Send Orders for Reprints to reprints@benthamscience.ae

44 *The Open Cybernetics & Systemics Journal,* 2015, *9,* 44-49

Open Access

Risk Prediction Model Based on Improved AdaBoost Method for Cloud Users

Lin Zhang[1,2,3,*], Kaili Rao[1], Ruchuan Wang[1,2,3] and Yishang Jia[1]

[1]*College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;* [2]*Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Jiangsu Province, Nanjing 210003, China;* [3]*Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nangjing University of Posts and Telecommunications, Ministry of Education Jiangsu Province, Nangjing 210003, China*

Abstract: Considering the problem how to protect the cloud services from being destroyed by cloud users, the risk-prediction model based on improved AdaBoost method is proposed. The risk prediction is regarded as two-class classification problem, and the risk of new cloud users could be predicted by the attributes of historical cloud users. In order to improve the result of predicted, AdaBoost method is adopted in this paper. The error rate of the last training is used to adjust the sample distribution of the next training, which can make the next training have stronger ability of identification for the error-classified samples. At the same time the weight of each weak classifier is set. After all, the strong classifier is generated by combined the weak classifiers through voting, which can improve the overall result of classification. Considering the wrongly-predicted cost, AdaBoost method is improved. The method of cost-sensitive is added into the model in order to minimal the misclassified-cost. Experiments show that the cost-sensitive AdaBoost method has better classification result than the traditional ones and it can predict the risk of the new cloud user effectively and protect the security of the cloud services.

Keywords: AdaBoost algorithm, cost-sensitive, risk prediction, strong classifier.

1. INTRODUCTION

With the rapid development of cloud computing [1], the numbers and types of cloud services are increasing explosively. Meanwhile, the numbers of cloud users is increasing. In the face of the request of many cloud users, it has became an important research direction that how to avoided destroying cloud services

At present, aiming at the question of how to protect the security of the cloud users, many researches have been done researched, and achieved certain results. Such as, password-based authentication, authentication based on smart card, etc. However, even if users pass the authentication, maybe they will destroy the security of cloud services because of the benefit of individual or group. As to the problem, many methods have been proposed by international and domestic academics. The paper [2] proposed a control model based on RBAC, the model divides the users into two categories, black and white list, and only the users in the white list can access to cloud services. But, because of the uncertainty of the users, even if the users in the white list, they also can destroy the security of cloud services. The paper [3] proposed a hierarchical cloud security model based on SACS, but the description of model is over simple, and has not point out how to protect the security of cloud services precisely. The paper [4] described how to build the trust model of cloud services, and decided if choose the service by the credibility of this service, thus protect the benefit of cloud users, however, this paper ignored the threaten of cloud users, malicious users maybe destroy cloud services. In this paper, risk prediction model based on improved Adaboost method is proposed for solving these above problems.

2. FRAME

The risk prediction of cloud users can be divided into two classes: dynamic prediction and static prediction. Static prediction analysis the historical data of cloud users and get a priori model, thereby improve the ability of predicting new users' behavior. Static analysis the attributes of users, the key of the technology lies in how to make full use of the historical data, and construct the prediction model.

Risk prediction of cloud users is a two-class classification problem, SVM [5-7], Decision Tree [8-10], Neural Network [11-13], Naive Bayes [14-16], etc, all can regard as the classification methods. However, these traditional classification methods have many disadvantages. Such as, in the model of risk prediction of cloud users, the cost of risky user classifying as risk-free user is much more than the cost of risk-free user classifying as risky user. Many of existing method shave not considered the unequal properties of cost. And the classification effect of a single classifier is limited, how to improve the classification effect is also a big problem.

