Multiple Classifier Prediction Improvements against Imbalanced Datasets through Added Synthetic Examples

Herna L. Viktor and Hongyu Guo

School of Information Technology and Engineering, University of Ottawa
800 King Edward Road, Ottawa, Ontario, Canada, K1N 6N5
{hlviktor,hguo028}@site.uottawa.ca

Abstract. Ensembles of classifiers have successfully been used to improve the overall predictive accuracy in many domains. In particular, the use of boosting which focuses on hard to learn examples, have application for difficult to learn problems. In a two-class imbalanced data set, the number of examples of the majority class is much higher than that of the minority class. This implies that, during training, the predictive accuracy against the minority class of a traditional boosting ensemble may be poor. This paper introduces an approach to address this shortcoming, through the generation of synthesis examples which are added to the original training set. In this way, the ensemble is able to focus not only on hard examples, but also on rare examples. The experimental results, when applying our Databoost-IM algorithm to eleven datasets, indicate that it surpasses a benchmarking individual classifier as well as a popular boosting method, when evaluated in terms of the *overall accuracy*, the *G-mean* and the *F-measures.*

1 Introduction

Over the past few years, ensembles have emerged as a promising technique with the ability to improve the performance of weak classification algorithms [1, 2]. Ensembles of classifiers consist of a set of individually trained classifiers whose predictions are combined to classify new instances [1, 2]. In particular, boosting is an ensemble method where the performance of weak classifiers is improved by focusing on hard examples which are difficult to classify. Boosting produces a series of classifiers and the outputs of these classifiers are combined using weighted voting in the final prediction of the model [3]. In each step of the series, the training examples are re-weighted and selected based on the performance of earlier classifiers in the training series. This produces a set of "easy" examples with low weights and a set of hard ones with high weights. During each of the iterations, boosting concentrates on classifying the hard examples correctly. Recent studies have indicated that boosting algorithm is applicable to a broad spectrum of problems with great success [3, 4].

The class imbalance problem corresponds to domains for which one class is represented by a large number of examples while the other is represented by only a few [5]. Many real world applications involve learning from imbalanced sets, such as fraud detection, telecommunications management, oil spill detection and text classification [6]. When learning from imbalanced data sets, machine learning algorithms tend to produce high predictive accuracy over the majority class, but poor predictive accuracy over the minority class [7]. There have been several proposals for coping with imbalanced data sets [5], including under-sampled examples of the majority class and/or over-sampling of the minority class [6, 7, 8] and weighing examples in an effort to bias the learning toward the minority class [7]. Research into the use of boosting includes [9], which evaluated boosting algorithms to classify rare classes; and [10] which combined boosting and synthetic data to improve the prediction of the minority class.

In this paper, we discuss the DataBoost-IM approach, which combines data generation and boosting procedures to improve the predictive accuracies of both the majority and minority classes, without forgoing one of the two classes. The aim of our approach is therefore to ensure that the resultant predictive accuracies of both classes are high. Our approach differs from prior work in the following ways. Firstly, we separately identify hard examples from, and generate synthetic examples for, the minority as well as the majority classes. Secondly, we generate synthetic examples with *bias* information toward the hard examples on which the next component classifier in the boosting procedures needs to focus. That is, we provide additional knowledge for the majority as well as the minority classes and thus prevent boosting over-emphasizing the hard examples. Thirdly, the class frequencies of the new training set are rebalanced to alleviate the learning algorithm's bias toward the majority class. Fourthly, the total weights of the different classes in the new training set are rebalanced to force the boosting algorithm to focus on not only the hard examples, but also the minority class examples. In this way, we focus on improving the predictions of both the minority and majority classes.

This paper is organized as follows. The performance measures used to evaluate the performance of our approach is introduced in Section 2. Section 3 describes the DataBoost-IM algorithm. This is followed, in Section 4, with an evaluation of the DataBoost-IM algorithm against eleven data sets from the UCI data set repository [11]. Finally, Section 5 concludes the paper.

