Alleviating the Class Imbalance problem in Data Mining

A. Sarmanova[1] **and S. Albayrak**[2]
[1]Computer Engineering, Yildiz Technical University, Istanbul, Turkey
[2] Computer Engineering, Yildiz Technical University, Istanbul, Turkey

Abstract - *The class imbalance problem in two-class data sets is one of the most important problems. When examples of one class in a training data set vastly outnumber examples of the other class, standard machine learning algorithms tend to be overwhelmed by the majority class and ignore the minority class. There are several algorithms to alleviate the problem of class imbalance in literature. In this paper the existing RUSBoost, EasyEnsemble and BalanceCascade algorithms have been compared with each other using different classifiers like C4.5, SVM, and KNN as the base learners. Several experiments have been done in order to find the best base learner and the algorithm which has the best performance according to the class distribution*.

Keywords: Class imbalance, binary classification, re-sampling, boosting.

1 Introduction

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 [1]. In addition, generally the minority class is the class of interest.

There exist techniques to develop better performing classifiers with imbalanced data sets, which are generally called Class Imbalance Learning methods [2]. These methods divided into two categories, data level and algorithm level. Data level, involve preprocessing of training data sets in order to make them balanced. Preprocessing can be implemented in two ways: re-weighting or re-balancing. For example, for data-level methods can be given re-sampling, boosting and bagging. Data re-sampling has received much attention in research related to class imbalance. Data re-sampling attempts to overcome imbalanced class distributions by adding examples to or removing examples from the data set. The second approach is algorithm level, develops new algorithms that can handle class imbalance efficiently to improve the classification performance. This category includes cost-sensitive learning [3], kernel-based algorithms [4] and recognition based algorithms [5].

Re-sampling technique can be categorized into three groups to balance the training data sets. First, the over-sampling the minority class examples, second, the under-sampling the examples of the majority class and the third, hybrid methods that combine both sampling methods mentioned above. Under-sampling methods create a subset of the original data set by eliminating some examples from majority class instances; over-sampling methods, create a superset of the original data set by replicating some examples or creating new examples from existing ones.

Boosting is the preferred algorithm when class is imbalanced. Boosting method increases the performance of classification by focusing on examples that are difficult to classify. The examples which are misclassified currently will be assigned larger weight, in order to be more likely to be chosen as a member of training subset during re-sampling at next round. A final classifier is formed using a weighted voting scheme; the weight of each classifier depends on its performance on the training set used to build it.

In this paper, the existing RUSBoost, BalanceCascade and EasyEnsemble algorithms at data level will be analyzed to alleviate the problem of class imbalance.

This paper is organized as follows. Section 2 reviews related works and in Section 3, the algorithms used in the comparison are described. Section 4 presents the experimental setting while in Section 5 experimental results obtained by different existing algorithms and finally, in Section 6 the paper is concluded.

2 Related Work

Many techniques have been proposed in literature to alleviate the problem of class imbalance. One of the newest algorithms was presented by K.Nageswara Rao et al [2]. This is a new hybrid subset filtering approach for learning from skewed training data. An easy way to sample a dataset is by selecting examples randomly from all classes. However, sampling in this way can break the dataset in an unequal priority way and more number of examples of the same class may be chosen in sampling. To resolve this problem and maintain uniformity in example, they proposed a sampling strategy called weighted component sampling. Before creating multiple subsets, they created the number of majority subsets depending upon the number of minority instances. The ratio of majority and minority examples in the imbalanced data set is used to decide the number of subset of majority examples to be created. Subsets of majority examples are combined with minority subset and multiple balanced subsets are formed. Correlation based Feature Subset (CFS) filters is applied to reduce the class imbalance effects.

There are several algorithms specifically designed for learning with minority classes. One of them is SMOTEBoost, approach for learning from imbalanced data sets which was presented by N.V. Chawla et al. [6]. The proposed SMOTEBoost algorithm is based on the integration of the SMOTE algorithm within the standard boosting procedure. Unlike standard boosting where all misclassified examples are given equal weights, SMOTEBoost creates synthetic examples from the rare or minority class, thus indirectly changing the updating weights and compensating for skewed distributions. SMOTE was used for improving the prediction of the minority classes.

