Increasing Diversity in Random Forests Using Naive Bayes

Christos K. Aridas, Sotiris B. Kotsiantis, Michael N. Vrahatis

Increasing diversity in Random Forests using Naive Bayes

Christos K. Aridas, Sotiris B. Kotsiantis, and Michael N. Vrahatis

Department of Mathematics
University of Patras
GR-26110 Patras, Greece
{char,sotos,vrahatis}@math.upatras.gr

Abstract. In this work a novel ensemble technique for generating random decision forests is presented. The proposed technique incorporates a Naive Bayes classification model to increase the diversity of the trees in the forest in order to improve the performance in terms of classification accuracy. Experimental results on several benchmark data sets show that the proposed method archives outstanding predictive performance compared to other state-of-the-art ensemble methods.

Keywords: ensemble methods; decision forests; pattern classification

1 Introduction

In machine learning and data mining, ensemble methods make use of single or multiple learning algorithms in order to generate a diverse set of classifiers aiming to improve performance / robustness over a single underlying classifier [16]. Experimental studies and machine learning applications prove that a certain supervised learning algorithm outperforms any other algorithm for a particular problem or for a particular subset of the input dataset, but it is unusual to discover a single classifier that will reach the best performance on the overall problem domain [17].

Ensembles of classifiers can be generated via several methods [2]. Common procedures that are used to create an ensemble of classifiers include, among others, i) Using different splits of a training data set with a single learning algorithm, ii) Using different training parameters with a single learning algorithm, iii) Using multi-learning methods.

Diversity [21] between the base classification models is considered to be a key aspect when constructing a classifier ensemble. In this work, we propose a variation of the Random Forests [6] algorithm that incorporates new features using Naive Bayes [8] before the construction of the forest. The new generated features aim to increase the diversity among the trees in the forest. Our empirical evaluation concludes that the new features increase the diversity and that they leed to a better final classifier.

The rest of the paper is organized as follows. In Section 2 some of the most well-known techniques for generating ensembles, that are based on a single learning algorithm, are discussed. In Section 3 the proposed method is presented. Furthermore, the results of the experiments on several real and laboratory data sets, after being compared with state-of-the-art ensemble methods, are portrayed and discussed. Finally, Section 4 concludes the paper and suggests further directions in current research.

2 Background Material

This section presents a brief survey of techniques for generating ensembles using a sole learning algorithm. These techniques rely on modifying the training data set. Methods of modifying the training data set include, among others, sampling the training patterns, sampling the feature space, a combination of the two and modifying the weight of the training patterns.

Bagging is a method for creating an ensemble of classifiers that was proposed by Breiman [5]. Bagging generates the classifiers in the ensemble by taking random subsets of the training data set with replacement and building one classifier on each bootstrap sample. The final classification prediction for an unseen pattern is constructed by taking the majority vote over the class labels produced by the base classification models.

While Bagging relies on random and independent changes in the training data implemented by bootstrap sampling, Boosting [11] encourages guided changes of the training data to direct further classifiers toward more "difficult cases". It assigns weights to the training patterns, which are then modified according to how well the coupled case is learned by the classifier. The weights for misclassified patterns are increased. Thus, re-sampling happens based on how well the training patterns are classified by the previous base classifier. Given that the training set for one classification model depends on the previous one, boosting requires sequential runs and therefore is not easily adapted to a parallel process. After several iterations, the prediction is made by taking a weighted vote of the predictions of each classifier, with the weights being relative to each classifiers accuracy on its training set. AdaBoost is a practical version of the boosting approach [11].

Ho [15] constructed a forest of decision trees named Random Subspace Method that preserves highest accuracy on the training patterns and improves on generalization accuracy as it grows in complexity. Random Subspace Method consists of global multiple decision trees created systematically by pseudorandomly selecting half of the available features, i.e. trees built in randomly chosen subspaces. The final classification prediction for an unseen pattern is constructed by averaging the estimates of posterior probabilities at the leaves of all the trees in the forest.