Aiming at the above problem, we use AdaBoost method in multi-classifiers, which combine a few weak classifiers

1874-110X/15 2015 Bentham Open

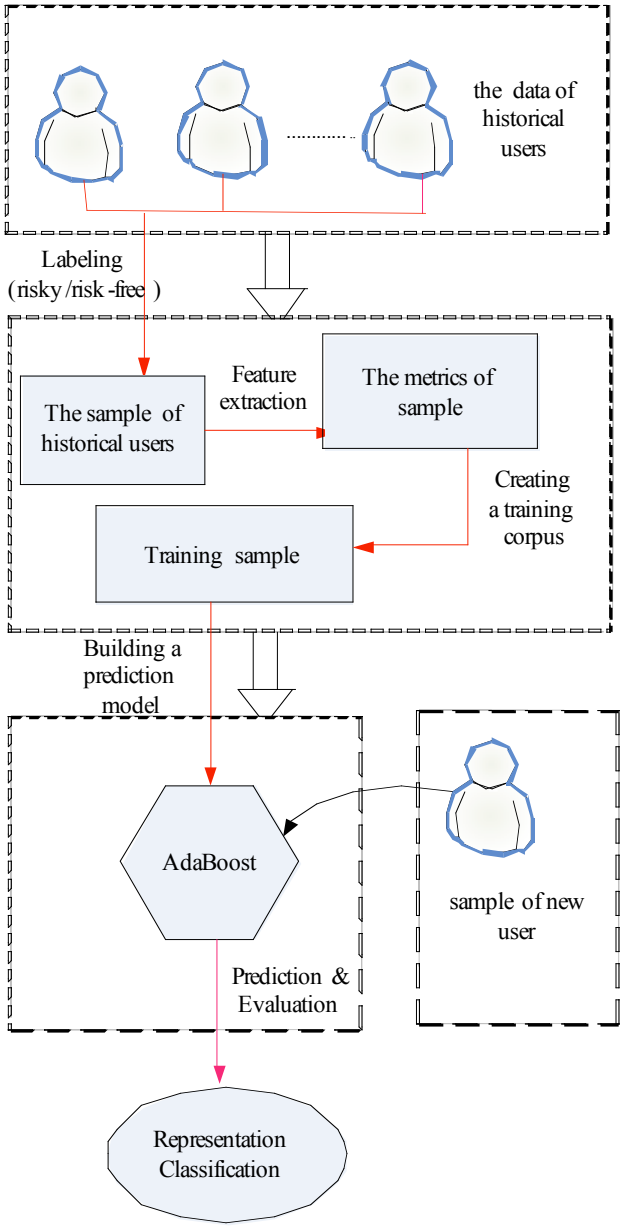

Fig. (1). A frame of prediction model.

and get a stronger classifier. Also, we achieve a better prediction result through introducing cost-sensitive factor. The process is shown in Fig. (**1**).

Step 1: Making authentication for each user, who applies for cloud service.

Step 2: Predicting the risky of users, who have passed the authentication; Otherwise, forcing the users out of the service request.

Step 3: Providing requested service for users, who have passed the risky prediction. Otherwise, refuse to provide the requested service.

Among them, the risk prediction of step 2 is the emphasis**,** the steps are as follow:

Step 2.1: Collecting the information of historical users, $S = (S_1, S_2, ..., S_n)$, S_i expresses the information of users.

Step 2.2: Extracting feature from the information of historical user S_i, and get the feature vector $X_n = (x_1, x_2, ..., x_m)$, among them, $x_1, x_2, ..., x_m$ express the cloud user's attributes, such as abnormal behavior, integrity degree, behavioral environment. etc; Mark each user as risky or risk-free, get the Label vector Y_n.

$Y_n = \{-1\}$ or $Y_n = \{1\}$.

Step 2.3: Through the cost-sensitive AdaBoost method, training sample (X_n, Y_n), which is got in the first two steps, then, get a strong classifier.

Step 2.4: Classifing the new user through this strong classifier, and predict the risk of new user.

3. THE ADABOOST METHOD BASED ON COST SENSITIVE

AdaBoost method as a common integrated learning method, this method adjusts the sample distribution of the next training through the error rate of the last training, make the next training has stronger ability of identification for the error samples, and reset a weight for each weak classifier. Last, through the means of weighted voting to generate the strong classifier, thus improve the overall results of classification.

Meanwhile, during the process of risk prediction of cloud users, we will meet two types of wrongly-predicted cost。 The error I is risk-free user classifying as risky user, and the error II is risky user classifying as risk-free user. Obviously, the cost of error I is higher than the cost of error II。 Naturally, we consider cost-sensitive factor into the AdaBoost method, and consider these two kinds of the wrongly-predicted cost into the structure of the classifier.

