Probabilistic Predictions of Ensemble of Classifiers Combined with Dynamically Weighted Majority Vote

Eftim Zdravevski

PROBABILISTIC PREDICTIONS OF ENSEMBLE OF CLASSIFIERS COMBINED WITH DYNAMICALLY WEIGHTED MAJORITY VOTE

Eftim Zdravevski[1,2], Andrea Kulakov[2], Slobodan Kalajdziski[2], Danco Davcev[2]
[1]NI TEKNA – Intelligent Technologies, Negotino, Macedonia
eftim.zdravevski@ni-tekna.com
[2]University Ss. Cyril and Methodius, Faculty of Electrical Engineering and Information Technologies, Skopje, Macedonia
{eftim,kulak,skalaj,etfdav}@feit.ukim.edu.mk

ABSTRACT
This paper presents a new method for dynamic calculation of weights that can be used in the process of aggregation of classifications by weighted majority vote. The proposed method can be used for all binary classification problems for classifiers that produce probabilistic classifications. Most aggregation functions produce an output which only represents the aggregated classification of an ensemble of classifiers and sometimes this isn't enough. This paper also proposes a method for estimation of the probability of an aggregated classification. The estimated probability of the aggregated classification is essential if the performance of the ensemble of classifiers needs to be expressed in terms of Area Under the Receiver Operating Curve or some other performance measures that classifications' probability. The experimental results demonstrate the performance improvements obtained by applying the proposed methods to an ensemble of classifiers compared to individual classifiers.

KEY WORDS
Machine learning, prediction methods, pattern classification, decision-making

1. Introduction

Almost all kinds of classification tasks go through the most of the following phases: data acquisition, data preprocessing, selection of features and classification algorithm, training, testing, and validation. Depending on the situation, some of these phases may be repeated several times. The designer of the system aims to improve its performance by carefully investigating each phase. After all possibilities are explored and when the potential of improvement of each phase is exhausted, the design of the system is final. This means that a plateau is reached and the achieved performance is optimal for the available data, tools and methodologies. In other words, when considerable knowledge of single classifier models has been accumulated, combining classifiers comes as a natural step forward, as it is suggested in [1].

There are three types of reasons, as it is elaborated in [2], why a design with classifier ensemble might be better than a design with single classifier. However, using multiple classifiers increases complexity of the systems and may raise new issues, which is discussed in [3].

The algorithm that we propose in this paper is a new scheme for dynamically assigning weights to individual classifiers for binary problems. Namely, during the training phase for each classifier several different weights are computed. During the test phase for each test case each classifier makes a classification. Depending on the probability of the classification a particular classifier makes, one of the predetermined weights for that classifier will be used. This method evaluates the competency of the classifiers dynamically, which results in dynamically weighted voting.

The weighted majority vote strategy is widely used. However, the fact that it does not produce probabilistic predictions stands in the way of computing the Area Under the Receiver Operating Curve (AUC ROC) of ensembles that use this combination strategy. In this paper we propose a technique of assigning probabilities to each output of an ensemble of classifiers for binary problems. Using the proposed technique we were able to compare the tested ensembles of classifiers to the individual classifiers and determine whether the ensembles were worthwhile complication to the system.

2. Related work

There are many combination logics that can be used to combine the classifications of more classifiers: majority vote with its variations (simple majority, plurality and unanimity), weighted majority vote, naive Bayes combination, feed forward neural network combination etc. In this section we give a brief description of the two most widely used combination strategies.

2.1 Majority vote

This strategy is widely used even in real life situations because it is very reasonable, intuitive and has some important advantages over other techniques. Generally, this voting strategy can contribute to better performance. Depending on the scenario and distribution of votes of the individual classifiers, the performance can vary, but the interval of accuracy can be empirically computed. This strategy is described in more detail in [1], pages 112-123, in [4] and in [5].