Random Forests [6] is an alternate method for building ensembles. It is a combination of Bagging and Random Subspace Method. In Random Forests, every tree in the forest is constructed from a bootstrapped sample from the

training set. Additionally, the split that is selected in tree construction is not the best split between all the available features. Instead, it is the best split among a random selection of the features [22].

Despite the fact that Random Forests yields one of the most successful [9] classification models, the improvement of its classification accuracy remains an open problem for the machine learning research field. Several authors [24], [3], [19] have studied and proposed techniques that could improve the performance of Random Forests. In [19] it is illustrated that the most important improvement in the performance of Random Forests is achieved by changing the mechanism of voting in the prediction. In [24] the authors implemented a similar approach, where class votes by trees in the forest are weighed according to their performance; Therefore, heavier weights are assigned to better performing trees. The authors of [3] experimentally show that the classification accuracy is enhanced when a random forest is composed of good and uncorrelated trees with high accuracies, while correlated and bad trees with low classification accuracies are ignored.

3 The Proposed Method

In this work, a modified version of Random Forests is proposed that is constructed not only by pseudorandomly selecting attribute subsets but also by encapsulating Naive Bayes estimation in the training phase, as well as in the classification phase.

Given a class variable y and a dependent feature vector x_1 through x_n, Bayes theorem states the following relationship:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{1}$$

Under the assumption that features are conditionally independent

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y), \tag{2}$$

for all i the 2 is simplified to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{3}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input the formula used by the Naive Bayes classifier is

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \Rightarrow \hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{4}$$

The assumption of independence is almost always wrong. Besides this, an extensive comparison of a simple Bayesian classifier with state-of-the-art algorithms showed that the former sometimes is superior to other supervised learning algorithms even on datasets with important feature dependencies [8]. In [12] Friedman explans why the simple Bayes method remains competitive.

Our intention is to generate a forest of decision trees that will be as diverse as possible for producing better results [4]. For this reason, in the training phase, we trained a classifier using the Naive Bayes algorithm. Afterwards, we used the same training set to generate predictions and class membership probabilities using the Naive Bayes classifier. The predictions and the class membership probabilities will increase the original feature space by concatenating them as new features. As far as predictions are concerned, a new feature vector will be generated. This vector will contain only the class label for each instance that is predicted by the Naive Bayes model. In the case of class membership probabilities, new feature vectors will be generated, as many as the number of classes. Assuming that we have a sample of a dataset with four features as presented in Table 1 and three classes. An example of the above process is illustrated in Tables 2 and 3. After the concatenation, a Random Forests classifier will be trained using the new n-dimensional feature vector. The same procedure, the generation of new feature vectors, will be used in the classification phase. The predicted class of an unseen instance will be the vote by the trees in the forest, weighted by their probability estimates. The proposed method is presented in Algorithm 1.

Table 1. Original feature space

x_0	x_1	x_2	x_3	y
4.7	3.2	1.3	0.2	0
4.6	3.1	1.5	0.2	0
6.4	3.2	4.5	1.5	1
6.9	3.1	4.9	1.5	1
5.8	2.7	5.1	1.9	2
7.1	3.0	5.9	2.1	2

Table 2. Feature space augmented with Naive Bayes model's predictions

x_0	x_1	x_2	x_3	f_0	y
4.7	3.2	1.3	0.2	0	0
4.6	3.1	1.5	0.2	0	0
6.4	3.2	4.5	1.5	1	1
6.9	3.1	4.9	1.5	2	1
5.8	2.7	5.1	1.9	2	2
7.1	3.0	5.9	2.1	2	2