Hongyu Guo, Herna L Viktor [7] presented hybrid method, called DataBoost-IM, combining synthetic over-sampling and boosting. Compare to SMOTE-Boost, DataBoost-IM synthesize new examples for both majority and minority classes, but much more examples for the minority class. DataBoost-IM chooses the hard-to-learn examples to synthesize new examples. Initially, each example is assigned with an equal weight. In every iteration, the method first identifies the hard-to-learn examples based on their weights; then it generates synthetic data based on the set and also the class distributions; more minority synthetic examples are produced than majority ones such that new training sets are balanced after combing original data and synthetic data; next, the weak learner is applied to this new training set, and error rate and weight distribution are re-calculated accordingly.

3 Algorithms Used in the Comparison

In this parer existing three algorithms: RUSBoost, EasyEnsemble and BalanceCascade, which are good at dealing with class imbalance problem, have been chosen and described in details below.

3.1 RUSBoost

C. Seiffert et al. [8] present hybrid sampling/boosting algorithm, called RUSBoost, for learning from skewed training data. This algorithm provides a simpler and faster alternative and they utilized boosting by re-sampling, which resamples the training data according to the examples' assigned weights. It is this re-sampled training data set that is used to construct the iteration's model.

RUSBoost applies RUS, which is a technique that randomly removes examples from the majority class. The motivations for introducing RUS into the boosting process are simplicity, speed, and performance. RUS decreases the time required to construct a model, which is a key benefit particularly when creating an ensemble of models, which is the case in boosting. The loss of information, which is the main drawback of RUS, is greatly overcome by combining it with boosting.

In first step, the weights of each example are initialized to 1/m, where m is the number of examples in the training data set. In second step, T (number of classifiers in the ensemble) weak hypotheses are iteratively trained. RUS is applied to remove the majority class examples. For example, if the desired class ratio is 50: 50, then the majority class examples are randomly removed until the numbers of majority and minority class examples are equal. As a result, S'_t will have a new weight distribution D'_t. S'_t and D'_t are passed to the base learner, *WeakLearn*, which creates the weak hypothesis h_t. The pseudo loss $\mathcal{E}_t$ (based on the original training data set S and weight distribution D_t) is calculated. The weight update parameter α is calculated as $\mathcal{E}_t / (1 - \mathcal{E}_t)$. Next, the weight distribution for the next iteration D_{t+1} is updated and normalized. After T iterations, the final hypothesis $H(x)$ is returned as a weighted vote of the T weak hypotheses. [8]

3.2 BalanceCascade

Under-sampling is an efficient strategy to deal with class imbalance. However, the drawback of under-sampling is that it throws away many potentially useful data. Xu-Ying Liu et al. [9] proposed two strategies to explore the majority class examples ignored by undersampling: BalanceCascade and EasyEnsemble.

BalanceCascade trains the learners sequentially, as new learners are built on examples that are filtered by previous learners. Initially, this method builds the first learner on a sampled subset containing partial majority class and the whole minority class; then a new sampled subset from majority class is filtered by such that the correct examples are removed and only incorrect ones are kept; with this refined majority subset and the minority set, a new ensemble learner is built. Iteratively, more learners are created on filtered sampling data set, and finally all learners are combined together. The BalanceCascade assumes the examples that have been correctly modeled are no longer useful on subsequent classifier construction. [9]

3.3 EasyEnsemble

EasyEnsemble samples several subsets from the majority class, trains a learner using each of them, and combines the outputs of those learners.

Given the minority training set P and the majority training set N, the under-sampling method randomly samples a subset N' from N, where |N'| < |N |. In this method, they independently sampled several subsets $N_1, N_2, ..., N_T$ from N. For each subset N_i ($1 \leq i \leq T$), a classifier H_i is trained using N_i and all of P. All generated classifiers are combined for the final decision. AdaBoost [10] is used to train the classifier H_i. [9]

4 Experimental Study

The experiments were conduct using eight real world benchmark data sets taken from the UCI Machine Learning Repository. The details of these datasets used in this study are shown in Table I. The experiments have been done using Matlab and WEKA. C4.5 (denoted J48 in WEKA) decision tree, support vector machine (SVM, denoted SMO in WEKA) and k-nearest neighbor (KNN, denoted Ibk in WEKA) algorithms are used as the base learners to validate the compared algorithms.