2.2 Weighted majority vote

If the classifiers within an ensemble don not have nearly identical accuracy, it is reasonable to attempt to assign greater voting power to the more competent classifiers in the process of making a final decision. One way to accomplish this is by weighted majority vote. Classifier's outputs can be represented as degrees of support for the classes. If classifier D_i labels the test case x with class ω_j, then the degree of support $d_{i,j}$ of classifier D_i for class ω_j is 1, and 0 for all other classes. This is represented with eq. (1).

$$d_{i,j} = \begin{cases} 1, & \text{if } D_i \text{ labels } x \text{ with } \omega_j \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$G_j(x) = \sum_{i=1}^{L} b_i d_{i,j} \quad (2)$$

The discriminant function G_j for class ω_j and test case x, obtained through weighted voting is represented in eq. (2), where b_i is the weight coefficient for classifier D_i. G_j is the sum of the weight coefficients of the ensemble classifiers who predicted the class ω_j, for the test case x. The ensemble predicts the class which has the maximum discriminant function.

2.3 Assigning weight of classifiers for weighted majority vote

There are many schemes of assigning weights to the classifiers of the ensemble and all of them aim to give greater weights to the more competent classifiers. Different schemes of assigning weights may produce different performance of the ensemble. The following theorem[1] formalizes one way of selecting weights for the classifiers:

[1] The proof of this theorem is in [1] (Theorem 4.1).

Theorem 1: Consider an ensemble of L *independent*[2] classifiers $D_1,...,D_L$ with individual accuracies $a_1,...,a_L$. The outputs are combined by the weighted majority vote. Then the accuracy of the ensemble P_{maj}^{w} is maximized by assigning weights b_i in the following manner:

$$b_i \propto log \frac{a_i}{1 - a_i} \quad (3)$$

where a_i is the accuracy, and b_i is the weight of i-th classifier.

3. Probabilistic classifications of ensemble of classifiers aggregated with dynamically weighted majority vote

In this section, we describe our algorithm for dynamically weighted vote within an ensemble of classifiers and introduce a method of estimating probabilities of the predictions of an ensemble that uses weighted voting for aggregation of individual predictions.

3.1 Dynamically weighted majority vote within ensemble of classifiers for binary problems

Most of schemes that are being used at the moment are static, i.e. they compute a weight for each classifier during the training phase and use this weight constantly throughout the test phase for all test cases. There are some dynamic approaches that apply adaptive weighted majority vote using genetic algorithms, described in [7], and depending on the position of the input data in the feature space, described in [8].

Let us consider an example of a probabilistic classifier, and assume that its performance can be summarized with Table 1. If this classifier is a voter in an ensemble, and if we use the static method described in section 2.3 to assign weights, then it would have voting weight of 0.226 (first row in Table 1).

All of the predictions of the exemplary classifier are grouped in four distinct groups (Table 1), depending of the predictions' probabilities. By doing this, the accuracy and recall for each group can be calculated. It can be noted that, the accuracy of the same classifier increases, as the probability of its predictions increases. In general, this may not be true, i.e. some classifier may have worse accuracy when it predicts with high

[2] Here the conditional independence is considered, i.e. $P(s|\omega_j) = \prod_{i=1}^{L} P(s_i|\omega_j)$.

probability. However, the performance of the exemplary classifier corresponds to the performance on many real classifiers on real data sets. This inspired us to dynamically assign weights in a majority weighted vote scheme. The values in column weight (b) in Table 1 are calculated using equation (3), and represent the weight of the exemplary classifier for each particular group of predictions. To elaborate that, let us consider an example with two classifiers, and let the performance of both classifiers is summarized with Table 1. They both predict one test case. The first classifier D_1 predicts the test case i with probability $p_{i,1}$=0.7 and the second classifier predicts the same test case with probability $p_{i,2}$=0.95. If the classifiers are part of an ensemble and if we use the method described in section 2.3, to both of them the weight b=0.226 would be assigned. However, Table 1 shows that the two predictions belong to different groups in respect to the probability of the predictions, and the accuracy of each group is different. It makes reasonable sense to assign different weights to the different classifiers, although they have identical overall accuracy. For the considered test case, because the first classifier predicted the test case with probability 0.7, and because this prediction belongs to Group 2, we can assign a weight of 0.269 to its vote. The second classifier predicted the same test case with probability 0.95, and because this prediction belongs to Group 4, the weight of its vote would be 0.753. Note that, we use weights from the same table for both classifiers only to save space in the paper. Two real classifiers would rarely have the same distribution of accuracy and recall, in their groups of probability. For the considered test case, the second classifier has greater credibility, and therefore, greater weight of the vote in the decision making process.

