Search for Optimized Cost matrix for Performance Enhancement of Anomaly Based Intrusion Detection System using Cost Sensitive Classifier

Manasi Gyanchandani[1] , J.L.Rana[2], and R.N.Yadav[3]

[1]Department of Computer Science and Engineering, MANIT, Bhopal, India
[2] Ex-HOD, Dept. of CS/IT, MANIT, Bhopal, India
[3] Dept of Electronics and Communication MANIT Bhopal

ABSTRACT

Intrusion Detection Systems (IDSs) should maximize security while minimize cost. Classic evaluation measures have been extensively studied in the past. Recently, cost-sensitive classification has received much attention. A cost-sensitive classifier uses cost values to evaluate the performance of the classifier. However, these cost values must be given in advance and are generally unknown for a given dataset. It is very time consuming to find these cost values. Again if it is possible to find out such cost values same cannot be used for other datasets. In a typical classification task, all types of misclassifications are treated equally. However, in many practical cases, not all misclassifications are equal. Therefore, it is critical to use a cost-sensitive classifier to minimize cost of misclassifications. This work uses MetaCost, a cost-sensitive meta-classifier that takes in a classification algorithm, training data, and a cost matrix. In order for MetaCost to be effective, we need to find an optimal cost matrix. In this paper we have proposed a new optimization technique for choosing the cost matrix: cost matrix optimization technique for Anomaly Based Intrusion Detection System (ABIDS). This approach can be applied for finding out optimized cost matrix for any datasets.

KEYWORDS

Intrusion Detection System, Anomaly based Intrusion Detection System, classification algorithm, accuracy, confusion matrix, cost matrix.

1. INTRODUCTION

Performance evaluation plays an important role in the field of intrusion detection and classification process. Classification is an important aspect of machine learning and computer science in general [1, 2]. It determines what class a particular instance belongs to based on prior knowledge. An instance consists of set of attributes that are meant to define it and are used to determine its label (or class). Given a set of instances and their attributes and labels, collectively referred to as dataset, a classification algorithm is trained to create a classification model (using training data). This classification model consists of a set of rules that allows it to determine what class a set of attributes belongs to. To determine the effectiveness of the classification model, it is then tested on a set of test data, typically data that was withheld from training [3]. As we are dealing with ABIDS we have taken all unknown instances in the test data (novelty data). The attributes are given to the model and the subsequent generated label is compared to the known label in order to determine its performance. The most basic performance metric to be used is accuracy, in which a percentage is given based on how many instances were correctly labeled.

In a typical classification task, all types of misclassifications are treated equally. In many practical cases, misclassifications are equal. Therefore, it is critical to use a cost-sensitive classifier to minimize these misclassifications. The goal of cost-sensitive classification is to build a classifier to minimize the expected misclassification costs rather than to minimize the expected number of misclassification errors.

Section 2 and section 3 discusses the description of cost matrix and based on confusion matrix, how to choose the values of cost matrix. Section 4 discusses the confusion matrix. Section 5 covers the performance parameters used in this paper. Section 6 discusses the MetaCost algorithm, a cost-sensitive meta-classifier. In section 7, proposed cost optimization technique id discussed. Simulation results are discussed on section 8.

2. COST MATRIX

A cost matrix is a mechanism used for decision making of a model. It is always domain-dependent. Depending on the application, the defined costs can be very different. It can cause the model to minimize costly misclassifications and maximize beneficial accurate classifications. For example, if a model classifies a customer with poor credit as low risk, this error is costly. A cost matrix could bias the model to avoid this type of error. The cost matrix might also be used to bias the model in favour of the correct classification of customers who have the worst credit history. Performance of the classifier can be measured by assigning a cost for misclassifications and a "reward" for correct classifications. Cost matrix for two class is defined as a C (i: j) where i is the actual class and j is the predicted class, for example:

C (i: j) =

C (1, 1)	C (1, 2)
C (2, 1)	C (2, 2)