3.1. AdaBoost Method

AdaBoost method was proposed by Freund and Schapire [17]. This method uses a basic classifier to generate a series of classifiers, and each classifier is resetted a corresponding weight. At each final training stage, we adjust the weights of training sample to influence the next training. We improve the weight of wrongly-predicted sample, and reduce the weight of correctly-predicted sample, these adjustments of weights make the classifier has stronger ability to identify wrongly-predicted sample. Finally, each classifier are combined through specific ways. Method process is as follow.

Algorithm 1. AdaBoost method.

Input: A training set G containing N samples (X_n, Y_n), $n = 1, 2, ..., N$ where X_n is a vector of attribute values of cloud users, $X_n = (x_1, x_2, ..., x_m)$, $Y_n \in Y = \{-1, 1\}$ is the class label.

Initialization: Let the weight vector : $W_1(n) = 1/N$ for $n = 1, 2, ..., N$, $y \in Y - \{Y_n\}$

Do for $t = 1, 2, ..., T$

1. Train neural network using the weight distribution W_t and obtain the

hypothesis $h_t \in [-1,1]$.

2. Calculate the error of h_t :

$$\varepsilon_t = \sum_{n \subset h_t(X_n) \neq Y_n} W_t(\mathrm{n}) \quad (1)$$

If $\varepsilon_t > 0.5$, set $T = t-1$ and abort loop.

3. Set the weight updating parameter

$$\alpha_t = \frac{1}{2}\log(\frac{1-\varepsilon_t}{\varepsilon_t}) \quad (2)$$

4. Update and normalized the weight vector

$$W_{t+1}(n) = \frac{W_t(n)\exp(-\alpha_t h_t(X_n)Y_n)}{Z_t} \quad (3)$$

Where $n = 1,2,...,N$, Z_t is a normalization factor chosen so that $W_{t+1}(n)$ becomes a proper distribution.

Output:the final classifier

$$h_f(x) = sign(\sum_{t=1}^{T}\alpha_t h_t(x)) \quad (4)$$

Among them, X_n is the attribute vector of cloud user, and $X_n = (x_1, x_2, ..., x_m)$, Y_n is sample label (risky or risk-free), $W_k(\mathrm{n})$ denotes the weight of the k-th training phase of the n-th sample.

3.2. Cost-Sensitive

Cost-sensitive has been widespread concerned in recent years [18]. The target of cost-sensitive is to consider diffident wrongly-predicted cost into the process of structuring classifier. For the problem of two-class classification of cloud users risk prediction, cost matrix is as shown in Table **1**. In the Table **1**, $C(\mathrm{i},\mathrm{j})(\mathrm{i},\mathrm{j} \in \{-1,1\})$ show the error cost that class i is classified as class j. In our method, $C(-1,1)$ show the cost that risky user is classified as risk-free user, and $C(1,-1)$ show the cost that risk-free user is classified as risky user. The global target is to consider the cost-sensitive matrix into the structure of classifier, thus, get the minimal cost classifier.

Considering the different wrongly-predicted cost, we need to add cost matrix into AdaBoost method.

As the above description of AdaBoost method, we initialize the each sample with same distribution. After adding cost-sensitive, a common method is to set different wrongly-predicted cost sample with different weight, for example, the higher wrongly-predicted cost sample with higher weight, and the lower wrongly-predicted cost sample with lower weight. Such as Schapire, Singer and Singhal [19], they all adopted this method. Karakoulas and Shawe-Taylor [20] also adopted the similar method.

There is another method, which is adding cost-sensitive factor into the process of updating weight [21]. Through this way, we can make the weight updating speed increasing faster for the high-cost sample if it is wrongly predicted, or make the weight updating speed reducing more slowly for the high-cost sample if it is correctly predicted. This method has been proved to achieve lower entirety wrongly-predicted cost. Method process is as follows.

Algorithm 2. Cost-sensitive AdaBoost method.

Input: A training set G containing N samples (X_n, Y_n) , $n = 1,2,...,N$ where X_n is a vector of attribute values of cloud users, $X_n = (x_1, x_2, ..., x_m)$, $Y_n \in Y = \{-1,1\}$ is the class label. And $C_n = (\mathrm{c}_1, \mathrm{c}_2, ..., \mathrm{c}_m)$.