Table 3. Feature space augmented with Naive Bayes model's class membership probabilities

x_0	x_1	x_2	x_3	f_1	f_2	f_3	y
4.7	3.2	1.3	0.2	1.000	0.000	0.000	0
4.6	3.1	1.5	0.2	1.000	0.000	0.000	0
6.4	3.2	4.5	1.5	0.000	0.945	0.055	1
6.9	3.1	4.9	1.5	0.000	0.456	0.544	1
5.8	2.7	5.1	1.9	0.000	0.025	0.975	2
7.1	3.0	5.9	2.1	0.000	0.000	1.000	2

Algorithm 1 NB Forest

procedure Training(X, y, $generateNBProbas$, $generateNBPredictions$)
  Build a Naives Bayes (NB) model using X and y
  $X_{gen} \leftarrow GenerateFeatures(X, generateNBProbas, generateNBPredictions)$
  Build a Random Forest model using X_{gen} and y
end procedure
procedure Classification(X, $generateNBProbas$, $generateNBPredictions$)
  $X_{gen} \leftarrow GenerateFeatures(X, generateProbabilities, generatePredictions)$
  Use the X_{gen} to classify a test instance
end procedure
function GenerateFeatures(X, $generateNBProbas$, $generateNBPredictions$)
  $X_{gen} \leftarrow X$
  if $generateNBProbas$ **then**
    Use NB to generate class membership probabilities as X_{probas}
    $X_{gen} \leftarrow X_{gen} \cup X_{probas}$
  end if
  if $generateNBPredictions$ **then**
    Use NB to generate class predictions as $X_{predictions}$
    $X_{gen} \leftarrow X_{gen} \cup X_{predictions}$
  end if
  return X_{gen}
end function

3.1 Numerical Experiments

In order to verify the performance of the proposed method, a number of experiments on some classification tasks were conducted and the results are reported in this section. From the KEEL data set repository [1] fourteen data sets were chosen and used as is without any further preprocessing. In Table 4 the name, the number of patterns, the attributes, as well as the number of different classes for each data set are shown.

All data sets were partitioned using a ten-fold cross-validation procedure. This method divides the instances in ten equal folds. Each tested method was

Table 4. Benchmark data sets used in the experiments

Data Set	#attributes	#patterns	#classes
appendicitis	7	106	2
banana	2	5300	2
cleveland	13	297	5
ecoli	7	336	8
glass	9	214	7
led7digit	7	500	10
libras	90	360	15
phoneme	5	5404	2
ring	20	7400	2
segment	19	2310	7
spambase	57	4597	2
texture	40	5500	11
twonorm	20	7400	2
yeast	8	1484	10

trained using nine folds and the fold left out was used for evaluation, using the metric of classification accuracy. This was repeated ten times. Then the average accuracy across all trials was computed.

Firstly, we ran experiments using different settings of the proposed method and compared them to the original Random Forests (RF) algorithm. The experiments were performed in python using the scikit-learn [18] library. All classifiers were built using the default settings in scikit-learn, which means that all ensembles were generated using ten base classifiers.

In Table 5 the results obtained using variants of the proposed method is presented. The variant that is better than the original Random Forests is reported in bold.

NBFB denotes a Random Forests classifier that was trained using the original space along with class membership probabilities generated by the Naive Bayes model. NBFD denotes a Random Forests classifier that was trained using the original space along with predictions generated by Naive Bayes model. NBFBD was trained using both class membership probabilities and predictions that were generated by the Naive Bayes model, concatenated with the original feature space. With the intention to discard the parameters of the proposed method, i.e. which new features should be generated by the Naive Bayes model, and get the most out of the generated features, we ran an experiment by selecting the best of RF, NBFB, NBFD and NBFBD using 5-fold cross-validation in the training set. The model selection was performed by selecting the variant that gave the minimum average error rate across all folds. The last variant is denoted as NBFCV. In the last row of the Table 5 counted the W(ins)/T(ies)/L(osses) of the algorithm in the column against the original Random Forests algorithm obtained by the Wilcoxon Singed Ranks Tests [23]. Apart from the combined methodology, it is clear that almost every variation of the proposed method performs better that the original Random Forests method in most cases.