This paper uses three different performance metrics to evaluate the algorithms compared for our experiments, all of which are more suitable than the overall accuracy when dealing with class imbalance. In general, for binary class problems the performances of classifiers are evaluated by a confusion matrix (Table II). Based on the confusion matrix, three popular measures have been proposed: AUC, F-measure and G-mean. In our experiment these three evaluation measures are used to validate the compared methods. The classification methods are repeated ten times considering that the re-sampling of subsets introduces randomness. The AUC, F-measure and G-mean are averaged from these ten runs. These well known and widely used measures are defined in the Table III:

TABLE I. DATA SETS

Datasets	Size	Atribute	Majority	Minority
breast	699	10	458	241
bupa	345	7	200	145
haberman	306	4	225	81
hepatitis	155	18	123	32
ionosphere	351	35	225	126
pima	768	9	500	268
transfusion	748	5	570	178
wpbc	198	35	151	47

TABLE II. CONFUSION MATRIX

	Predicted class (Positive)	Predicted class (Negative)
Actual class (Positive)	True Positives TP	False Negatives FN
Actual class (Negative)	False Positives FP	True Negatives TN

TABLE III. EVALUATION MEASURES

False Positive Rate	FP_{RATE} (fpr) = FP/(FP+TN)
True Positive Rate	TP_{RATE} (Acc_+) = TP/(TP+FN)
True Negative Rate	TN_{RATE} (Acc_-) = TN/(TN+FP)
AUC	AUC = $(1+TP_{RATE}- FP_{RATE})/2$
G-mean	G− mean =$\sqrt{Acc_+ \times Acc_-}$
Precision	Precision = TP/(TP+FP)
Recall	Recall = Acc_+
F−measure	F-measure = (2×Precision×Recall)/(Precision+Recall)

5 Experimental Results

In this section the several experiments have been done using RUSBoost, BalanceCascade and EasyEnsemble algorithms. In subsection 5.1 the experiments have been done in order to find the best base learner among the C4.5, SVM and KNN and the best performed algorithm has tried to be found in subsection 5.2. In subsection 5.3 the experiments have been done with good performed algorithm according to the different class distribution.

5.1 Base Learners Performance

This section presents the results of our experiments with RUSBoost, BalanceCascade and EasyEnsemble. We investigated the performance of these techniques by different learners when classification models are trained using C4.5, SVM and KNN. We used AUC, F-measure, G-mean to evaluate the compared algorithms. The best results for the three algorithms have been obtained when the C4.5 is used as the base learner which we can observe from Fig. 1-9. The details are given below.

RUSBoost. When we calculated AUC (Fig.1) C4.5 has performed well on tree data sets and SVM has outperformed for four of eight data sets and KNN on one data sets. From Fig.2-3, we can see the results of RUSBoost in terms of F-measure and G-mean. When the C4.5 is used as the base learner it has outperformed for four of eight data sets, SVM has performed well on tree data sets and KNN on one data set.

