Evaluating the Effect of Voting Methods on Ensemble-Based Classification

Florin Leon, Sabina-Adriana Floria
Faculty of Automatic Control and Computer Engineering
"Gheorghe Asachi" Technical University of Iaşi
Iaşi, Romania
florin.leon@tuiasi.ro, sabina.floria@tuiasi.ro

Costin Bădică
Faculty of Automatics, Computers and Electronics
University of Craiova
Craiova, Romania
cbadica@software.ucv.ro

***Abstract*—Bagging is a popular method used to increase the accuracy of classification, by training a set of classifiers on slightly different datasets and aggregating their output by voting. Usually, the majority voting is used for this purpose, or the plurality voting, when the problem has multiple class values. In this study, we analyze the influence of several voting methods on the performance of two classification algorithms used for datasets with different levels of difficulty. The results reveal that the single transferable vote can be a good alternative to plurality voting, although it has the drawback of a higher computational cost related to the calculation of preference ordering.**

Keywords—bagging; voting; k-nearest neighbor; Naïve Bayes; classification

I. INTRODUCTION

Nowadays, data processing techniques are the main focus of research in various fields, especially since the growth of the Internet and the use of increasingly larger databases. It is hard for a human user to comprehend or to handle such data without dedicated tools or automatic software mechanisms. For such needs, machine learning is commonly employed in many applications such as classification, recommendation systems, pattern recognition, etc.

Ensemble methods have been proven to be a good approach for the improvement of classification accuracy. Among the most popular methods, one can mention bagging, boosting and stacking.

Currently, most classification algorithms that rely on bagging use majority or plurality voting to aggregate the results on the individual ensemble components. While very simple, this may not always be the best approach in case of multiclass problems.

The present article aims at studying the role of different voting methods combined with bagging: we use ensembles of *k*-nearest neighbor and Naïve Bayes classifiers aggregated by four different voting schemes and compare their performance in terms of classification accuracy.

All the algorithms used here have been implemented by the authors; no third-party libraries have been used.

978-1-5090-5795-5/17/$31.00 ©2017 IEEE

We organize our paper as follows. Section II presents some related work in this area, Section III briefly describes bagging and the voting methods under analysis, and Section IV presents the two base classifiers used to assess the performance of bagging. The case studies related to the application of the presented methods on three multi-class classification problems of increasing difficulty are included in Section V, while Section VI presents the conclusions and some ideas for future work.

II. RELATED WORK

Classification methods are probably the most popular of all machine learning classes of algorithms. Therefore, there is a constant interest in devising new techniques that can increase the accuracy of classification.

One idea is by using groups of classifiers instead of individual ones. The best known ensemble methods were introduced with bagging [1] and boosting [2] in which several classifiers were used to produce one single outcome with improved accuracy. However, the classic version only uses the majority or plurality vote to aggregate the outcomes of individual classifiers. That is why research has continued with the study of other voting methods combined with ensemble methods, and some results are presented as follows.

Several voting methods used for testing ensembles of classifiers trained by the bagging procedure are presented in [3]. The classifiers are multi-layer perceptrons (MLPs). Two datasets are used: digits and capital letters, and both were equally divided for training and testing. These datasets were run on different MLP structures (small and large). The results showed that the best voting methods for small MLPs are the sum rule, the product rule and the Borda count. Also, Borda count provided good recognition results on large MLPs.

The performance of certain voting methods was also analyzed in [4], where bagging was used for the reconciliation model, which is a process of combining classification models. The chosen classifier was the classification tree, with the dataset divided into two-thirds for training and one-third for testing. The experimental results show the generalization error performance of bagging using voting methods such as

plurality, anti-plurality, Borda count, plurality with elimination, and Condorcet's pairwise comparison. Several configurations were used for the testing observations: various datasets with and without noise, different sizes of datasets and different number of classes. Anti-plurality and plurality showed better performances when the number of classes was two. In the presence of noise, the Borda count showed better performance than other voting methods. The overall performances show the following order in classification accuracy of the used voting methods: Borda count, Condorcet's pairwise comparison, anti-plurality, plurality, plurality with elimination.