2 Performance Measures

Traditionally, the performance of a classifier is evaluated by considering the overall accuracy against test cases [12]. However, when learning from imbalanced data sets, this measure is often not sufficient [12]. Following [7, 8, 10, 13], we employ the *overall accuracy*, *G-Mean* [8] and *F-Measures* [14] metrics to evaluate our DataBoost-IM method. The confusion matrix, as shown in Table 1, represents the typical metrics for evaluating the performance of machine learning algorithms on skew class problems.

Table 1. Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	*TN* (the number of True Negatives)	*FP*(the number of False Positives)
Actual Positive	*FN* (the number of False Negatives)	*TP*(the number of True Positives)

In Table 1, the *Precision* and *Recall* are calculated as *TP / (TP + FP)* and *TP / (TP + FN)*. The *F-measure* is defined as

$$((1+\beta^2)\times Recall \times Precision) / (\beta^2 \times Recall + Precision) \qquad (1)$$

where β corresponds to the relative importance of *precision* versus the *recall* and it is usually set to 1. The *F-measure* incorporates the *recall* and *precision* into a single number. It follows that the *F-measure* is high when both the *recall* and *precision* are high [9]. This implies that the *F-measure* is able to measure the "goodness" of a learning algorithm on the current class of interest. Note that we also use this measure for the majority class, since we are interested in measuring the performance of both classes. Another criteria used to evaluate a classifier's performance on skew data is the *G-mean* [8, 10, 13]. The *G-mean* is defined as

$$\sqrt{PositiveAcuracy \times NegativeAcuracy} \qquad (2)$$

where *Positive Accuracy* and *Negative Accuracy* are calculated as *TP / (FN+TP)* and *TN / (TN+FP)*. This measure relates to a point on the *ROC* curve and the idea is to maximize the accuracy on each of the two classes while keeping these accuracies balanced [8]. For instance, a high *Positive accurac*y by a low *Negative accuracy* will result in poor *G-mean* [8].

3 DataBoost-IM Algorithm

The DataBoost-IM approach extends our earlier DataBoost algorithm which was successfully used to produce highly accuracy classifiers in balanced domains containing hard to learn examples [15]. In this section, we describe a variation, the DataBoost-IM algorithm, applied to imbalanced data sets. This approach extends the original DataBoost algorithm as follows. Firstly, we *separately* identify hard examples from and generate synthetic examples for different classes. Secondly, the class distribution and the total weights of different classes are *rebalanced* to alleviate the learning algorithms' bias toward the majority class.

Recall that boosting involves the creation of a series of classifiers which aims to correctly classify hard to learn examples, through focusing on these hard examples during training. Following this mechanism, the DataBoost-IM algorithm, as shown in Figure 1, consists of the following three stages. Firstly, each example of the original training set is assigned an equal weight. The original training set is used to train the first classifier of the DataBoost-IM ensembles. Secondly, the hard examples (so-called seed examples) are identified and for each of these seed examples, a set of synthetic examples is generated. During the third of the algorithm, the synthetic examples are added to the original training set and the class distribution and the total weights of different classes are rebalanced. The second and third stages of the DataBoost-IM algorithm are re-executed until reaching a user-specified number of iterations or the current component classifier's error rate is worse than a threshold value. Following the AdaBoost ensemble method, this threshold is set to 0.5 [1, 2].

The seed selection, data generation and re-balancing process of the DataBoost-IM algorithm are described next.

```
Input:      Sequence of $m$ examples $\langle (x_1, y_1),\ldots,(x_m, y_m)\rangle$ with labels $y_i \in Y = \{1,\ldots,k\}$
            Weak learning algorithm WeakLearn
            Integer $T$ specifying number of iterations
Initialize  $D_j(i) = 1/m$ for all $i$.
Do for      $t = 1, 2, \ldots, T$
    1. Identify hard examples from the original data set for different classes.
    2. Generate synthetic data to balance the training knowledge of different classes
    3. Add synthetic data to the original training set to form a new training data set
    4. Update and balance the total weights of the different classes in the new training data set
    5. Call WeakLearn, providing it with the new training set with synthetic data and rebalanced
       weights
    6. Get back a hypothesis $h_t : X \rightarrow Y$.
    7. Calculate error of $h_t : \varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ If $\varepsilon_t > 1/2$, set $T = t-1$ and abort loop.
    8. Set $\beta_t = \varepsilon_t / (1' - \varepsilon_t)$.
    9. Update distribution $D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t \leftarrow if h_t(x_i) = y_i \\ 1 \leftarrow otherwise \end{cases}$, where $Z_t$ is a normali-
       zation constant (chosen so that $D_{t+1}$ will be a distribution).
Output      The final hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$
```

Fig. 1. Pseudo-code of the DataBoost-IM algorithm

3.1 Identify Seed Examples

The aim of the seed selection process is to identify hard examples for both the majority and minority classes. These examples, are used as input for the data generation process as discussed in Section 3.2.