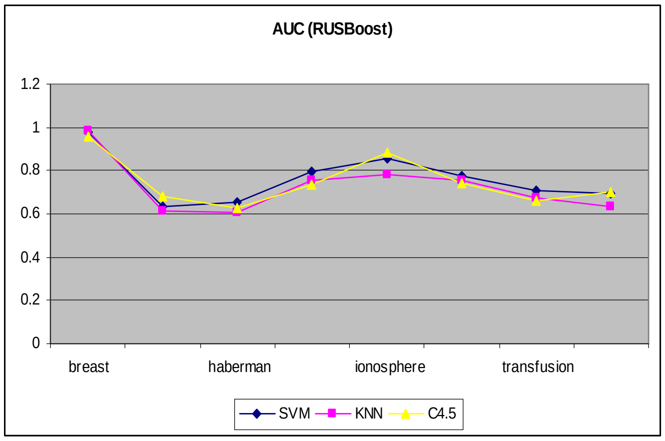

Fig. 1. AUC results of RUSBoost algorithm

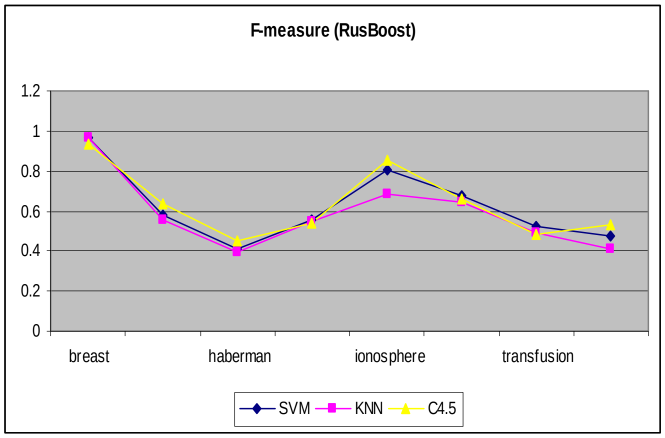

Fig. 2. F-measure result of RUSBoost algorithm

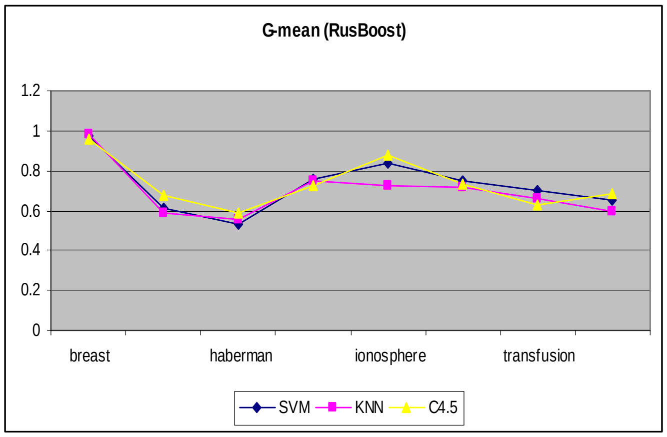

Fig. 3. G-mean results of RUSBoost algorithm

BalanceCascade. From Fig.4, we can observe the results of BalanceCascade algorithm in terms of AUC. When the C4.5 is used as the base learner the good results have been obtained in six out of eight data sets. When SVM is taken as the base learner it has been successful on two data sets. F-measure and G-mean (Fig.5-6) using C4.5 have been performed well over all data set. SVM and KNN are not performed well when they were used as the base learners.