Here, C (1, 1) means that classifying a positive as a positive (true positive) is considered a reward and so also C (2, 2) means classifying negative as negative (true negative). The cost for false negatives C (1, 2) is severe and is defined as misclassification 1 , and a cost for false positives C (2, 1) is considered as an moderate and is defined as misclassification 2 [4].There are many applications where we can use this cost matrix and see the difference of misclassification 1 and misclassification 2. For examples:

- In loan applications, the cost of rejecting an applicant who will not pay back is minimal as compared to accepting an applicant who will not pay back. While accepting an applicant who will pay back is a gain and accepting an applicant who will not pay back is a big loss.
- In Spam-Mail Filtering rejecting good E-mails (ham) is much worse than accepting a few spam mails
- In database marketing the cost of mailing to a non-respondent person is very small, as compared to the cost of not mailing to respondent person.
- In many real-world applications such as medical diagnosis misclassifying a patient as healthy (i.e., false negative) is more costly than misclassifying a normal person as patient.

3. CHOICE OF THE COST MATRIX

As an example, consider a cost matrix C for two classes, normal and attack. The cost of a false positive is C (2, 1), while that of a false negative is C (1, 2) and we can set C (1, 1) = C (2, 2) = 0, i.e. a correct classification will have no cost or there is an assumption that the correct classification has no cost, this assumption is according to a corollary given in [5]: "a cost matrix can always be transformed into an equivalent matrix with zero values on the diagonal". For intrusion detection applications, it is common to refer to attacks as positive and normal instances as negative example. Based on the confusion matrix, a cost matrix C allows us to punish misclassifications and reward correct classifications. So C (1, 2) and C (2, 1) represent punishments for misclassifications and C (1, 1) and C (2, 2) represent rewards for correct classifications. To differentiate between rewards and punishments we use positive values for punishments and negative values for rewards. This cost matrix is then used to alter the learning of a classification algorithm according to these rewards/punishments. This cost matrix adjusts the learning of the classification algorithm to classify the instances.

4. CONFUSION MATRIX

In the two-class case with classes positive and negative, a single prediction has the four different possible outcomes as shown in Table 1. The true positives (TP) and true negatives (TN) are correct classifications. A false positive (FP) occurs when the outcome is incorrectly predicted as positive when it is actually negative. A false negative (FN) occurs when the outcome is incorrectly predicted as negative when it is actually positive [2]

Table 1: Different outcomes of a two-class prediction [2]

	Predicted Class		
Actual Class		Normal	Attack
	Normal	TP	FN
	Attack	FP	TN

If the costs are known, they can be incorporated into a financial analysis of the decision-making process. In the two-class case, in which the confusion matrix is like that of Table 1, the two kinds of misclassification (error): false positives and false negatives will have different costs; similarly, the two types of correct classification may have different benefits. In the two-class case, costs can be summarized in the form of a 2 X 2 matrix where the diagonal elements represent the two types of correct classification and the off-diagonal elements represent the two types of misclassifications. In the multiclass case this generalizes to a square matrix whose size is the number of classes, and again the diagonal elements represent the cost of correct classification.

5. PERFORMANCE COMPARISON MEASURES

Metrics which are mainly used to evaluate the performance of classifier are present in [6] [2] and are given here for ready reference.

- Accuracy: Percentage of correctly classified examples which can be calculated as below:

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

- Cost Per Example: Given the cost matrix for two class and the confusion matrix obtained during training and testing process, cost per example (CPE) is calculated using the formula:

$$CPE = \frac{1}{N} \sum_{i=0}^{2} \sum_{j=0}^{2} CM(i, j) * C(i, j)$$

where CM corresponds to confusion matrix, C corresponds to the cost matrix, N represents the number of samples in training and testing phase patterns tested, i and j are the dimensions of the matrix.