In [5], three voting methods were used with bagging on the MLEM2 rule induction algorithm: support, strength and democratic. The chosen number of LERS (Learning from Examples based on Rough Sets) classifier instances (i.e. bootstrapped data) was 100. On every bootstrap set, the MLEM2 induction rule was used and the resulting rules were then provided as an input for the LERS classifiers. The experiments were based on 16 datasets and the error rates were presented for each rule set used in bagging including the original prediction without aggregation. The results showed the stability of bagging in which the error rate became almost constant when the number of classifiers used reached a certain number (e.g. for the breast cancer dataset the error rate became constant at an ensemble size of 27 with strength rule voting). The democratic voting presented the best overall performance while the voting based on strength had the worst performance.

The results of bagging on text classification is presented in [6]. Binary decision trees were created using the C4.5 algorithm and the efficiency of bagging was compared with the standard classifier. The number of classifiers was limited to 200 and the tests were carried out on two datasets: the Reuter's collection and the Markiza collection. The results showed the precision of bagging and the precision of standard decision tree classifier. For more frequent classes, the bagging method presented better results than the standard one, otherwise vice-versa. Also, bagging had a higher precision for a number of classifiers greater than 20, for the Reuter's collection, and 10, for the Markiza collection.

In [7] the performance of ensemble methods was analyzed using 10% of the KDD CUP '99 dataset, i.e. 494020 records. Data instances have 41 attributes and the classification labels that must be determined are: "normal" for the records with normal behavior and "attack" for those that are considered to be malicious. The malicious attacks can also be classified into three different categories: denial of service attack, user to root attack and remote to local attack. A comparison was made between bagging, AdaBoost and the standard C4.5 algorithm. Bagging and C4.5 obtained similar performances, both with an error of 1.99%, while AdaBoost performed slightly better with an error of 1.95%.

A weighted voting ensemble learning classifier based on differential evolution (DEWVote) was presented in [8]. Compared with bagging, DEWVote did not use the same base classifier, instead it used randomly chosen base instances, for diversification purpose, from the following set of five classifiers: C4.5, Naïve Bayes, Bayesian Nets, *k*-Nearest Neighbor (*k*NN) and ZeroR. The voting system in DEWVote was majority voting but what makes it different is the weighting of outputs from the base classifiers. The differential evolution method was used to optimize the weights of each classifier: high accuracy of an instance would lead to a higher weight of that particular classifier and vice-versa. A comparison was made between bagging, AdaBoost, majority voting and DEWVote using 15 datasets. In both Bagging and AdaBoost, the Naïve Bayes classifier was used. In most cases DEWVote obtained the highest performances, i.e. 13 out of 15 datasets.

III. Ensemble-Based Classification and Voting Methods

The problem of making a good choice out of multiple possible solutions or opinions has lead us to a common agreement called voting. By voting, each voter has the right to express his/her preference for one or more candidates. Evaluating the votes, one can reach to a final decision which is usually in favor of the majority opinion, i.e. the winner will be the most preferred candidate. By analogy, choosing the solution of a problem from a set of several solutions, one can obtain a better solution or even worse if the majority solutions are worse. Our problem is that of classification.

Currently, there are many proposed techniques to construct voting systems: parametric, nonparametric, heuristic, logical, probabilistic methods, etc. For example, one approach of using voting systems with different learning algorithms consists in choosing different classifiers such as decision trees, *k*NN, multilayer perceptron, a.o., where the same training dataset is provided for each of them. Each classifier will obtain different predictions. To get the best prediction, one can use a voting system in which the winning class will be the majority. Using ensembles of different classifiers has the advantage that not all of them will make the same mistake.

Another method is to use a single type of classifier, but on several training datasets. The approach used in this paper is the one in which we have a single classifier and several training datasets, i.e. bagging. In addition to bagging, one can train each of the same classifiers on various feature samples. In particular, this idea is used in the Random Forest algorithm, where each of the constructed decision trees is combined by majority voting and trained not only on different samples, but also on randomly chosen features.