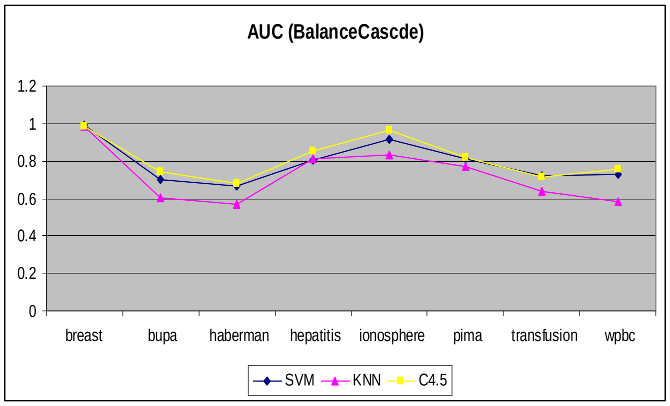

Fig. 4. AUC results of BalanceCascade algorithm

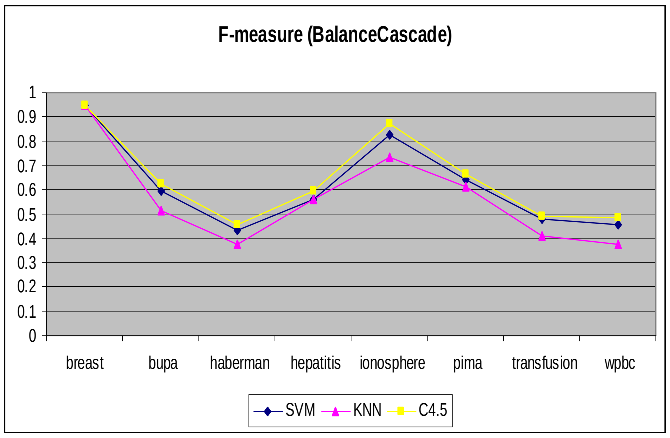

Fig. 5. F-measure results of BalanceCascade algorithm

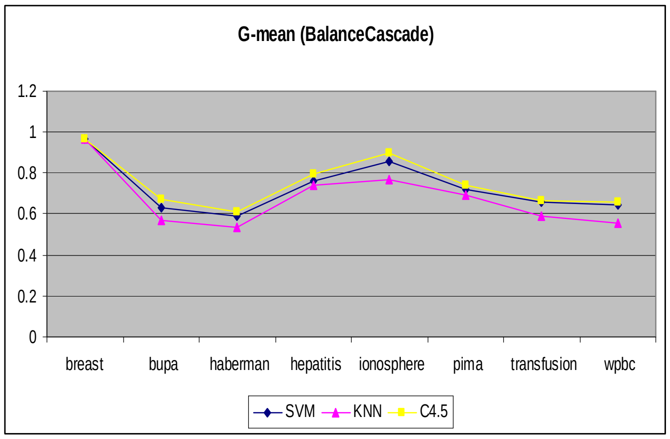

Fig. 6. G-mean results of BalanceCascade algorithm

EasyEnsemble. Fig. 7, 9 show the performance of EasyEnsemble algorithm as measured using AUC and G-mean. C4.5 has performed well in six out of eight data sets and SVM in two data sets. When using F-measure (Fig.8) to measure the performance using C4.5 it has been successful on five data sets and SVM on tree data sets. When KNN is selected as the base learner it has not performed well.