6. METACOST

MetaCost developed by Pedro Domingos [7], is a method for making a classifier cost-sensitive. The basic idea of MetaCost is to take a classifier and adjust the learning with a cost matrix. The following are the steps followed by MetaCost procedure:-

1. The first step is to take the training data and create multiple bootstrap samples of the data. These bootstrap samples are then used for training to create an ensemble of classifiers.
2. The ensemble of classifiers are then combined through a majority vote to determine the probability of each data object x belonging to each class label.
3. Next, each data object in the training data is relabeled based on the evaluation of a conditional risk function as shown in equation 1, and a final classifier is then produced after applying the classification algorithm to the relabeled training data.

$$R(i/x) = \sum_j P(j|x)C(i,j) \quad \ldots\ldots\ldots\ldots (1)$$

Here R (i/x): conditional risk defining the cost of predicting that data object x belongs to class label i instead of class label j. The complete algorithm is shown in the Table 2.

Table 2: The MetaCost algorithm

S is the training set.
L is the classification learning algorithm.
C is a cost matrix.
m is the number of resamples to generate.
n is the number of examples in each resample.

1) For i in range 1 to m
   Create S_i as a resample of S with n examples.
   Create model M_i by applying L to S_i.

2) For each example x in S
   (a)For each class j
   Create $P(j|x) = \frac{1}{\sum_i 1} \sum_i P(j|x, Mi)$
   (b)Change the the class of x to the class k that minimize $\sum_j P(j|x)C(k,j)$

3) Create final model M by applying L to S.

7. COST OPTIMIZATION TECHNIQUE:

A cost-sensitive classifier uses cost values to alter the learning of the classifier. These cost values must be given in advance and are generally unknown for a given dataset. This is usually solved by experimenting with many cost values, which is very time-consuming. Again if we have obtained an ideal cost values for a particular algorithm it can't be transferable to a different dataset. Here we have used Metacost, an cost sensitive classifier for experimentation. This algorithm uses cost values to punish a classifier for misclassifying an instance. In [8] researchers have shown that it is better to punish misclassification of both class members (normal and attack). While in [4, 9] researchers states that in order to produce good results punishment should be given only to misclassification of the attack class. The problem is to find the optimal cost matrix computationally. An exhaustive search of all possible cost matrices will guarantee optimal cost matrices. We propose an simple cost optimization technique to find a "best" cost matrix. In [3] the author has applied cost optimization technique for Solving Rare-Class Biological Problems. Here we have applied the optimization technique to find cost matrix for ABIDS. The basic concept of cost optimization in as shown in the Figure 1.