The explained example is an illustration of Algorithm 1, which formalizes the proposed algorithm for dynamically weighted majority vote within ensemble of classifiers for binary problems. The algorithm assumes that there is an ensemble of L independent probabilistic classifiers. The output of these classifiers is a predicted class and a probability of the prediction. The interval of probabilities of the predictions [0.5, 1] is divided into n subintervals SI: SI_1=[0.5, s_1], SI_2= (s_1, s_2],..., SI_n=(s_{n-1}, 1].

In Algorithm 1, the method EnsemblePredictTestCase uses eq. (3) for weighted majority vote, and uses different weights of the classifiers for each test case. As it was already mentioned, to each subinterval corresponds a portion of all predictions a particular classifier makes, and the recall of the subintervals can be calculated. The subintervals should be chosen with respect to their recall, i.e. the number of predictions that belong to each subinterval should be considered. If to some subinterval correspond only a small percent of the total predictions, then its recall will be small. Subintervals like that should be avoided, because they would not be able to contribute to better overall results of the ensemble, instead, they would just increase the complexity of the system. That being said, the recall of subintervals should not be smaller than 0.1, and there should not be more than 10 subintervals. Consider the opposite case: a particular classifier we have selected a subinterval which corresponds to less than 10% of all predictions (recall<0.1). However, let us assume that the accuracy, hence the weight, of that subinterval are very high. In an ensemble with many classifiers, such situation will occur rarely (less than 10% of the cases), and even then, the final decision would still depend on the majority of classifiers, regardless of the very strong weight of that particular classifier. If that subinterval was merged with some adjacent subinterval, then its weight would be slightly weaker than before the merging. However, the overall decision of the ensemble is not likely to be changed, meaning that the small subinterval didn't make any contributions to better results. These conclusions were inferred from the tests that we conducted (described in section 4). If the number of subintervals was greater

Table 1. Accuracy, recall and weight of an exemplary classifier

Group	Probability (p)	Accuracy (a)	Recall (r)	Weight (b)	Description
All	$p \geq 0.5$	0.6275	1	0.226	All test cases (r=1) are predicted with p≥0.5 and the overall accuracy is 0.6.
Group 1	$0.6 > p \geq 0.5$	0.55	0.5	0.087	50% of the test cases (r=0.5) are predicted with 0.6>p≥0.5and the accuracy is 0.55.
Group 2	$0.75 > p \geq 0.6$	0.65	0.25	0.269	25% of the test cases (r=0.25) are predicted with 0.75>p≥0.6 and for those test cases the accuracy is 0.65.
Group 3	$0.9 > p \geq 0.75$	0.7	0.15	0.368	15% of the test cases (r=0.15) are predicted with 0.9>p≥0.75 and for those test cases the accuracy is 0.7.
Group 4	$p > 0.9$	0.85	0.1	0.753	Only 10% of the test cases (r=0.1) are predicted with p>0.9, but for those test cases the accuracy is 0.85.

than five, the system implementation was slightly more complicated, but there was not any performance gain.

3.2 Probabilistic predictions of ensemble of classifiers for binary problems

Algorithm 1. Dynamically weighted majority vote within ensemble of classifiers for binary problems

For each classifier D_i in D, $0 \le i \le L$
  M_i = BuildModel(in D_i, in training_data_set);
  For each subinterval SI_j in SI, $1 \le j \le n$
    $Acc_{j,i}$ = CalculateAccuracy(in SI_j, in M_j);
    $Rec_{j,i}$ = CalculateRecall(in SI_j, in M_j);
    $W_{j,i}$ = CalculateWeight(in $Acc_{j,i}$, in M_j);
  End for each
End for each

For each test case TC_k in test_data_set, $0 \le k < m$
  For each classifier D_i in D, $0 \le i \le L$
    PredictTestCase(in M_i, in TC_k, out $p_{k,i}$, out $cl_{k,i}$);
    // D_i labels TC_k with class $cl_{k,i}$ and probability $p_{k,i}$
    j = GetProbabilitySubintervalIndex(in $p_{k,i}$);
    $w_{k,i} = W_{j,i}$; // the weight that would be given to
                // classifier D_i when making
                // a decision for test case TC_k
  End for each
  EnsemblePredictTestCase(in w_k, out p_k, out cl_k);
  // combines predictions
  // of individual classifiers using weighted majority vote
  // w_k is the array of weights of all classifiers for test case TC_k
  // the ensemble labels TC_k with class cl_k and probability p_k
End for each

In order to evaluate the performance of the ensemble of classifiers, a performance metric should be chosen. Usually the accuracy (the hit rate) is used as a metric, based on which, the performance of classifiers is compared. An alternative metric that is also used for the same purpose is the AUC ROC. Both metrics represent the performance of a given classifier by a single number. However, the accuracy depicts the performance for a particular decision threshold, whereas the AUC ROC is invariant of the decision threshold. There is a formal mathematical proof in [6], which shows that the AUC ROC is statistically more consistent, and more discriminating than the accuracy. The calculation of the AUC ROC uses the probability for each classification, and if this information is not available, the AUC ROC would not be computable.