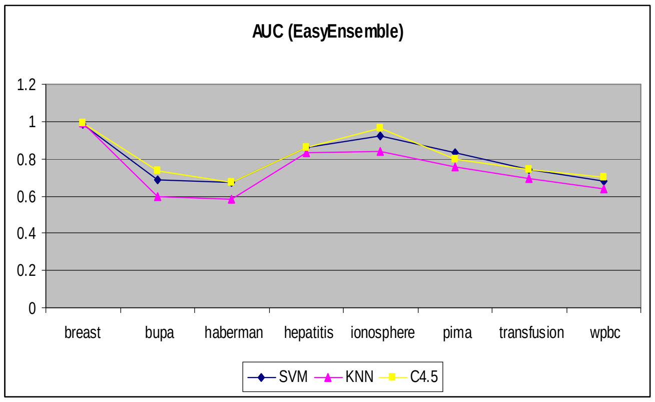

Fig. 7. AUC results of EasyEnsemble algorithm

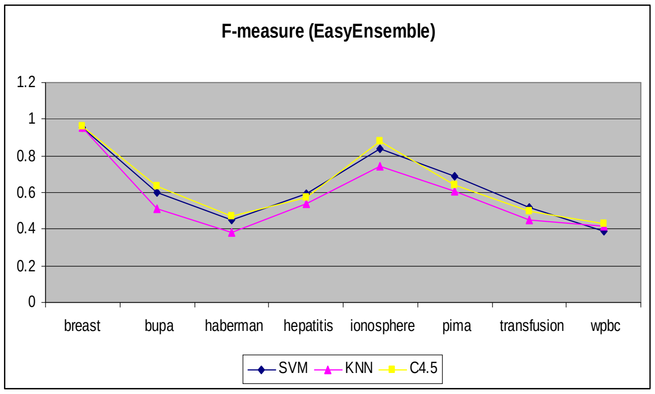

Fig. 8. F-measure results of EasyEnsemble algorithm

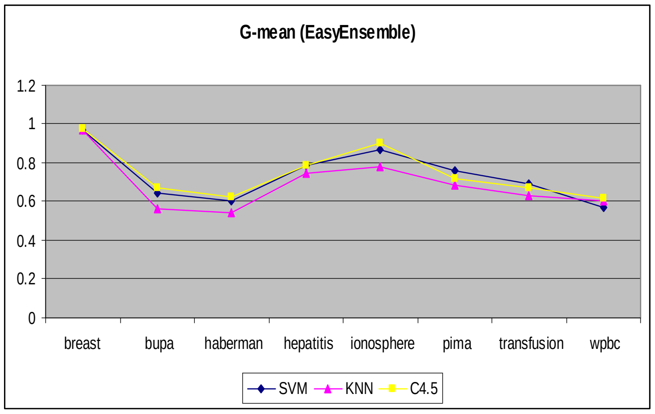

Fig. 9. G-mean results of EasyEnsemble algorithm

Tables IV-VI show the performance of C4.5, SVM and KNN using RUSboost, BalanceCascade and EasyEnsemble algorithms according to AUC, F-measure and G-mean, which are averaged over all data sets.

Table IV shows that the RUSBoost algorithm has got good results according to F-measure and G-mean using C4.5 and good AUC result using SVM.

Tables V, VI show that the BalanceCascade and EasyEnsemble algorithms have got good results according to AUC, F-measure and G-mean using C4.5.

According to these tables we can see that when C4.5 is used as the base learner these evaluation measures are obtained more successful results than SVM and KNN.

TABLE IV. MEAN VALUE FOR RUSBOOST

Algorithm	Base learners	AUC	F-measure	G-mean
RUSBoost	C4.5	0.7483	**0.6366**	**0.7345**
	SVM	**0.7625**	0.6234	0.7269
	KNN	0.7266	0.5863	0.6979

TABLE V. MEAN VALUE FOR BALANCECASCADE

Algorithm	Base learners	AUC	F-measure	G-mean
Balance Cascade	C4.5	**0.814**	**0.644**	**0.752**
	SVM	0.792	0.619	0.728
	KNN	0.723	0.567	0.678

TABLE VI. MEAN VALUE FOR EASYENSEMBLE

Algorithm	Base learners	AUC	F-measure	G-mean
Easy Ensemble	C4.5	**0.808**	**0.636**	**0.747**
	SVM	0.797	0.628	0.735
	KNN	0.741	0.576	0.689

5.2 Algorithm Performance

According to our prior experiment all three algorithms have produced good results when C4.5 is used as the base learner. Our aim is to find out which algorithm is better than the others according to the C4.5 as the base learner.

Table VII show the performance of RUSBoost, BalanceCascade and EasyEnsemble algorithms according to AUC, F-measure and G-mean. When AUC was calculated EasyEnsemble and BalanceCascade has performed well on four data sets, RUSBoost has not performed well. When F-measure and G-mean was calculated, EasyEnsemble has performed better than RUSBoost and BalanceCascade on four data sets. RUSBoost and BalanceCascade algorithms have been successful on two data sets**.**

The experiments show that the EasyEnsemble performs better than RUSBoost and BalanceCascade when C4.5 is used as the base learner.

5.3 Class Distribution Analysis

EasyEnsemble has been found as more successful algorithm according to our prior experiments. In EasyEnsemble there were given the minority training set P and the majority training set N, the under-sampling method has randomly sampled a subset N' from N, where |N'| <|N|. In our previous experiments, examples of P minority and N majority class were resampled equally (50-50). In this section, the experiments have been done like the distribution of majority and minority class examples are 55-45, 60-40 and 65-35. We can see the experimental results from Table VIII. According to the results EasyEnsemble has produced more successful results when the distribution of majority and minority class examples was 55-45.