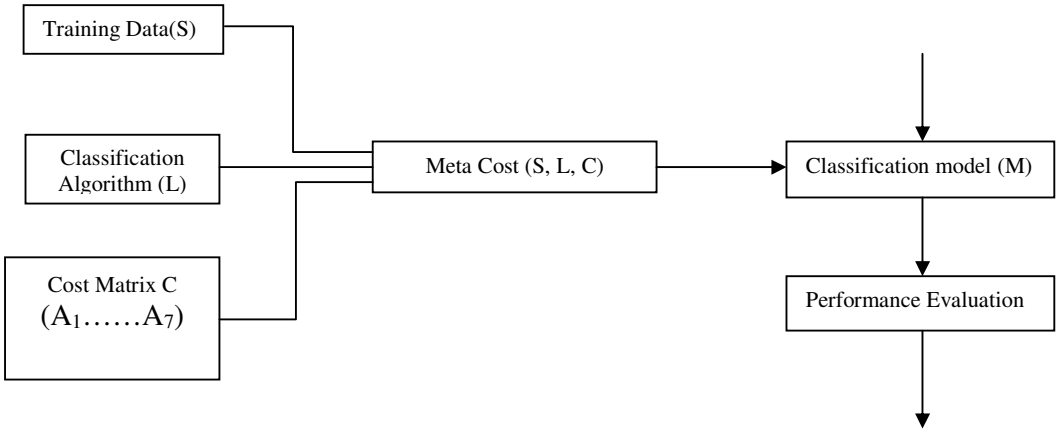

Figure 1: Basic Concept of Search Process

This technique is a very simple and is same as a greedy search method. It takes a current solution to a problem, generate new solutions based on this solution, and then replace the current solution with a new, if it is better. If no new, better solution can be found, the process is halted. The complete process of cost optimization is shown in Figure 2 and the algorithm is as shown in the Table 3. Here we are keeping track of two solutions: a current best and an overall best. The basic idea of this method is to start with an initial cost matrix and to increment its costs to find a cost matrix that achieves a better accuracy, reducing false alarm rate and CPE. An initial cost matrix is typically a cost matrix that will create a model that is the same as the model created by a classifier. Starting with this initial cost matrix, the method creates seven new ones. Each of these cost matrices represents a different combination of incrementing/decrementing the costs (correct classifications are decremented by ten and misclassifications are incremented by ten). Step size of ten is chosen as this value shows an appropriate change in performance metrics. Here we have not adjusted the cost value of correct classifications of the negative class (TN) and left at zero. This is due to the fact that typically a poor classifier will order most (if not all) instances as belonging to the correct classification. Therefore, there is no need to "reward such behavior" and our method has fewer cost matrices to test. After creating these seven new cost matrices, each one is used to create a new model through MetaCost, using the given training data S and classification algorithm L. After these models have been created, they are evaluated on the given test set T using the

evaluation function Acc(M, T). The model that has the highest accuracy value is kept and its cost matrix is used to initialize the next iteration of cost matrix creation.

8. SIMULATION RESULTS AND DISCUSSION

Almost all the experiments of Intrusion Detection are done on KDD 1999 dataset [10] the benchmark data set used for research. In our experiments, we have used 10% training data consisting of 4, 94,021 connection records for training and 79,366 records for testing.The editing (required to create novelty dataset) over the data set is done using the tool termed as 'notepad++'. The implementation of different types of algorithm is performed using 'WEKA 3.7'. J48 is the base classifier used for MetaCost.

Initializations: The initial cost matrix to start with is considered as a matrix with no reward is given for correct classifications and no punishments for misclassification i.e. matrix with all zero values. As expected, the results are the same as the basic classifier with the value of accuracy as 76.389%. So both the Overall Best Model (M_O) and Current Best Model (M_C) were set to this new model.