Using the proposed method for dynamically weighted majority vote, the ensemble of classifiers predicts the class that has greater sum of weighted votes. So far, the output of the ensemble only contains a predicted class. With this information only the accuracy of the ensemble could be computed. In order to calculate the AUC ROC of the classifier, the ensemble's output has to contain the probability of each prediction, as well as, the predicted class.

For binary problems, the classifiers of an ensemble are assigned weights, and they vote for one of the two classes. Note that, their weights can be different for each test case according to the proposed method in section 3.1. After the votes for each class are summed up, two values are obtained – s_1 (votes for class 1) and s_2 (votes for the class 2). The *first* idea for calculation of the probability of the prediction of the ensemble, p_a, is expressed with eq. (4):

$$p_a^1 = \begin{cases} \frac{s_1}{s_1 + s_2}, \text{when } s_1 \ge s_2. \text{ Class 1 is predicted.} \\ \frac{s_2}{s_1 + s_2}, \text{when } s_2 > s_1. \text{ Class 2 is predicted.} \end{cases} \quad (4)$$

This approach is very reasonable, but has one flaw. Namely, if, for some particular test case, all classifiers within the ensemble can vote unanimously (predict the same class), then p_a^1 will be 1 regardless of the probabilities of the predictions of the individual classifiers. This approach doesn't make a difference between the following opposite scenarios: all classifiers predict the same class with very low probability of the individual predictions (for instance, about 0.55), and all classifiers predict the same class with very high probability of the individual predictions (above 0.9). Obviously, these scenarios differ, but by using this approach, p_a^1 would be 1 for both of them. A modification has to be made so the method will discriminate on these scenarios.

An alternative idea is the following: after the ensemble makes a decision on the predicted class (comparing s_1 and s_2), the maximum probability of predictions of the individual classifiers that voted in favor of the predicted class can be used as a probability of the prediction of the ensemble. Let the sets D_1 and D_2 contain the indices of the classifiers that predicted class 1 and class 2, respectively. Then, the probability of the prediction of the ensemble, p_a^2, would be computed using eq. (5):

$$p_a^2 = \begin{cases} \max_{i \in D_1}(p_i), \text{when } s_1 \ge s_2. \text{Class 1 is predicted.} \\ \max_{i \in D_2}(p_i), \text{when } s_2 > s_1. \text{Class 2 is predicted.} \end{cases} \quad (5)$$

This approach also has flaws in some particular cases. Namely, if s_1 and s_2 are nearly identical (but $s_1 > s_2$), then the decision would be a close call, and s_1 would be the probability of the final decision. If s_1 is great (greater than 0.9), then the final decision would seem to be very reliable, although this was not the case. Obviously, in such cases discrimination has to be made.

A *third* idea is to make a combination of the previous two ideas. The flaws of the both previous ideas are that, they suggest predictions with *great* probabilities, even in cases when the decisions are close calls, hence the probability of the predictions should be smaller. It is obvious that the probabilities computed with the previous ideas are greater than they should actually be. It should, also, be noted that the problems of the both ideas are opposite, i.e. they cannot occur at the same time, so using the smaller of p_a^1 and p_a^2 would balance the system:

$$p_a = \min(p_a^1, p_a^2) \quad (6)$$

All tree ideas were tested on a real problem, and the third idea gave best results. This will be illustrated in the next section.