The seed examples are selected as follows. Firstly, the examples in the training set (E_{train}) are sorted descending, based on their weights. The original training set E_{train} contains N_{maj} examples from the majority class and N_{min} examples from the minority class. The number of examples that is considered to be hard (denoted by N_s) is calculated as $(E_{train} \times Err)$, where Err is the error rate of the currently trained classifier. Next, the set E_s, which contains the N_s examples with the highest weights in E_{train}, is created. The set E_s consists of two subset of examples E_{smin} and E_{smaj}, i.e. examples from the minority and majority classes, respectively. Here, E_{smin} and E_{smaj} contain N_{smin} and N_{smaj} examples, where $N_{smin} < N_{min}$ and $N_{smaj} < N_{maj}$. We select a number of seed examples of the majority class in E_{smaj} by calculating M_L, which is equal to $min (N_{maj} / N_{min}, N_{smaj})$. Correspondingly, a subset M_S of the minority class examples in E_{smin}, is selected as seeds, where M_S is calculated as $min ((N_{maj} \times M_L) / N_{min}, N_{smin})$. The final sets of seed examples are placed in sets E_{maj} and E_{min}.

3.2 Generate Synthetic Data and Balance Class Frequencies

The aim of the data generation process is to generate additional synthesis instances to *add to* the original training set E_{train}. The data generation process extends our earlier work, as presented in [15, 16, 17], by generating data for the majority and minority classes separately. That is, the data generation process generates two sets of data. Firstly, a total of M_L sets of new majority class examples, based on each seed instance in $E_{maj,}$ are generated. For each attribute included in the synthetic example, a new value is generated based on the following constraints [15, 16, 17].

For ***Nominal*** attribute, the data generation produces a total of N_{maj} attribute values for each seed in E_{maj}. The values are chosen to reflect the distribution of values contained in the original training attribute with respect to the particular class. This is achieved by considering, for each class, the number of occurrences of different attribute values in the original data set.

For ***Continuous*** attribute, the data generation produces a total of N_{maj} attribute values. The values are chosen by considering the range *[min, max]* of the original attribute values with respect to the seed class. Also, the distribution of original attribute values, in terms of the deviation and the mean, is used during data generation.

Similarly, M_s different sets of new minority class examples, each based on a seed instance in E_{min}, are constructed. These sets of instances are added to the original training set. Interested readers are referred to [15, 16, 17] for a description of the data generation process and its evolution.

3.3 Balance the Training Weights of Separate Classes

In the final step prior to re-training, the total weights of the examples in the different classes are rebalanced. By rebalancing the total weights of the different classes, boosting is forced to focus on hard as well as rare examples.

Recall that the data generation process generates sets of synthetic examples based on seed examples E_{maj} and E_{min} corresponding to the majority and minority classes. Before the generated data are added to the original data set, each of the synthetic examples is assigned an initial weight. The initial weight of each example is calculated by dividing the weight of the seed example by the number of instances generated from it. In this way, the very high weights associated with the hard examples are balanced out. Rebalancing ensures that the boosting algorithm focuses on hard as well as minority class examples.

When the new training set is formed, the total weights of the majority class examples (denoted by W_{maj}) and the minority class examples (denote by W_{min}) in the new training data are rebalanced as follows. If $W_{maj} > W_{min,}$ the weight of each instance in the *minority* class is multiplied by $W_{maj} / W_{min,}$ Otherwise, the weight of each instance in the *majority* class is multiplied by $W_{min} / W_{maj.}$ In this way, the total weight of the majority and minority classes will be balanced. Note that, prior to training, the weights of the new training set will be renormalized, following the AdaBoost method, so that their sum equals 1 [1, 2, 3].