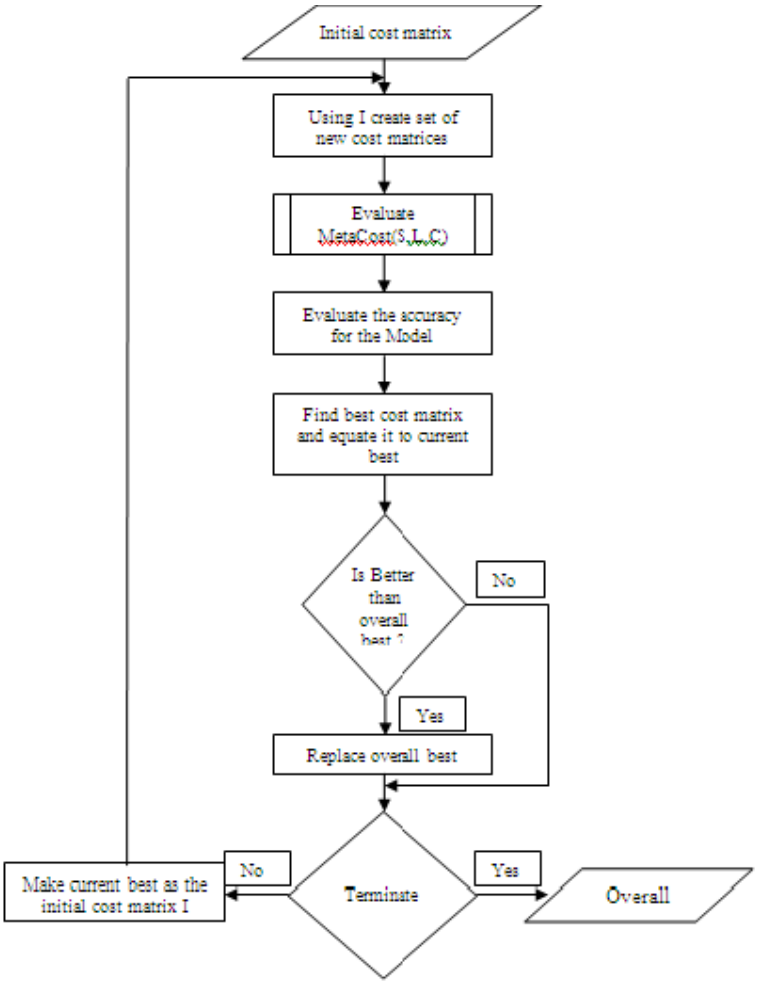

Figure 2: Cost Matrix Optimization Technique

Table 3: Algorithm for Cost Matrix Optimization Technique

Input:

S is the training set.
T is the test set.
L is a classification algorithm.
n is the number of iterations to run the algorithm.
Let Acc (M, T) return an accuracy of how Model M performed on test set T.

Procedure Cost Optimization (S, T, L, n)

I: initial cost matrix where all correctly classifications and misclassifications are zero.
C: current best cost matrix, initialized to I.
M_C : current best model, initialized to MetaCost (S, L,C)
O: overall best cost matrix, initialized to I.
M_O : overall best model, initialized to M_C.

For i = 1 to n do
Let A be a set of cost matrices

$A_1 = C + \begin{bmatrix} 0 & 10 \\ 0 & 0 \end{bmatrix}$ $A_2 = C + \begin{bmatrix} 0 & 0 \\ 10 & 0 \end{bmatrix}$

$A_3 = C + \begin{bmatrix} -10 & 0 \\ 0 & 0 \end{bmatrix}$ $A_4 = C + \begin{bmatrix} 0 & 10 \\ 10 & 0 \end{bmatrix}$

$A_5 = C + \begin{bmatrix} -10 & 10 \\ 0 & 0 \end{bmatrix}$ $A_6 = C + \begin{bmatrix} -10 & 0 \\ 10 & 0 \end{bmatrix}$

$A_7 = C + \begin{bmatrix} -10 & 10 \\ 10 & 0 \end{bmatrix}$

Set C and M_C to Null
for j = 1 to 7 do
M = MetaCost(S,L,A_j)
if Acc(M,T) > Acc(M_C,T) then
C = A_j
M_C = M
end if
end for
if Acc (M_C,T) > Acc(M_O,T) then
O = C
M_O = M_C
end if
end for
return O, M_O

Iteration 1: Using this initial cost matrix, we then created seven new cost matrices as detailed in Table 3. These were in turn used to create seven new models, which were then compared based on their generated accuracy. The best results were achieved through the two cost matrix-

$A_4 = \begin{bmatrix} 0 & 10 \\ 10 & 0 \end{bmatrix}$ and $A_6 = \begin{bmatrix} -10 & 0 \\ 10 & 0 \end{bmatrix}$

The accuracy obtained for both of the cost matrix was 77.8737%. But value of CPE for A_4 cost matrix was found to be more (2.2126) as compared to A_6 (-5.426). The Overall Best Model (M_O) and Current Best Model (M_C) were set to this new model i.e. model obtained by applying A_4 and A_6 cost matrix.