4. Results

In this section we present the experimental results that were obtained using the algorithm we propose. We worked on a problem from a real domain – credit risk assessment on a credit card application. Basically, a retail chain offers credit to potential clients, and they can use it to buy goods in the stores of the retail chain. This problem is the topic of PAKDD 2009 Data Mining Competition [9]. There are three available data sets: training, validation and a test set. The data in these sets is from different and nonconsecutive time intervals, and different selection procedures; therefore it is a challenging task to build a robust model that would not have significant degradation of performance over time. The initial data sets consist of 31 attributes, and using some transformations some irrelevant attributes were ignored, some interactions between the attributes were discovered and properly modeled, and some new attributes were generated.[3] The architecture of the system that is used to solve the problem is similar to the one used in [10], with an improvement in the feature selection process. Here, the feature selection process is an iterative cycle, i.e., it is performed until the best subset of the available features[4] is determined.

The tested classifiers are consistent with the naming convention in [11], where they are described in more detail. After the feature selection, based on information value and stability over time of the attributes, the data sets consist of 17 independent attributes. The training data set has 50000 data point that are collected during a period of one year. The validation data set has 10000 data points that are also collected during a period of one year. There is a gap of one year between the data in training and the validation data sets. All classifiers listed in Table 2 use the same data sets. This is a binary problem and the distribution of classes in the training set is 80% versus 20%, whereas the validation data set has undisclosed distribution of classes.

From Table 2 can be noted that the performance, in terms of AUC ROC, of the ensembles is generally improved over the performance of the individual classifiers on the unseen validation data set, while maintaining similar performance using 10 fold cross validation of the training set. Our best result is achieved with an ensemble of three fairly independent classifiers, and is comparable to the result of the winner of the competition. Their best result is AUC ROC of 0.6134, which is less than 1% better than our best result.

At one point of our research, we started testing other static approaches of combinational logics. The initial results showed that if we use the maximum function to combine the individual predictions, then there is always some classifier that is too optimistic making a prediction with high probability. If it makes a wrong prediction and all other classifiers make a correct prediction, but with smaller probability, the ensemble would make a wrong decision. This is a very common situation, particularly when the data set is not balanced.

[3] Irrelevant attributes have low information value [12]. Interactions between variables were discovered using the methods described in [12], and they were modeled by generating new combinational and dummy variables.

[4] The set of features consists of original and transformed attributes, as well as, newly generated attributes.

Table 2. Performance of individual classifiers and ensembles of classifiers

	Classifier	Unseen validation data set	10 fold cross validation	
		AUC ROC	AUC ROC	Accuracy
1	bayes_BayesNet_&_misc_VFI_&_functions_RBFNetwork	0,6090	0,6833	0,7671
2	bayes_BayesNet_&_trees_ADTree	0,6085	0,6927	0,8056
3	bayes_BayesNet_&_misc_VFI	0,6079	0,6823	0,7038
4	bayes_BayesNet_&_misc_VFI_&_functions_Logistic	0,6070	0,6942	0,7928
5	bayes_BayesNet_&_functions_Logistic	0,6061	0,6987	0,8086
6	bayes_BayesNet	0.6058	0.6865	0.6944
7	bayes_BayesNet_&_misc_VFI_&_trees_ADTree	0,6051	0,6884	0,7906
8	bayes_BayesNet_&_functions_Logistic_&_trees_ADTree	0,6038	0,7036	0,8077
9	functions_RBFNetwork	0.5969	0.6607	0.8025
10	misc_VFI	0.5944	0.6451	0.5809
11	meta_LogitBoost	0.5942	0.6974	0.8057
12	functions_Logistic	0.5941	0.7162	0.8085
13	trees_ADTree	0.5941	0.6943	0.8056
14	bayes_NaiveBayes	0.5937	0.6734	0.6260
15	meta_RacedIncrementalLogitBoost	0.5918	0.6702	0.7979
16	functions_Logistic_&_trees_ADTree	0,5909	0,7160	0,8085
17	trees_NBTree	0.5891	0.6689	0.7982
18	trees_REPTree	0.5783	0.6634	0.7963
19	meta_FilteredClassifier	0.5775	0.6723	0.8063
20	meta_AttributeSelectedClassifier	0.5667	0.6457	0.8046

All of this resulted in poor performance, namely the AUC ROC was about 0.55 for the same features, the same classifiers and the same validation data set. When using the average function, the probabilities were a little more balanced, which resulted in slightly better, but still poor results. The minimum function was better than both of them, because it makes more conservative predictions with low probabilities. However, its predictions are sometimes with too low probability, hence the performance is about 0.58 in terms of AUC ROC. The majority rule produces best results of these combinational schemes; namely, the AUC ROC is from 0.58 to 0.595, depending on the classifiers in the ensemble. However, this performance is worse than the performance of some individual classifiers (Table 1), so there is no point of making things more complicated, and not gaining anything.