A. Bagging

Bagging (bootstrap aggregating) is a resampling technique that has the original training dataset as an input and aims at obtaining several slightly different datasets from the original one. Let D_0 be the training dataset and D_i be the new training datasets obtained, where $i = 1, \ldots, m$, where m is the desired number of classifiers. The classifiers are denoted by C_i and can use the same algorithm. D_i is obtained by randomly extracting samples from D_0 using a uniform distribution, with replacement, i.e. the sample is not removed from D_0; it can be extracted multiple times for the same set D_i. Each new set is

used to train the classifier C_i. Then, a voting system is used to combine the outputs of the m classifiers.

In terms of sampling probability, each sample has the probability of *not* being selected in one sampling of $P_1 = 1 - 1 / n$, where n is the number of instances in the D_0 dataset. The probability that an instance is not selected at all in a dataset with n instances is $P_n = (1 - 1 / n)^n$. When $n \to \infty$, $P_n \to 1 / e \approx 0.368$. Therefore, if the original dataset has a large number of instances, then any bootstrapped set D_i contains approximately 63% distinct samples from D_0.

The classification with bagging leads to a significant improvement in accuracy compared to the use of a single trained classifier because the obtained training sets D_i have a reduced variance and thus overfitting is reduced. It also has an advantage in dealing with noisy data.

Next, we present some voting methods.

B. *Voting methods*

The plurality method is the simplest method of voting, where each voter has the right to choose only one alternative, i.e. the most preferred one. The winner is the candidate that has the maximum total number of votes. For example, if 45% of voters choose alternative A, 25% choose B and 30% choose C, then the winner is A.

According to Condorcet's jury theorem, if there are m voters that decide by simple majority voting, and each voter has the probability p of making the right decision, than the probability of the entire jury of making the right decision is:

$$p_m = \sum_{i=\lceil m/2 \rceil}^{m} \left(\frac{m!}{(m-i)! \cdot i!} \right) \cdot p^i \cdot (1-p)^{m-i} \qquad (1)$$

Thus, if $p > 0.5$, then $p_m > p$. This means that the ensemble has a higher probability of making the correct decision than any individual voter. When the number of voters increases, the probability of the ensemble to make the right decision also increases. Ideally, when $m \to \infty$, $p_m \to 1$ [9].

Another voting method is the Borda count. The voters have to specify a ranking for all the alternatives ordered by their preferences. Each alternative gets points from each ballot depending on its ranking position. For instance, let's consider the possible alternatives A, B and C, and the following rule for point distribution: position one gets 3 points, 2 points for the second place and 1 point for the last one, respectively. If we have the scenario with 3 votes with preferences $C \succ A \succ B$ and 4 for $A \succ C \succ B$ then we can easily determine the total points for each alternative: A gets $3 \cdot 2 + 4 \cdot 3 = 18$ points, B gets $3 \cdot 1 + 4 \cdot 1 = 7$ points and C gets $3 \cdot 3 + 4 \cdot 2 = 17$ points. In this case, the winner is A. This voting system has the possibility to give an advantage to the alternatives with higher rankings just by modifying the rule of distributed points.

In Copeland's method, the votes are considered to be a set of pairwise rankings of the alternatives. The main idea is to order the alternatives according to their win-loss difference. The final winner is the one with the highest score. To determine wins and losses of a single alternative, the comparison must be done relative to one other candidate at a time. For each comparison between two alternatives, all the ballots must be inspected. For instance let's say we have the same votes and preferences as in the previous example. There are three possible pairwise comparisons between candidates: $\{A, B\}$, $\{A, C\}$, $\{B, C\}$. Comparing the points of each pair, we obtain the following winners: $\{A, B\} = \{7, 0\}$, so the winner is A, $\{A, C\} = \{4, 3\}$, therefore the winner is A again, $\{B, C\} = \{0, 7\}$, so the winner is C. Next, the win-loss difference (WLD) is computed for each candidate. For example, A has 2 wins, against B and C, and 0 losses. Therefore: $WLD_A = 2 - 0 = 2$, $WLD_B = 0 - 2 = -2$ and $WLD_C = 1 - 1 = 0$. Candidate A is the final winner with the most points.