Table 5. Average accuracy of Random Forests variants

Data Set	RF	NBFB	NBFD	NBFBD	NBFCV
appendicitis	0.85000	**0.87000**	**0.87909**	0.85000	**0.86909**
banana	0.89000	0.88830	**0.88962**	**0.89094**	**0.89170**
cleveland	0.54555	**0.54624**	**0.60362**	**0.57051**	**0.57336**
ecoli	0.81569	0.79795	**0.83957**	0.80704	**0.82487**
glass	0.74536	0.71355	0.74304	0.73080	**0.74575**
led7digit	0.70600	**0.71400**	0.70800	0.70200	**0.70800**
libras	0.78056	0.77500	**0.78611**	0.76111	**0.79722**
phoneme	0.89415	**0.89433**	**0.89896**	0.89026	**0.89637**
ring	0.92946	**0.97824**	**0.97892**	**0.97865**	**0.97905**
segment	0.97229	0.97229	**0.97662**	0.96667	**0.97576**
spambase	0.94736	0.94649	**0.94758**	0.94562	**0.94758**
texture	0.96709	0.96709	**0.97073**	0.96473	**0.96964**
twonorm	0.94054	**0.97689**	**0.97865**	**0.97878**	**0.97865**
yeast	0.57277	0.56940	**0.58159**	**0.58828**	**0.58426**
Statistic		-0.235	-2.919	-0.035	-3.296
p-value		0.814	0.004	0.972	0.001
W/T/L		6/2/6	12/0/2	5/1/8	14/0/0

We chose NBFCV and RF and we measured the diversity using two artificial data sets.The first data set was generated by taking a multi-dimensional standard normal distribution and defining classes separated by nested concentric multi-dimensional spheres, so that, roughly, equal numbers of samples are in each class (quantiles of the χ^2 distribution). The second data set is a binary classification problem used in [14]. The data set has ten features that are sampled from standard independent Gaussian and the target class y is defined by:

$$Y = \begin{cases} 1, & \text{if } \sum_{i=1}^{10} X^2 > 9.34 \\ -1, & \text{otherwise.} \end{cases}$$

In Table 6 the name, the number of patterns, the attributes as well as the number of different classes for each data set are shown.

Table 6. Artificial data sets for measuring diversity

Data Set	#attributes	#patterns	#classes
gaussian quantiles	10	1000	2
hastie	10	12000	2

We used the five-fold cross-validation procedure and measured the diversity in each test fold using the Kohavi-Wolpert Variance [21]. In Table 7 the mean of the Kohavi-Wolpert Variance across all testing folds is presented. The diversity increases as the variance decreases, so the lower the better. In parentheses the

mean accuracy score across all folds is reported. The results of Table 7 indicate that involving Naive Bayes for feature generation makes the ensemble more diverse and this results in an increment of classification accuracy.

Table 7. Average measurement of diversity using Kohavi-Wolpert Variance

Data Set	RF	NBFCV
gaussian quantiles	0.159 (0.775)	0.040 (0.945)
hastie	0.074 (0.855)	0.004 (0.990)

Afterwards, we trained models using different ensemble methods that were presented in Section 2. We trained a Naive Bayes (NB) classifier, a Bagging classifier that used Decision Tree (BGT) as a base learning algorithm and a Bagging classifier that used Naive Bayes(BGN) as a base learning algorithm. Also, we trained Boosting (AdaBoost) and Random Space Method ensembles using Decision Tree (BST, RT) and Naive Bayes (BSN, RN) as base learners. Finally, we trained a voting classifier, denoted as VTN, using Random Forests and Naive Bayes that predicts the average of class membership probabilities. All ensemble methods were built by using ten base classifiers apart from Boosting methods that were built by using fifty base classifiers.