4. Conclusion

Using multiple classifiers for a given problem becomes a trend; therefore various methods for combination of predictions of individual classifiers are being tested. The weighted majority strategy is widely used. This paper presented a modification of this strategy, which dynamically assigns weights to the votes of the individual classifiers. The various classifiers within an ensemble can be experts for different types of test cases, although they have similar overall performance. The proposed algorithm takes this into account by assigning different weights to the classifiers depending on the probability of each prediction they make. Compared to the static weighted majority vote, described in section 2.3, it performs better, because it better estimates the reliability of the individual classifiers, and it does not make a compromise in speed. Compared to the other approaches for dynamically weighted majority vote [7] and [8] it will definitely perform faster[5], because it does not have to look into the feature space or into the data sets. Our approach uses the probability of the prediction of each classifier, and it uses a pre-calculated weight for the probability interval, that the prediction belongs to. However, we did not test the approaches described in [7] and [8] on this particular problem, so we cannot make comparison of the classification performance.

The proposed method for estimation of the probability of the predictions of ensembles enables comparison of the performance of ensembles with the performances of individual classifiers in terms of AUC ROC. This is very essential, because the incomplete, and possibly inaccurate and inadequate comparisons, based only on accuracy, can be avoided. The proposed method was tested in conjunction with the proposed algorithm for dynamically weighted majority voting, but can be used with other schemes of weighted majority voting.

[5] This is inferred from the complexity of the algorithms: our approach has O(4) (because N=4), and those approaches have O(logN) or worse complexity.

The presented results show that the proposed algorithm is suitable for the particular problem described in section 4, and the particular classifiers that were tested here. However, additional research should be conducted, using different data sets and different ensembles, in order to establish the proposed algorithm as a reliable and justified alternative to the static approaches.

References

[1] Kuncheva, *Combining Pattern Classifiers. Methods and Algorithms* (Wiley, 2004).

[2] T. G. Dietterich, Ensemble methods in machine learning, *Multiple Classifier Systems*, Cagliari, Italy, Springer-Verlag, 2000, pp. 1–15.

[3] T. K. Ho., Multiple classifier combination: Lessons and the next steps, *Hybrid Methods in Pattern Recognition*, World Scientific Publishing, 2002, pp. 171–198.

[4] L. Shapley & B. Grofman, Optimizing group judgemental accuracy in the presence of interdependencies, *Public Choice*, 1984, pp. 329–343.

[5] J. Lang, M. S. Pini, F. Rossi, K. B. Venable & T. Walsh, Winner Determination in Sequential Majority Voting, *Proceedings of the 20th international joint conference on Artifical intelligence*, Morgan Kaufmann Publishers Inc., 2007, pp. 1372-1377.

[6] Charles Ling Jin, Charles X. Ling, Jin Huang & Harry Zhang, AUC: a Statistically Consistent and more Discriminating Measure than Accuracy, *In Proceedings of 18th International Conference on Artificial Intelligence (IJCAI-2003)*, Acapulco, Mexico, 2003, pp. 329-341.

[7] C. De Stefano, A. Della Cioppa & A. Marcelli, An Adaptive Weighted Majority Vote Rule for Combining Multiple Classifiers, *16th International Conference on Pattern Recognition (ICPR'02)* - Volume 2, 2002, pp. 192–195.

[8] A. Gangardiwala & R. Polikar, Dynamically Weighted Majority Voting for Incremental Learning and Comparison of Three Boosting Based Approaches, *Proceedings of International Joint Conference on Neural Networks*, Montreal, Canada, July 31 - August 4, 2005.

[9] PAKDD 2009 Data Mining Competition, http://sede.neurotech.com.br/PAKDD2009

[10] Eftim Zdravevski & Andrea Kulakov, System for Prediction of the Winner in a Sports Game*, ICT Innovations 2009*, Berlin: Springer-Verlag, 2010, pp. 55-64.

[11] Ian H. Witten & Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques* (2nd Edition, Morgan Kaufmann, June 2005).

[12] Raymond Anderson, *The Credit Scoring Toolkit - Theory and Practice for Retail Credit Risk Management and Decision Automation* (Oxford University Press Inc., New York, 2007).