4 Experiments

This section describes the results of evaluating the performance of the DataBoost-IM algorithm against imbalanced data sets, in comparison with the AdaBoost benchmarking boosting algorithm [1, 2] and as well as the C4.5 decision tree, which has become a de facto standard against which new algorithms are being judged [18]. The C4.5 algorithm is also used as base classifier.

Table 2. Summary of the data sets used in this paper. Shown are the number of examples in the data set; the number of minority class; the number of majority class; the class distribution; the number of continous, and the number of discrete input features

Data set	Case	Minority Class	Majority Class	Class Distribution	Feature	
					Continuous	Discrete
CREDIT-G	1000	300	700	0.30:0.70	7	13
BREAST-CANCER	286	85	201	0.30:0.70	0	9
PHONEME	5484	1586	3818	0.29:0.71	5	0
VEHICLE	846	199	647	0.23:0.77	18	0
HEPATITIS	155	32	123	0.20:0.80	6	13
SEGMENT	2310	330	1980	0.14:0.86	19	0
GLASS	214	29	185	0.13:0.87	9	0
SATIMAGE	6435	626	5809	0.09:0.91	33	0
VOWEL	990	90	900	0.09:0.91	10	3
SICK	3772	231	3541	0.06:0.94	6	23
PRIMARY-TUMOR	339	14	325	0.04:0.96	0	17

To evaluate the performance of the DataBoost-IM method, we obtained eleven data sets from the UCI data repository [11]. These data sets were carefully selected to ensure that they (a) are based on real-world problems, (b) varied in feature characteristics, and (c) vary extensively in size and class distribution. Table 2 gives the characteristics of the data sets used for the experiments. Shown are the number of cases, the number of the majority and minority classes, the class distribution, and the type of the features. For the glass, vowel, vehicle, satimage and primary-tumor data sets, we increased the degree of skew by converting all but the smallest class into a single class. For the sick data sets, we deleted the 'TBG' attribute due to the high occurrence of missing values.

4.1 Methodology and Experimental Results

We implemented the experiments using Weka [19], a Java-based knowledge learning and analysis environment developed at the University of Waikato in New Zealand. Results for the data sets, as shown in Table 2, are averaged over five standard 10-fold cross validation experiments. For each 10-fold cross validation the data set was first partitioned into 10 equal sized sets and each set was then in turn used as the test set while the classifier trains on the other nine sets. A stratified sampling technique was applied here to ensure that each of the sets had the same proportion of different classes. For each fold an ensemble of *ten* component classifiers was created. In the experiments, the C4.5 decision trees were pruned [18].

Table 3. Test set G-mean, F-measure of minority class, F-measure of majority class, overall accuracy , true positive rate of minority class, and true positive rate of majority class for the data sets using (1) C4.5, (2) AdaBoost ensembles, and (3) DataBoost-IM ensembles