Single transferable vote (STV) is a method in which every voter ranks all the alternatives ordered by preference. After all the votes are tallied, a quota q is set:

$$q = floor\left(\frac{total\ number\ of\ votes}{desired\ winners + 1} \right) + 1 \qquad (2)$$

This is the most common quota, i.e. Droop quota. If a candidate reaches or exceeds the quota, then it is declared to be the winner. Its surplus votes are transferred to the candidates chosen as the second preference from the same ballot using (3), where $V_{transfer}$ is the number of votes to be transferred to the second candidate, V_{second} is number of votes from the second candidate, $V_{total\ winner}$ is the total number of votes from the current winner, and $V_{surplus}$ represents the surplus votes of the winner:

$$V_{transfer} = \left(\frac{V_{\text{second}}}{V_{\text{total winner}}} \right) \cdot V_{surplus} \qquad (3)$$

However there are more variants of transferring the votes: random selection, transfer only to hopeful candidates and others. The process is repeated until the desired number of winners has been reached. If the process gets stuck and none of the remaining candidates does not reach the quota, then it is removed the candidate with the fewest votes and his votes are transferred to others. Let's consider an example with the following preference ordering: 9 votes for $B \succ A \succ D \succ C$, 14 for $B \succ C \succ A \succ D$ and 7 for C. Let's set the desired number of winners to 2. First the quota is computed: $q = (30 / (2 + 1)) + 1 = 11$. In the first round, the number of votes for the first choices are 23 for B and 7 for 7. B is already a winner and its surplus of votes (12) is transferred to A, who gets 5 votes and to C, who gets 7 votes. The process is repeated until the number of desired winners is reached.

For the case studies described in Section V, we used the STV variant where the candidate with simple majority wins, otherwise the alternative with the least number of points is removed and its votes are transferred to the next alternative in its voters' preference ordering, and so on, until one alternative reaches 50% of the votes. In our particular case of classification, only one winner is needed.

IV. Base Classification Algorithms

In order to study the influence of voting algorithms on the performance of the bagging procedure, it is required that the base classification algorithms provide an ordering of their voting options. This is not straightforward in the general case, because, on the one hand, the classifiers need to be able to directly handle multi-class problems, and on the other hand, to output a preference ordering on all individual classes. One could also imagine a way to apply this process to binary classifiers, using well-known methods such as "one against all" and "one against one", but these methods themselves rely on voting to establish the "winning" class.

Given these constraints, we focused our analysis on two classification algorithms, namely the *k*-Nearest Neighbor and Naïve Bayes, which belong to different classification paradigms.

Methods such as neural networks can also be used, given that each class were assigned a distinct output, and the vote preference were interpreted based on the numerical values of those outputs.

It is not so easy to use decision trees for this procedure, because the class response is distributed among different leaves, and by removing the leaves of a certain class there is no guarantee that the tree would still be able to provide a meaningful "second guess".

The *k*-Nearest Neighbor (*k*NN) classifier works by storing the training set and then matching a test instance against all the training instances to assess their similarity, based on a distance function, usually the Euclidian distance. Since the nearest neighbor could be affected by noise, a larger number of neighbors can be used, $k > 1$, and the classification decision is made by majority vote. However, if k is too large, the classification decision can be affected by the instances that belong to a different class than the correct one.

It is still possible to use all the information in the training set by weighting the influence of the neighbors by the inverse of their distance to the test instance. There are different ways to compute the weights, e.g.:

$$w(\mathbf{x}_i, \mathbf{x}_t) = \frac{1}{\left(d(\mathbf{x}_i, \mathbf{x}_t)\right)^2} \tag{4}$$

$$w(\mathbf{x}_i, \mathbf{x}_t) = \frac{1}{d(\mathbf{x}_i, \mathbf{x}_t)} \tag{5}$$

$$w(\mathbf{x}_i, \mathbf{x}_t) = \exp\left(-\alpha \cdot d(\mathbf{x}_i, \mathbf{x}_t)\right) \tag{6}$$

where $\mathbf{x}_i$ is an instance from the training set, $\mathbf{x}_t$ is the testing instance and d is the distance function.