Iteration 2: With the initial cost matrix as A_4 and A_6 again seven new cost matrices each were generated. For A_4 the seven new matrix generated will be given by $A_{i,j}$ where i represents iteration number and j is the matrix number.

$$A_{21} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \quad A_{22} = \begin{bmatrix} 20 & 0 \\ -10 & 10 \end{bmatrix} \quad A_{23} = \begin{bmatrix} 10 & 0 \\ 0 & 20 \end{bmatrix}$$

$$A_{24} = \begin{bmatrix} 0 & 20 \\ 20 & 0 \end{bmatrix} \quad A_{25} = \begin{bmatrix} -10 & 20 \\ 10 & 0 \end{bmatrix} \quad A_{26} = \begin{bmatrix} -10 & 10 \\ 20 & 0 \end{bmatrix}$$

$$A_{27} = \begin{bmatrix} -10 & 20 \\ 20 & 0 \end{bmatrix}$$

It was found that A_{27} performed well with the accuracy of 78.0313%. While for A_6:

$$A_{21} = \begin{bmatrix} -10 & 10 \\ 10 & 0 \end{bmatrix} \quad A_{22} = \begin{bmatrix} -10 & 0 \\ 20 & 0 \end{bmatrix} \quad A_{23} = \begin{bmatrix} -20 & 0 \\ 10 & 0 \end{bmatrix}$$

$$A_{24} = \begin{bmatrix} -10 & 10 \\ 20 & 0 \end{bmatrix} \quad A_{25} = \begin{bmatrix} -20 & 10 \\ 10 & 0 \end{bmatrix} \quad A_{26} = \begin{bmatrix} -20 & 0 \\ 20 & 0 \end{bmatrix}$$

$$A_{27} = \begin{bmatrix} -20 & 10 \\ 20 & 0 \end{bmatrix}$$

It was found that AA_7 gave the best result with the accuracy of 78.0313%. Again Overall Best Model (M_O) and Current Best Model (M_C) were set to this new model i.e. model obtained by applying $A_4 \rightarrow A_{27}$ and $A_6 \rightarrow A_{27}$ cost matrix.

Iteration 3: Now with the initial cost matrix as A_{27} in both cases the seven cost matrix for each was generated. For $A_4 \rightarrow A_{27}$

$$A_{31} = \begin{bmatrix} -10 & 30 \\ 20 & 0 \end{bmatrix} \quad A_{32} = \begin{bmatrix} -10 & 20 \\ 30 & 0 \end{bmatrix} \quad A_{33} = \begin{bmatrix} -20 & 20 \\ 20 & 0 \end{bmatrix}$$

$$A_{34} = \begin{bmatrix} -10 & 30 \\ 30 & 0 \end{bmatrix} \quad A_{35} = \begin{bmatrix} -20 & 30 \\ 20 & 0 \end{bmatrix} \quad A_{36} = \begin{bmatrix} -20 & 20 \\ 30 & 0 \end{bmatrix}$$

$$A_{37} = \begin{bmatrix} -20 & 30 \\ 30 & 0 \end{bmatrix}$$

Again the matrix A_{37} gave the better results with the accuracy as 78.049%.

For $A_6 \rightarrow A_{27}$

$$A_{31} = \begin{bmatrix} -10 & 30 \\ 20 & 0 \end{bmatrix} \quad A_{32} = \begin{bmatrix} -10 & 20 \\ 30 & 0 \end{bmatrix} \quad A_{33} = \begin{bmatrix} -20 & 20 \\ 20 & 0 \end{bmatrix}$$

$A_{34} = \begin{bmatrix} -10 & 30 \\ 30 & 0 \end{bmatrix}$ $A_{35} = \begin{bmatrix} -20 & 30 \\ 20 & 0 \end{bmatrix}$ $A_{36} = \begin{bmatrix} -20 & 20 \\ 30 & 0 \end{bmatrix}$

$A_{37} = \begin{bmatrix} -20 & 30 \\ 30 & 0 \end{bmatrix}$

Here the matrix A_{37} gave the better results with the accuracy as 78.049%. So Overall Best Model (M_O) and Current Best Model (M_C) were set to this new model i.e. model obtained by applying $A_4 \rightarrow A_{27} \rightarrow A_{37}$ and $A_6 \rightarrow A_{27} \rightarrow A_{37}$ cost matrix.