Initialization: Let the weight vector : $W_1(\mathrm{n}) = c_i / \sum_j^m c_j$ for $n = 1,2,...,N$, $y \in Y - \{\mathrm{Y}_n\}$

Do for $t = 1,2,...,T$

1. Train neural network using the weight distribution W_t and obtain the hypothesis $h_t \in [-1,1]$.

2. Calculate the error of h_t :use equal (1)

If $\varepsilon_t > 0.5$, set $T = t-1$ and abort loop.

3. Set the weight updating parameter: use equal (2)

4. Update

$$W_{t+1}(i) = \frac{W_t(i)\exp(-\alpha_t y_i h_t(x_i)\beta(\mathrm{i}))}{Z_t} \quad (5)$$

Where $\beta(i)$ is a cost-adjustment function.

$$\beta(i) = \beta(sign(y_i h_t(x_i)), c_i) \quad (6)$$

Where $n = 1,2,...,N$, Z_t is a normalization factor chosen so that $W_{t+1}(n)$ becomes a proper distribution.

Output: the final hypothesis:

$$h_f(x) = sign(\sum_{t=1}^{T}\alpha_t h_t(x)) \quad (7)$$

We modify the updated method of weights, and obtain a simple method, as follows:

$$W_{t+1}(i) = \frac{\beta(\mathrm{i})\,\mathrm{W}_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t} \quad (8)$$

Table 1. The cost matrix of risk prediction.

	Actual Defect-Prone	**Actual Not-Defect-Prone**
Predict defect-prone	$C(-1,-1) = c_{-1,-1}$	$C(1,-1) = c_{1,-1}$
Actual not-defect-prone	$C(-1,1) = c_{-1,1}$	$C(1,1) = c_{1,1}$

In this method, the cost-sensitive factor is placed out of the formula and change the updating speed of different wrongly-predicted sample. In this paper, we adopt the second method.

4. EXPERIMENT

In this section, we will describe the experimental data sets, measurement metric and the design of experiment detailedly.

4.1. Dataset

As the static data of cloud user is difficult to collect, and there is no open experimental data sets, we chose to use the data set in similar environments, such as bank credit risk prediction data set. In this experiment, we use bank credit users of UCI datasets to provide simulation. It is a collection of data for the Australian credit certification. Due to the privacy issue, get 14 dimensions of user attributes data (http://archive.ics.uci.edu/ml/datasets/Statlog+%28Australian+Credit+Approval%29).

4.2. Measurement Metric

There are four metrics to evaluate the effectiveness of prediction models: recall rate, false alarm rate, precision and accuracy. we use A, B, C, D to define these four metrics as shown in Table **2**, where A, B,C, D, respectively, is the number of risky users being predicted as risky ones, being predicted as risk-free ones, and the number of risk-free users being predicted as risky users, risk-free users being predicted as risk-free ones.

Table 2. Defect prediction metric.

	Predict as Risky Users	Predict as Risk-Free Users
Risky users	A	B
Risk-free	C	D

Recall: It is the proportion of risky users being accurately predicted as risky users, which is $A/(A+B)$. This ratio is very important for cloud-users risk prediction. The purpose of the model is to identify risk cloud-users. We use Pd to express recall.

False positive rate: The proportion of risky users being predicted as risk-free, which is $C/(C+D)$. We use Pf to express false positives.

Precision: The proportion of risk-free users being correctly predicted as risk-free, which is $A/(A+C)$. We use pre to express precision.

A good method should have a high recall and precision. However, this two are mutual restraint. So we need a combination of the two metrics. F-measure is the harmonic mean of recall and precision, which is defined as follows:

$$F-measure = 2*recall*precision/\left(recall\ +\ precision\right) \quad (9)$$

Obviously, a good method should have a high recall rate, precision, F-measure, and low false alarm rate.

In addition, for some of these methods, such as neural networks, SVM, etc., we can also draw their ROC curves, to compare its effect.

We use recall rate as X-axis, false positives as Y-axis rate of, and obtain a classifier ROC space, and the ROC curve can be drawn. Assuming the output of a classifier is the confidence of a sample belongs to positive class, then the area under the ROC curve represents, for any pair of samples, the probability that the confidence of a positive sample is greater than that of the negative samples, which is AUC value. By calculating the AUC value, we can have a good comparison measure for classifiers.