Table 8. Average accuracy of compared algorithms

Data Set	NBFCV	RF	NB	BGT	BGN	BST	BSN	RT	RN	VTN
appendicitis	0.869	0.850	0.850	0.860	0.851	0.804	0.743	0.879	**0.887**	0.860
banana	**0.892**	0.890	0.613	0.888	0.614	0.747	0.600	0.585	0.605	0.889
cleveland	**0.573**	0.546	0.553	0.549	0.530	0.452	0.493	0.566	0.559	0.543
ecoli	**0.825**	0.816	0.589	0.795	0.426	0.780	0.601	0.738	0.631	0.688
glass	**0.746**	0.745	0.448	0.732	0.442	0.663	0.581	0.735	0.444	0.615
led7digit	0.708	0.706	0.672	**0.712**	0.590	0.698	0.662	0.690	0.666	0.686
libras	**0.797**	0.781	0.631	0.758	0.639	0.722	0.675	0.747	0.633	0.650
phoneme	0.896	0.894	0.761	**0.905**	0.762	0.872	0.561	0.813	0.760	0.845
ring	0.979	0.929	0.980	0.927	**0.980**	0.874	**0.980**	0.936	0.970	**0.980**
segment	**0.976**	0.972	0.798	0.975	0.798	0.964	0.851	0.951	0.750	0.894
spambase	**0.948**	0.947	0.821	0.938	0.822	0.944	0.721	0.945	0.795	0.893
texture	**0.970**	0.967	0.774	0.961	0.773	0.931	0.855	0.975	0.772	0.839
twonorm	**0.979**	0.941	**0.979**	0.937	**0.979**	0.843	0.976	0.937	0.970	0.977
yeast	**0.584**	0.573	0.141	0.570	0.183	0.503	0.350	0.513	0.406	0.228

The results obtained are presented in Table 8. In the comparisons, we include the NBFCV variant because it performed better against the original Random Forests. According to 8 the proposed method seems to perform better than the well-known ensemble methods.

Demšar [7] recommends that the non-parametric tests should be preferred over the parametric in the context of machine learning, since they do not assume

normal distributions or homogeneity of variance. Hence, in order to validate the significance of the results, the Friedman test [13], which is a rank-based non-parametric test for comparing several machine learning algorithms on multiple data sets, was used. The null hypothesis of the test states that all the methods perform equivalently and thus their ranks should be equivalent. The average rankings, according to the Friedman test, are presented in Table 9.

Table 9. Average rankings of the algorithms according to the Friedman test

Algorithm	Ranking
NBFCV	1.7143
RF	3.7500
BGT	4.0714
RT	4.6786
VTN	5.5714
BST	6.0000
NB	7.1071
RN	7.2143
BGN	7.3571
BSN	7.5357
Statistic	49.5896
p-value	$<10^{-6}$

Friedman's test ranks our algorithm in the first place. Besides this, assuming a significance level of 0.05 in Table 9, the p-value of the Friedman test implies that the null hypothesis has to be rejected. So, there is at least one method that performs statistically different from the proposed method. In order to investigate the aforesaid, Finner's [10] hoc procedure was used.

Table 10. Post hoc comparison for the Friedman Test

i	Algorithm	$z = (R_0 - R_i)/SE$	p	Finner
9	BSN	5.087130	$<10^{-5}$	0.005683
8	BGN	4.931083	$<10^{-5}$	0.011334
7	RN	4.806246	$<10^{-5}$	0.016952
6	NB	4.712618	$<10^{-5}$	0.022539
5	BST	3.745127	0.000180	0.028094
4	VTN	3.370614	0.000705	0.033617
3	RT	2.590379	0.009587	0.039109
2	BGT	2.059820	0.039416	0.044570
1	RF	1.778935	0.075250	0.050000

In Table 10 the p-values obtained by applying post hoc procedure, over the results of the Friedman statistical test, are presented. Finner's procedure rejects the hypotheses that have unadjusted p-values ≤ 0.05. That said, the adjusted

p-values obtained through the application of the Finner's post hoc procedure are presented in Table 11.