For example, equation (4) was used in many previous studies [10], [11], [12], equation (5) is used in the Weka [13] implementation and equation (6) was reported in cognitive psychology studies [14].

Given the importance of the distance metric for this algorithm, more advanced distance metric learning methods have been proposed [15], [16], that try to adapt the distance function to the data in order to maximize the margin between instances of different classes, or with different values, for regression problems.

In our case studies, equation (4) was used, in order to give more importance to the closer neighbors. For each class, its total weight or preference intensity is computed as the sum of the weights of the training instances that indicate that class:

$$I_t^c = \sum_{i:\, y_i = c} w(\mathbf{x}_i, \mathbf{x}_t) \tag{7}$$

Since all classes are represented in the training set, it is clear that every class will have a non-zero preference intensity.

The Naïve Bayes (NB) method computes the conditional probabilities that a test instance belongs to each class. The result is the class with the maximum probability. In our case, these conditional probabilities are used as preference intensity indicators:

$$I_t^c = P(c \mid \mathbf{x}_t) / P(\mathbf{x}_t) = P(c) \cdot P(\mathbf{x}_t \mid c) \tag{8}$$

One can ignore $P(\mathbf{x}_t)$ because it is the same for all classes and thus it doesn't affect the preference ordering.

The Naïve Bayes method assumes that the attributes of the problem are independent given a class value, and thus $P(\mathbf{x}_t \mid c)$ can be expressed as a product of the probabilities of each attribute value of instance $\mathbf{x}_t$, conditioned by class c.

In the following case studies, the continuous numeric data was discretized prior to the application of the Naïve Bayes algorithm. There are methods to use the probability distribution of data, e.g. the normal distribution, to assess the likelihood of data, however in the general case there is no guarantee that the data follows a well-known distribution. That is why we have chosen the simpler method of using only symbolic data, by discretization.

V. Case Studies

In order to test the effect of voting methods on the ensemble performance, three classification problems were considered, in increasing levels of complexity.

The first problem is the well-known *Iris* problem [17], [18]. The dataset contains 3 classes with 50 instances each, where each class refers to a type of iris plant: setosa, versicolor and virginica. The setosa class is linearly separable from the other two, which in turn are not linearly separable from each other. There are 4 numeric attributes: petal length, petal width, sepal length and sepal width, all expressed in centimeters. There are no missing attributes.

The second one is the so-called *Glass* problem. The classification of different types of glass was motivated by criminological investigation, because at the scene of a crime, the glass left can be used as evidence [19]. The attributes have numerical values, and represent glass properties such as the refractive index, and proportions of different components such as sodium, magnesium, aluminum, silicon, potassium, calcium, barium and iron. The class has 7 values.

The third dataset, called *Letters*, refers to a letter recognition problem [20] . The character images are based on 20 different fonts, where each letter was randomly distorted to produce unique indicators. Each indicator was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 to 15. The problem has 26 classes, corresponding to the letters of the English alphabet.

The Naïve Bayes classifier was applied on data discretized into 10 equal intervals for the *Iris* and *Glass* problems and into 16 values for the *Letters* problem, because the original data contains integers between 0 and 15, and this was the most natural discretization.

The ensembles considered in the following case studies are formed of 10 voters, a smaller number which can better reveal the influence of the voting methods. With more voters, one can expect that the results would be better anyway, regardless of the voting method.

Tables I, II and III show the results obtained by the *k*NN and NB classifiers with different voting methods.

TABLE I. BAGGING RESULTS FOR THE *IRIS* PROBLEM (150 INSTANCES, 3 CLASSES)

Voting method	Number of errors for each classifier	
	***k*NN**	**NB**
Plurality method	2	6
Single transferable vote	2	8
Borda count	2	6
Copeland method	2	6
Simple classifier for the training set	0	6
Simple classifier for cross-validation	6	8

TABLE II. BAGGING RESULTS FOR THE *GLASS* PROBLEM (214 INSTANCES, 7 CLASSES)

Voting method	Number of errors for each classifier	
	***k*NN**	**NB**
Plurality method	24	58
Single transferable vote	20	58
Borda count	43	64
Copeland method	43	64
Simple classifier for the training set	0	59
Simple classifier for cross-validation	79	85

