification costs, and de-emphasizing those with lower misclassification costs in the feature selection stage. The experimental results on the public NASA Metrics Data Program repository validate the efficacy of the proposed methods.

2. Related Work

Cost sensitive learning has been studied in the data mining community for addressing the class imbalance problem. The cost information is used to evaluate misclassification cost from different types of errors. Misclassification cost corresponds to the cost occurred by misclassifying any example of class *i* as class *j*. Standard learning algorithms are designed to make classifiers cost sensitive. For this, the training set given to the learning algorithm is rebalanced by changing the proportion of positive and negative training examples in the training set.

Metacost is a procedure that has been proposed to make classifiers cost sensitive. In this, the underlying classifier is treated as a black box requiring no knowledge of its functioning. It estimates class probabilities by learning multiple classifiers. It works by forming multiple bootstrap replicates of the training set and learning a classifier on each, estimating each class's probability for each example. Metacost produces large cost reductions compared to cost-blind classifier. It is applicable to any number of classes and can be effectively applied to large databases.

Cost sensitive boosting has been done on neural networks that are used for software defect prediction. A neural network consists of an input layer that receives external inputs; one or more hidden layers and an output layer that gives the classification results. When data are presented at the input layer, the network nodes perform calculations in the successive layers until an output value is obtained at each of the output nodes. The output layer has a node to indicate whether the software modules are defect-prone or not. Over sampling is a method used to make a neural network classifier cost sensitive in which the number of high cost training examples are increased. Under sampling is another method in which the number of inexpensive training examples is decreased. AdaBoost is an effective method by

which a composite classifier is constructed by sequentially training individual base classifiers. During the training process, the weights of training examples are adjusted in a way that the weights of misclassified examples are increased while the weights of correctly classified examples are decreased in each training round. Finally, the constructed individual classifiers are combined to form the composite classifier by weighted or simple voting schemes.

3. Proposed Methods

The proposed method involves a two-stage cost sensitive learning for software defect prediction in which the cost information is used in both the feature selection stage and in the classification stage. In cost sensitive feature selection, the attributes that are associated with the defect prone modules of the software are selected. The cost sensitive classification makes sure that the classifier is not dominated by the not-defect-prone module.

Fig. 1 illustrates the general architecture of the proposed method. The historical data captured from software systems are divided into the training set and the test set. The features are selected using the variance score, laplacian score and constraint score algorithms. The results will contain only optimal features. Then resampling is done by which the values of attributes that do not contribute for defect prediction are reset. The next step is to build a decision tree in which the selected attributes are given a rank. The ranking is done in such a way that the most relevant attribute is given the highest rank. Then the test data is loaded and the learned model is evaluated on the test data set. The performance evaluation is done by analyzing the data set on the three algorithms. Different measures such as processing time, sensitivity, accuracy and misclassification cost are calculated.

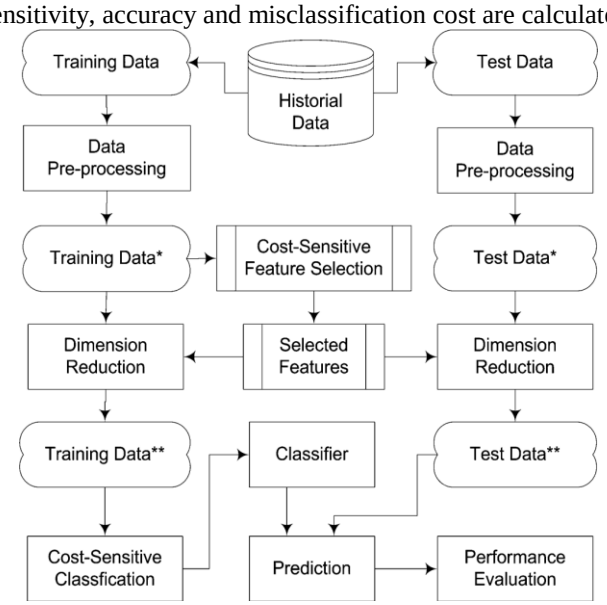

Figure 1: General architecture of the TSCS method

3.1 Preprocessing

The training and test data are preprocessed in order to improve the quality of data. This includes removing irrelevant attributes, inconsistent data and repeated rows. If the value of an attribute results in both defect-prone and not-defect prone modules, that particular attribute cannot be taken for making a factor in defect prediction process. Such attributes are removed in this stage. Thus after pre-processing, the dataset will be a reduced one containing only relevant attributes.

3.2 Feature Extraction

The features are extracted in a cost-sensitive way that addresses the issue of classification in the presence of varying costs associated with different types of misclassification. Consider the class labels $\{1,\ldots,c\}$. The cost of misclassifying a sample from the ith class (i is in $\{1,\ldots,c\text{-}1\}$) as the cth class will be very high. The class from 1st class to c-1th class is the in-group class where as the cth class is the out-group class. The cost of misclassifying a sample from the out-group class as being from the in-group class is known as the cost of false acceptance. The cost of misclassifying a sample from the in-group class as being from the out-group class is known as the cost of false rejection. The cost of misclassifying a sample from the in-group class as being from another in-group class is known as the cost of false identification. Defect-prone modules are considered as being from the out-group class and not-defect-prone modules from the in-group class.