Data Set Name	Methods	G-Mean	F-measure of min. class	F-measure of maj. class	Overall Accuracy	TP rate of min. class	TP rate of Maj. Class
CREDIT-G	C4.5	56.72	43.88	80.53	71.10	37.66	85.42
	AdaBoost	58.91	45.74	78.69	69.40	43.00	80.71
	DataBoost-IM	61.96	49.64	80.22	71.60	46.66	82.28
BREAST-CANCER	C4.5	50.84	39.31	84.39	75.17	27.05	95.52
	AdaBoost	58.12	44.06	74.93	65.38	45.88	73.63
	DataBoost-IM	60.04	46.51	77.00	67.83	47.05	76.61
PHONEME	C4.5	84.22	77.23	90.07	86.17	79.88	88.78
	AdaBoost	86.74	81.86	92.60	89.48	80.83	93.08
	DataBoost-IM	88.40	83.83	93.29	90.52	83.73	93.34
VEHICLE	C4.5	92.50	87.90	96.19	94.20	89.44	95.67
	AdaBoost	95.58	92.57	97.67	96.45	93.96	97.21
	DataBoost-IM	95.77	93.70	98.06	97.04	93.46	98.14
HEPATITIS	C4.5	57.91	42.10	86.95	78.70	37.50	89.43
	AdaBoost	66.86	52.45	88.35	81.29	50.00	89.43
	DataBoost-IM	76.25	62.68	89.71	83.87	65.62	88.61
SEGMENT	C4.5	92.28	87.23	97.87	96.36	86.96	97.92
	AdaBoost	95.98	93.59	98.94	98.18	93.03	99.04
	DataBoost-IM	97.35	95.59	99.26	98.74	95.45	99.29
GLASS	C4.5	85.91	78.57	96.77	94.39	75.86	97.29
	AdaBoost	89.48	81.35	97.01	94.85	82.75	96.75
	DataBoost-IM	92.34	89.28	98.38	97.19	86.20	98.91
SATIMAGE	C4.5	72.70	56.44	95.41	91.70	55.27	95.62
	AdaBoost	77.01	66.72	96.76	94.11	60.70	97.71
	DataBoost-IM	80.42	68.86	96.76	94.14	66.61	97.10
VOWEL	C4.5	95.81	93.78	99.38	98.88	92.22	99.55
	AdaBoost	97.69	97.17	99.72	99.49	95.55	99.88
	DataBoost-IM	99.38	98.88	99.88	99.79	98.88	99.88
SICK	C4.5	93.03	89.13	99.30	98.70	87.01	99.46
	AdaBoost	94.23	91.15	99.43	98.93	89.17	99.57
	DataBoost-IM	95.96	91.84	99.46	98.99	92.64	99.40
PRIMARY-TUMOR	C4.5	0.00	0.00	97.89	95.87	0.00	100.0
	AdaBoost	37.50	19.04	97.41	94.98	14.28	98.46
	DataBoost-IM	52.62	28.57	96.92	94.10	28.57	96.92

The experimental results for all eleven data sets described in Table 2 are presented in Table 3. For each data set, we present the results achieved when using the C4.5, AdaBoostM1 and DataBoost-IM methods. Also, for each algorithm, the table presents the results in terms of the *G-mean*, *overall accuracy rates, TP rate* of the majority class and the minority class, *F-measure* of the majority class and the minority class.

One conclusion drawn from the experimental results is that the DataBoost-IM method consistently achieves higher performance on the minority class as well as the majority class in comparison with the C4.5 and AdaBoost algorithms. However, this improvement varies with different data sets. Consider the *G-mean* scores, which reflect the classifier's predictive accuracies against the majority and the minority class, as well as the degree of balance between classes. The results show that, for the *G-mean* score, the DataBoost-IM method surpasses the performance of the C4.5 and the AdaBoost algorithms in all cases. The experimental results also show that in many cases, the improvements are large. For example, in the Hepatitis data set, the AdaBoost algorithm improves on the C4.5 method's *G-mean*, i.e. 57.91% compared to 66.86%, and the DataBoost-IM algorithm significantly improved on the AdaBoost algorithm's performance, with a *G-mean* of 76.25%. In the case of the Primary-

Tumor data sets, the *G-mean* for the C4.5 method is 0%, since the C4.5 classifier achieved a 100% accuracy rate for the majority class but misclassified all minority examples. In this case, the AdaBoost algorithm produced a *G-mean* of 37.50%, whilst the DataBoost-IM ensemble achieved a *G-mean* of 52.62%. This improvement was credited with the achievement of the minority class's *TP Rate* of 28.57% by the DataBoost-IM compared to 0% by the C4.5 classifier and 14.28% by the AdaBoostM1 method. When considering the *F-measures* a similar conclusion holds. Table 3 moreover shows that, when considering the overall accuracy, the DataBoost-IM method consistently produces highly accurate ensembles.

In conclusion, the results, as shown in Table 3, indicate that the DataBoost-IM approach described here extends the predictive capabilities of the boosting ensemble and the component classifier when evaluated in terms of *overall accuracy*, *G-mean* and *F-measures.* This is achieved through integrating the data generation approach, in which class frequencies and training weights on different classes are rebalanced, into the boosting procedures.