4.3. Experiment Setting

To assess the effect of the proposed method, we have done a lot of experiments. We initialize all samples as randomly distributed, take half of the sample as training set, and the other half as test set. we only take the average of the 20 random sampling experiments and analysis the experimental results in Section 4.4. we set the cost-sensitive factor as $cost(1,2):cost(2,1)=1:5$ and achieve the best result.

4.4. Results Analysis

We compared several common classification methods, including support vector machine (SVM), C4.5 decision tree, naive Bayes, neural networks, etc.

Considering the background of cloud-user risk prediction, it is a critical task to predict risky-user correctly. So

Table 3. Experimental results: Pd, Pf, and Pre comparisons in five method.

Dataset	M	SVM	C 4.5	Bays	network
None	Pd	0.88	0.82	0.84	0.78
	Pf	0.16	0.16	0.13	0.16
	Pre	0.81	0.80	0.84	0.79
C-Adaboot	Pd	0.88	0.85	0.87	0.89
	Pf	0.10	0.17	0.15	0.15
	Pre	0.88	0.80	0.83	0.83

Table 4. Experimental results: F-measure comparisons in five method.

F-measure	SVM	C4.5	Bays	Network
None	0.84	0.81	0.85	0.78
C-AdaBoost	0.88	0.83	0.87	0.86

Table 5. AUC value of 15 weak classifiers in cost-sensitive AdaBoost neural network.

K	1	2	3	4	5	6	7	Strong Classifier
AUC	0.89	0.88	0.86	0.84	0.87	0.86	0.85	0.92
	0.80	0.88	0.86	0.87	0.87	0.87	0.82	
K	8	9	10	11	12	13	14	

recall rate(pd) is a very important indicator. Observed on the Table **3**, we can see that through the usage of C-AdaBoost each method, the average pd value has a conspicuous increase at an average of 5.4%.

Table **4** shows the comparison of F-measure between our method and other methods, which is average of 20 randomized trials.

From theabove table we can see, through cost-sensitive AdaBoost method, the general classifier effect has been significantly improved, with an average increase of 0.23 (0.1-0.4);

We calculate the AUC value of each weak classifier and the final strong classifier in AdaBoost neural networks. The result is showed below:

In the above Table **5**, we can see that a strong classifier being generated by combining weak classifiers. The final AUC value is greater than all other weak classifiers. Our approach improves the average AUC of 5% (2% -12%).

CONCLUSION

The risk prediction of cloud users can be divided into two classes: dynamic prediction and static prediction. Static prediction mine the historical data of cloud users and get a priori model, thus achieve the risk prediction of the new cloud users. This paper, we treat the risk prediction of cloud users as a two-class classification problem.

Adaboost method is widely applied in various fields. But, as we know, it is applied to risk prediction for the first time. Many weak classifiers are aggregated into a strong classifier by Adaboost method, then, we get strong classifier. As to the problem that the uneven wrongly-predicted cost of the prediction, we introduce the cost-sensitive factor into our method, thereby, improving the predictive ability of classifier for higher wrongly-predicted cost. Through the comparison with some traditional classifiers and our experiment, we can see the better result of our method-cost-sensitive Adaboost method. On the UCI data sets, our method improves the average value of pd and F-measure effectively. Through the comparison of AUC value, we can obviously make out that AdaBoost integrates weak classifiers continually and get the final strong classifier, the strong classifier improve the classification results effectively.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by The National Natural Science Foundation of China (61170065, 61373017, 61171053, 61103195, 61203217, 61201163, 61202004, 61202354, 61402241); The Natural Science Foundation of Jiangsu Province (BK2012436); Scientific & Technological Support Project (Industry) of Jiangsu Province (BE2012183, BE2012755); Natural Science Key Fund for Colleges and Universities of Jiangsu Province (11KJA520001, 12KJA520002); The Natural Science Fund for Colleges and Universities of Jiangsu Province (13KJB520017); The Science Foundation of NJUPT (NY213155);Scientific Research & Industry Promotion Project for Higher Education Institutions (JHB2012-7).