Further iterations produced no improvement in the accuracy so the Overall Best Model (M_O) and Current Best Model (M_C) were set to the model obtained by applying $A_4 \rightarrow A_{27} \rightarrow A_{37}$ and $A_6 \rightarrow A_{27} \rightarrow A_{37}$ cost matrix.

9. CONCLUSION

In most of the cases where the cost of misclassification plays an important role, there just being aware of the cost of misclassifications is not enough. The use of cost-sensitive classifiers takes the data mining procedure much closer to what the application actually demands. MetaCost is a flexible model which is used in this kind of situation: one can use any classifier algorithm, MetaCost wraps around it and makes it cost-sensitive. Domingos [7] has stated that the MetaCost algorithm has a black box approach so that the user can easily adjust the learning of the training data through a cost matrix. With our search method, we are able to continue this black box approach, even giving on more advantage that the user does not even need to generate the cost matrices themselves. Our main aim is to find a cost matrix that is optimal for a given classification algorithm and a given set of data. For our experiments, we choose the accuracy as a performance measure because it results in an increase in true positives, without increasing the number of false positives too much.

REFERENCES

[1] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, "Introduction to Data Mining", Addison-Wesley, 2006.

[2] Ian H. Witten and Eibe Frank, "Data Mining: Practical Machine Learning Tools and Techniques", Elsevier, second edition, 2005.

[3] Mark J. Lawson, "The Search for a Cost Matrix to Solve Rare-Class Biological Problems", PhD Thesis, Department of Computer Science, VirginiaTech, Virginia, 2009.

[4] Charles Elkan, "The Foundations of Cost-Sensitive Learning", Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01), 2001.

[5] D. D. Margineantu, "Methods for Cost-Sensitive Learning", PhD Thesis, Oregon State University, 2001.

[6] M. V. Joshi, " On Evaluating Performance of Classifiers for Rare Classes", In Proceeding of the 2002 IEEE International conference on Data Mining, Maebashi City, Japan, December 2002.

[7] Pedro Domingos, "MetaCost: A general method for making classifiers cost-sensitive", Fifth International Conference on Knowledge Discovery and Data Mining, pp. 155-164, 1999.

[8] Mahesh V. Joshi, Vipin Kumar, and Ramesh C. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements", Proceedings of the First IEEE International Conference on Data Mining (ICDM'01), 2001.

[9] Yamnin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang, "Cost-sensitive boosting for classification of imbalanced data", Pattern Recognition, Vol. 40, pp. 3358-3378, 2007.

[10] WEKA software, Machine Learning, http://www.cs.waikato.ac.nz/ml/weka/, The University of Waikato, Hamilton, New Zealand.

Authors

Dr. Manasi Gyanchandani PhD from MANIT, Bhopal. M.E & B.E. in Computer Science & Engineering. She has around 19 years teaching experience. Guided 20 Mtech theses. Area of interest includes Artificial Intelligence, Neural Networks and Intrusion Detection Systems. Currently completed her Ph.D in the area of Anomaly Based Intrusion Detection systems. Working as an Assistant Professor at Maulana Azad National Institute of Technology, Bhopal since 2005.

Dr. J. L.Rana Professor and Ex-Head of Dept of MANIT Bhopal. PhD from IIT Mumbai MS from USA (Hawaii) Guided 20 Ph.Ds.

Dr. R.N.Yadav B.E. and Mtech from Motilal Nehru Regional Engineering College Allahabad, and Maulana Azad National Institute of Technology, Bhopal. Ph.D from IIT Kanpur. Working as an Professor in the Department of Electronics at Maulana Azad National Institute of Technology, Bhopal.