TABLE VII. THE AUC, F-MEASUE AND G-MEAN RESULTS OF RUSBOOST, EASYENSEMBLE AND BALANCECASCADE ALGORITHMS USING C4.5

	AUC			F-measure			G-mean		
	RUSBoost	Balance Cascade	Easy Ensembl e	RUSBoost	Balance Cascade	Easy Ensembl e	RUSBoost	Balance Cascade	Easy Ensembl e
Breast	0.9569	0.987	**0.993**	0.9380	0.951	**0.961**	0.9567	0.969	**0.975**
Bupa	0.6807	**0.745**	0.738	**0.6362**	0.626	0.636	**0.6778**	0.673	0.672
Haberman	0.6303	**0.677**	0.674	0.4487	0.454	**0.468**	0.5889	0.612	**0.623**
Hepatitis	0.7353	0.852	**0.862**	0.5375	**0.597**	0.573	0.7242	**0.797**	0.789
Ionosphere	0.8845	0.961	**0.963**	0.8567	0.875	**0.878**	0.8808	0.897	**0.904**
Pima	0.7413	**0.821**	0.797	0.6634	**0.666**	0.644	0.7360	**0.739**	0.720
Transfusion	0.6598	0.712	**0.741**	0.4832	0.493	**0.501**	0.6254	0.668	**0.674**
Wpbc	0.6978	**0.754**	0.697	**0.529**	0.488	0.431	**0.6858**	0.657	0.618
	0/8	***4/8***	***4/8***	2/8	2/8	**4/8**	2/8	2/8	***4/8***
Number of the best performed data sets									

TABLE VIII. PERFORMANCE OF EASYENSEMBLE AVERAGED OVER ALL DATA SETS IN TERMS OF CLASS DISTRIBUTION

	Class distribution (majority – minority)			
	50-50	55-45	60-40	65-35
AUC	0.794	**0.801**	0.788	0.79
F-measure	0.596	**0.61**	0.595	0.59
G-mean	0.733	**0.738**	0.724	0.716

6 Conclusion

In this paper RUSBoost, BalanceCascade and EasyEnsemble algorithms have been compared in order to alleviate the problem of class imbalance, which is one of the most important problems faced in data mining, according to base learners performance and algorithm performance. Experiments have been done on real-world data sets using the C4.5, SVM and KNN as the base learners. The performances of classifiers have been compared using AUC, G-mean and F-measure. When C4.5 is used as the base learner it has given better results than SVM and KNN learners for all three algorithms. According to the results of the experiment EasyEnsemble algorithm has been found as the best algorithm to alleviate the problem of class imbalance and with class distribution 55-45 (majority - minority).

7 References

[1] N. Japkowicz. "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, 2000.

[2] K. Nageswara Rao, Prof. T. Venkateswara rao, Dr. D. Rajya Lakshmi, "A Novel Class Imbalance Learning Method using Subset Filtering". International Journal of Scientific & Engineering Research Volume 3, Issue 9, September-2012 1 ISSN 2229-5518.

[3] W. Fan, S. J. Stolfo, J. Zhang, "AdaCost: misclassification cost-sensitive boosting," Proc.Int. Conf. Machine Learning, Bled, Slovenia, June, 1999, pp. 97-105.

[4] Yuchun Tang, Yan-Qing Zhang, N. V. Chawla, "SVMs modeling for highly imbalanced classification," IEEE Trans. Syst., Man, and Cybern. - Part B, vol. 39, no. 1, pp. 281 - 288, Feb. 2009.

[5] Zhi-Qiang Zeng and Ji Gao, "Improving SVM classification with imbalance data set," Proc. 16th Int.Conf. Neural Information Processing (ICoNIP 2009), Bangkok, Thailand, 2009, pp. 389-398.

[6] N.V. Chawla, A. Lazarevic, L. O. Hall, and K.W.Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in Proc. Knowl. Discov. Databases, 2003, pp. 107–119.

[7] Hongyu Guo, Herna L Viktor, "Learning from Imbalanced Data Sets with Boosting and Data Generation: The DataBoost-IM Approach", ACM SIGKDD Explorations Newsletter, 2004.

[8] Seiffert C., Khoshgoftaar T. M., Van Hulse J., & Napolitano A., "RUSBoost: A Hybrid Approach to Alleviating Class Imbalance," IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 40(1), 185-197, 2010.

[9] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou, "Exploratory undersampling for class imbalance learning," IEEE Transactions on Systems, Man and Cybernetics, 39(2):539-550, 2009.

[10] R. E. Schapire, "A brief introduction to Boosting," in Proceedings of the 16th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 1999, pp. 1401–1406.