REFERENCES

[1] D. Nurmi, R. Wolski, C. Grzegorczyk, and G. Obertelli, "The eucalyptus open-source cloud-computing system//Cluster Computing and the Grid", In: *CCGRID'09. 9th IEEE/ACM International Symposium on IEEE*, pp. 124-131, 2009.

[2] A. SIrisha, and G. Geethakumari, "API access control in cloud using the role based access control model", *Trendz in Information Sciences and Computing-TISC*, pp. 135-137, 2010.

[3] J. Xue, and J. J. Zhang, "A brief survey on the security model of cloud computing", *Ninth International Symposium on Distributed Computing and Applications to Business Engineering and Science (DCABES)*, 2010.

[4] L. Zhang, J. Liu, R. Wang, H. Wang, "Trust evaluation model based on improved D-S evidence theory, *Journal on Communications*, vol. 34, no. 7, pp. 167-173, 2013.

[5] K. Elish, and M. Elish, "Predicting defect-prone software modules using support vector machines," *Journal Systems and Software*, vol. 81, no. 5, pp. 649-660, 2008.

[6] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Using the support vector machine as a classification method for software defect prediction with static code metrics," *Engineering Applications of Neural Networks*, vol. 43, pp. 223-234, 2009.

[7] Z. Yan, X. Y. Chen, and P. Guo, "Software defect prediction using

fuzzy support vector regression," *Advances in Neural Networks*, pp. 17-24, 2010.

[8] J. Wang, B. J. Shen, and Y. T. Chen, "Compressed C4.5 Models for Software Defect Prediction," In: *Int. Conf. Quality Software*, pp. 13-16, 2012.

[9] T. M. Khoshgoftaar, and N. Seliya, "Tree-based software quality esti-mation models for fault prediction," *IEEE Symp. Software Metrics*, pp. 203-214, 2002.

[10] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[11] N. Gayatri, S. Nickolas, and A.V. Reddy, "Feature Selection Using Decision Tree Induction in Class level Metrics Dataset for Software Defect Predictions," *the proceeding of World Congress on Engineering and Computer Science*, vol. 1, pp. 124-129, 2010.

[12] M. M. T. Thwin, and T. S. Quah, "Application of neural networks for software quality prediction using object-oriented metrics," *Journal of Systems and Software*, vol. 76, no. 2, pp. 147-156, 2005.

[13] E. Paikari, M. M. Richter, and G. Ruhe, "Defect prediction using case-based reasoning: An attribute weighting technique based upon sensitivity analysis in neural networks," *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 5, pp. 2012.

[14] T. Wang and W. Li, "Naïve Bayes Software Defect Prediction Model," *Int. Conf. Computational Intelligence and Software Engineering*, pp. 1-4, 2010.

[15] S. Amasaki, Y. Takagi, O. Mizuno, and T. Kikuno, "A Bayesian Belief Network for Assessing the Likelihood of Fault Content," *Int. Symp. Software Reliability Engineering*, pp. 215-226, 2003.

[16] B. Turhan, and A. Bener, "Software Defect Prediction: Heuristics for Weighted Naïve Bayes," *Int. Conf. Software and Data Technologies*, pp. 244-249, 2007.

[17] Y. Freund and R. Schapire, "A decisiontheoretic generalization of on-line learning and an application to boosting", *Journal of Computer and SystemSciences*, vol. 55, no. 1, pp. 119-139, 1997.

[18] P. Chan and S. Stolfo, "Towards Scalable Learning with Non-Uniform Class and Cost Distributions: ACase Study in Credit Card Fraud Detection" In *Proc. Fourth Int'l Conf on Knowledge Discovery and DataMining*, New York, pp. 164-168, 1998.

[19] R. Schapire, Y. Singer and A. Singhal, "Boosting and Rochhio applied to text filtering", In *SIGIR*, 1998.

[20] G. Karakoulas, and J. Shawe-Taylor, "Optimizing classifiers for imbalanced training sets", *Adv neural process syst*, proceeding of the 1998 conference, MIT press, 1999, vol. 11, p.253.

[21] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: misclassification cost-sensitive boosting", *ICML*, pp. 97-105, 1999.

Received: September 22, 2014 Revised: November 30, 2014 Accepted: December 02, 2014