5 Conclusion

This paper introduced a technique to create an ensemble of highly accuracy classifiers, when learning from imbalanced data sets. The DataBoost-IM algorithm extends the standard boosting approach by generating additional synthetic examples for both the majority and the minority classes, balancing the class distribution and rebalancing the training weights on different classes. In this way, boosting focuses not only on hard examples, but also on rare minority class examples. The DataBoost-IM algorithm was illustrated by means of eleven UCI data sets with various features, degrees of imbalance and sizes. The results obtained indicate that the DataBoost-IM approach increases the performance power of boosting algorithms when applied to imbalanced data sets. In particular, the DataBoost-IM algorithm achieved better predictions, in terms of the *G-mean* and *F-measures metrics*, against both the minority and majority classes, when compared with the C4.5 and AdaBoostM1 algorithms. Importantly, our method does not sacrifice one class for the other, but produce high predictive accuracy against both the majority and the minority class.

It follows that our approach should address two issues, namely finding optimal way to re-balance the learning bias toward majority class and investigating the performance against noisy data**.** Also, other weight-assignment methods will be further investigated. Future work will also include studying the voting mechanism of the boosting algorithm using different metrics such as the *ROC* curve and to provide a sound theoretical justification of the Databoost-IM parameter selection process.

References

1. Y. Freund and R. Schapire (1996): Experiments with a new boosting algorithm. the Proceedings of the Thirteenth International Conference on Machine Learning, Bari, Italy, 148-156.
2. L. Breiman (1996): Bias, Variance and Arcing Classifiers. Technical report 460, University of California: Berkeley, Berkeley, CA: USA.

3. H. Schwenk and Y. Bengio (1997): AdaBoosting Neural Networks: Application to On-line Character Recognition, International Conference on Artificial Neural Networks (ICANN'97), Springer-Verlag, 969-972.
4. T.G. Dietterich (2000): An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40, 139-157.
5. N. Japkowicz (2000): Learning from imbalanced data sets: A comparison of various strategies, Learning from imbalanced data sets: The AAAI Workshop 10-15. Menlo Park, CA: AAAI Press. Technical Report WS-00-05.
6. N. Chawla, K. Bowyer, L. Hall and W. Kegelmeyer (2002): SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.
7. M.A. Maloof (2003): Learning when Data Sets are Imbalanced and when Costs are Unequal and unknown, ICML-2003 Workshop on Learning from Imbalanced Data Sets II.
8. M. Kubat and S. Matwin (1997): Addressing the curse of imbalanced training sets: One-sided selection. Proceedings of the Fourteenth International Conference on Machine Learning San Francisco, CA, Morgan Kaufmann, 179-186.
9. M. Joshi, V. Kumar and R. Agarwal (2001): Evaluating boosting algorithms to classify rare classes: comparison and improvements. Technical Report RC-22147, IBM Research Division
10. N. Chawla, A. Lazarevic, L. Hall and K. Bowyer (2003): SMOTEBoost: improving prediction of the minority class in boosting. 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia , 107-119.
11. C.L. Blake and C. J. Merz (1998): UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, CA.
12. F. Provost and T. Fawcett (1997): Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In proceedings of the Third international conference on Knowledge discovery and data mining, Menlo park, CS. AAAI Press, 43-48.
13. M. Kubat, R. Holte and S. Matwin (1998): Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning, 30, 195–215.
14. C. J. van Rijsbergen (1979): Information Retrieval. Butterworths, London.
15. H Guo and HL Viktor (2004): Boosting with data generation: Improving the Classification of Hard to Learn Examples, the 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE), Ottawa, Canada, May 17-20, 2004.
16. HL Viktor (1999), The CILT multi-agent learning system, South African Computer Journal (SACJ), 24, 171-181.
17. HL Viktor and I Skrypnik (2001) : Improving the Competency of Ensembles of Classifiers through Data Generation, ICANNGA'2001, Prague: Czech Republic, April 21-25, 59-62.
18. JR Quinlan, C4.5 (1994): Programs for Machine Learning, Morgan Kaufmann, California: USA.
19. I. Witten, E.Frank (2000): Data Mining: Practical Machine Learning tools and Techniques with Java Implementations, Chapter 8, Morgan Kaufmann Publishers.