A cost matrix is constructed as shown in Table I, where the element cost (i,j) indicates the cost of classifying a sample from the ith class as the jth class. This mainly focuses on calculating the cost of misclassification so that identifying it at an earlier stage will reduce the total cost. The diagonal elements in the cost matrix are zero because they will represent a correct classification. The importance of each class can be identified from the cost matrix. It is calculated from the cost of false identification and false rejection for the in-group class and from the cost of false acceptance for the out-group class. It is necessary to know how far a class is important, as it is one of the main factors used in the feature selection algorithms..

Table 1: The 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

3.2.1 Cost Sensitive Variance Score

Variance score is a simple unsupervised evaluation criterion of features. It selects features that have the maximum variance among all samples with the basic idea that variance among a feature space reflects the representative power of this feature. In this method, the mean of each attribute among all the samples is calculated. Then the variance of each feature from the mean value is calculated. The variance of a good feature from the out-group class should be larger than that of the in-group class. So the features with maximum

variance score are considered to be important in the defect prediction process and they are selected.

3.2.2 Cost Sensitive Laplacian Score

Laplacian score is an unsupervised feature selection method which performs feature selection according to the constraint preserving ability of features with stronger locality preserving ability. The laplacian score of a feature is calculated by considering the neighborhood relation between two samples and the cost of misclassification of a sample from one class as being from another class. The misclassification cost can be obtained from the cost matrix. The features having minimum laplacian score are selected.

3.2.3 Cost Sensitive Constraint Score

Constraint score is a semi-supervised feature selection method which selects features with stronger locality preserving ability. It uses must-link and cannot-link pair-wise constraints as supervision information. Must-link pair-wise constraints specify a pair of data samples that belong to the same class and cannot-link pair-wise constraints specify a pair of data samples that belong to different classes. The constraint score is calculated by considering the importance of the class for must-link constraints and the cost of misclassification for cannot-link constraints. The features having minimum constraint score are selected.

3.3 Defect Prediction

Once the relevant features are selected, they are re-sampled. It involves resetting the values of attributes that do not contribute for defect prediction to zero. Then a decision tree is constructed by which each of the selected attribute is given a rank. The attribute with highest rank will form the root of the tree. Each attribute will be assigned the range of values it falls within. The defect prediction is done by finding out whether the attribute values are defect prone or not. The test data is loaded and the learned model is evaluated on the test data set.

4. Analysis

The classification results can be represented by the confusion matrix with two rows and columns as shown in Table II. True Positive (TP) refers to the number of defect-prone values that are correctly classified. True Negative (TN) refers to the number of not-defect-prone values that are correctly classified. False Positive (FP) refers to the number of not-defect-prone values that are classified as defect-prone. False Negative refers to the number of defect-prone values that are classified as not-defect-prone. Sensitivity measures the proportion of defect-prone modules correctly classified. Accuracy measures the proportion of samples correctly classified among the whole population.

Table 2: Confusion matrix

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

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

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

The analysis also includes the comparison of the three feature selection algorithms. The factors included for comparison are the number of true positive, false positive, true negative, false negative, sensitivity, accuracy, processing time and the misclassification cost.

3.4 Conclusion

A two-stage cost-sensitive learning by utilizing cost information in the feature selection stage and in the classification stage has been proposed in this paper. The datasets from NASA has been used for evaluating the proposed method. It has been found that the cost-sensitive learning performed better than cost-blind learning. The usage of cost information in the feature selection stage resulted in early defect prediction so that lesser time and effort are required for maintenance of the software. Identifying the misclassification cost at an earlier stage helps in reducing the total cost of building the software.

References

[1] Mingxia Liu, Linsong Miao, and Daoqiang Zhang, "Two –Stage Cost-Sensitive Learning for Software Defect Prediction ," IEEE transactions on reliability, Vol. 63, No.2, June 2014
[2] C. Elkan, "The foundations of cost-sensitive learning," in Proc. 17th Int. Joint Conf. Artif. Intell., Seattle, WA, USA, 2001, pp. 973–978
[3] P. Domingos, "MetaCost: A general method for making classifiers cost-sensitive," in Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, San Diego, California, USA, 1999, pp.155–164.
[4] P. DomingosT. M. Khoshgoftaar, A. S. Pandya, and D. L. Lanning, "Application of neural networks for predicting defects," Annal. Software Eng., vol. 1,pp. 141–154, 1995.
[5] D. Zhang, S. Chen, and Z. Zhou, "Constraint Score: A new filter method for feature selection with pairwise constraints," Pattern Recogn., vol. 41, pp. 1440–1451, 2008.