TABLE III. BAGGING RESULTS FOR THE *LETTERS* PROBLEM (20000 INSTANCES, 26 CLASSES)

Voting method	Number of errors for each classifier		
	***k*NN Numeric**	***k*NN Symbolic**	**NB**
Plurality method	372	421	4534
Single transferable vote	342	451	4421
Borda count	2483	3228	4895
Copeland method	2242	3522	4996
Simple classifier for the training set	0	0	4985
Simple classifier for cross-validation	5751	7866	5252

For each of the four voting methods under consideration, the tables contain the number of errors made by the ensemble on the whole dataset. Also, the performance of the simple classifiers (i.e. the classic variant of each single classifier) is included, as a means for comparison, both on the training set alone, and for 10 fold cross-validation.

In case of the *Letters* problem, the attributes are integers and they can be interpreted by the *k*NN algorithm both as numerical values, where the distance is computed between their normalized values, and as symbolic values, with a distance of 1 if the values are different and 0 if the attribute values are the same. Thus, there are two columns in Table III that correspond to these *k*NN cases.

We should stress the fact that the goal of our study is to assess the influence of different voting methods, not to obtain an error rate as small as possible on the datasets. Especially for the simple problems, other classification algorithms may be more appropriate, e.g. decision trees or random forests that also rely on bagging and voting. However, as we explained in Section IV, it is not straightforward to customize these algorithms to express preference ordering, as required by the voting schemes employed here.

Fig. 1 presents a comparison between the methods, in terms of the number of errors obtained:

$$p_e = n_e / n_{cv}, \tag{7}$$

where n_e is the number of errors obtained by a certain ensemble and n_{cv} is the number of errors obtained by the single classifier for 10 fold cross-validation. Although expressed as a percentage, p_e actually expresses the accuracy improvement obtained by an ensemble of voters in comparison with a single classifier without any voting. When p_e is 1, no improvement is obtained. The lower p_e is, the better an ensemble method performs.

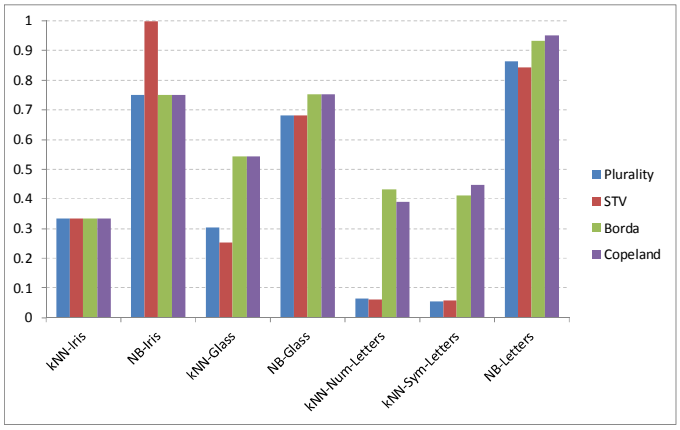

Fig. 1. Accuracy improvement by using different voting methods

In general, *k*NN seems to be better than Naïve Bayes for the simpler problems. However, in case of the *Letters* problem, NB can use the higher amount of information about the probability distribution of data and thus can give better generalization results than *k*NN. Since NB is less prone to overfitting, the error rates are comparable on the training set

results and on cross-validation results. Since *k*NN is less stable, it has the most to gain from being used in an ensemble. The error rate is much smaller for *k*NN, regardless of the voting method, than for the cross-validation case.

Still, plurality and single transferable vote appear to bring the most benefits to both classification algorithms.

VI. CONCLUSIONS

Although the plurality method is the easiest and generally gives good results, these experiments show that also the single transferable vote method can sometimes be a better choice. The main difficulty lies in the computation of the preference ordering, which is time-consuming compared to the simple vote count. Some methods can be naturally used to express such an ordering, while in other cases this is not straightforward.

However, we can conclude that, when possible, the *single transferable vote* method can be a good alternative to the *plurality* voting method that is generally used with ensemble-based classification.

A natural direction for future investigation is the use of other classification algorithms and other classification problems, in order to assess more clearly the influence of different voting methods on bagging. Also, since the number of bootstrapped training sets is small, it is important to make a thorough statistical analysis regarding the performance of the voting methods.

REFERENCES

[1] L. Breiman, "Bagging predictors", Machine Learning, vol.24, pp. 123-140, 1996.

[2] R.E. Schapire, Y. Singer "Improved boosting algorithms using confidence-rated predictions", Machine Learning, vol. 37, pp. 297-336, 1999.

[3] M. Van Erp, L. Vuurpijl, L. Schomaker, "An overview and comparison of voting methods for pattern recognition", IWFHR '02 Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition, pp. 195-200, 2002.

[4] K.T. Leung, D.S. Parker, "Empirical comparisons of various voting methods in bagging", KDD '03 Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 595-600, 2003.

[5] C. Cohagan, J.W. Grzymala-Busse, Z.S. Hippe, "A comparison of three voting methods for bagging with the MLEM2 algorithm", IDEAL'10 Proceedings of the 11th international conference on Intelligent data engineering and automated learning, pp. 118-125, 2010.

[6] K. Machová, F. Barčák, P. Bednár, "A bagging method using decision trees in the role of base classifiers", Acta Polytechnica Hungarica, vol.3, pp. 121-132, April 2006.

[7] R.D. Kulkarni, "Using esemble methods for improving classification of the KDD CUP'99 data set", IOSR Journal of Computer Engineering (IOSR-JCE), vol. 16, pp. 57-61, 2014.

[8] Y. Zhang, H. Zhang, J. Cai, B. Yang, "A weighted voting classifier based on differential evolution", Abstract and Applied Analysis, 2014.

[9] T. Saito, "Theoretical Model: Condorcet's Jury Theorem, Part 1", Wolfram Demonstrations Project, http://demonstrations.wolfram.com/TheoreticalModelCondorcetsJuryTheoremPart1, 2017.

[10] F. Leon, C.G. Piuleac, S. Curteanu, I. Poulios, "Instance-based regression with missing data applied to a photocatalytic oxidation process", Central European Journal of Chemistry, vol. 10, no. 4, pp. 1149-1156, 2012.

[11] F. Leon, C. Lisa, S. Curteanu, "Prediction of the liquid crystalline property using different classification methods", Molecular Crystals and Liquid Crystals, vol. 518, pp. 129-148, 2010.

[12] F. Leon, S. Curteanu, C. Lisa, N. Hurduc, "Machine learning methods used to predict the liquid-crystalline behavior of some copolyethers", Molecular Crystals and Liquid Crystals, vol. 469, pp. 1-22, Taylor and Francis Group, USA, 2007, DOI: 10.1080/15421400701431232.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I.H. Witten, "The WEKA data mining software: An Update", ACM SIGKDD Explorations, vol. 11, no. 1, pp. 10–18, 2009.

[14] R.N. Shepard, "Toward a universal law of generalization for psychological science", Science, vol. 237, pp. 1317–1323, 1987.

[15] K.Q. Weinberger and L.K. Saul, "Distance metric learning for large margin nearest neighbor classification", Journal of Machine Learning Research, vol. 10, pp. 207–244, 2009.

[16] F. Leon, S. Curteanu, "Large margin nearest neighbour regression using different optimization techniques", Journal of Intelligent & Fuzzy Systems, vol. 32, pp. 1321-1332, 2017, DOI:10.3233/JIFS-169130.

[17] R. A. Fisher, "The use of multiple measurements in taxonomic problems", Annual Eugenics, vol. 7, part II, pp. 179-188, 1936.

[18] R. O. Duda, P. E. Hart," Pattern classification and scene analysis", John Wiley & Sons, p. 218,1973.

[19] I.W. Evett and E.J. Spiehler, "Rule induction in forensic science", Central Research Establishment. Home Office Forensic Science Service. Aldermaston, Reading, Berkshire RG7 4PN.

[20] P. W. Frey and D. J. Slate, "Letter recognition using holland-style adaptive classifiers", Machine Learning, vol 6, 1991.