The results obtained by Tables 9, 11 indicate that the proposed method performs better than any other method. Nonetheless, when involving other algorithms in the comparisons the proposed variant does not seem to perform statistically better than the original Random Forest method.

Table 11. Post hoc comparison for the Friedman Test with adjusted p-values

i	Algorithm	$p_{Unadjusted}$	p_{Finner}
9	BSN	$<10^{-5}$	$<10^{-5}$
8	BGN	$<10^{-5}$	$<10^{-5}$
7	RN	$<10^{-5}$	$<10^{-5}$
6	NB	$<10^{-5}$	$<10^{-5}$
5	BST	0.000180	0.000325
4	VTN	0.000750	0.001125
3	RT	0.009587	0.012309
2	BGT	0.039416	0.044232
1	RF	0.075250	0.075250

4 Conclusions and Future Work

An ensemble of classifiers is a collection of classification models whose individual predictions are blended, typically by weighted or unweighted voting, to assign a class label to each new pattern. The creation of effective ensemble methods is an active research field in machine learning. Ensembles of classifiers are usually significantly more accurate than the individual underlying classifiers. The main explanation is that many learning algorithms apply local optimization techniques, which may get stuck in local optima.

It was demonstrated after a number of comparisons with Random Forests and other well-known ensembles, that increasing the feature space of a small random forest with predictions and class membership probabilities of a Naive Bayes model can increase the performance in terms of classification accuracy, in most cases.

In a following work, the proposed method will be investigated as far as regression problems and the problem of choosing the right number of trees in the forest are concerned. Also, the implementation and the evaluation of the proposed method in on-line learning [20] problems will be addressed.

References

1. Alcala-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., Garcia, S.: KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Multiple-Valued Logic and Soft Computing 17(2-3), 255–287 (2011)

2. Bernardini, F., Monard, M., Prati, R.: Constructing ensembles of symbolic classifiers. p. 6 pp. IEEE (2005)
3. Bharathidason, S., Jothi Venkataeswaran, C.: Improving Classification Accuracy based on Random Forest Model with Uncorrelated High Performing Trees. International Journal of Computer Applications 101(13), 26–30 (Sep 2014)
4. Biau, G.: Analysis of a random forests model. The Journal of Machine Learning Research 13(1), 1063–1095 (2012)
5. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (Aug 1996)
6. Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001)
7. Demar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. 7, 1–30 (2006)
8. Domingos, P., Pazzani, M.: On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. Machine Learning 29(2), 103–130 (1997)
9. Fernndez-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? Journal of Machine Learning Research 15, 3133–3181 (2014)
10. Finner, H.: On a Monotonicity Problem in Step-Down Multiple Test Procedures. Journal of the American Statistical Association 88(423), 920–923 (Sep 1993)
11. Freund, Y., Schapire, R.E., others: Experiments with a new boosting algorithm. In: ICML. vol. 96, pp. 148–156 (1996)
12. Friedman, J.H.: On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. Data Mining and Knowledge Discovery 1(1), 55–77 (1997)
13. Friedman, M.: The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. Journal of the American Statistical Association 32(200), 675 (Dec 1937)
14. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics, Springer New York, New York, NY (2009)
15. Ho, T.K.: The random subspace method for constructing decision forests. Pattern Analysis and Machine Intelligence, IEEE Transactions on 20(8), 832–844 (Aug 1998)
16. Kotsiantis, S.: Combining bagging, boosting, rotation forest and random subspace methods. Artificial Intelligence Review 35(3), 223–240 (Mar 2011)
17. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research pp. 169–198 (1999)
18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, d.: Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825–2830 (Nov 2011)
19. Robnik-ikonja, M.: Improving Random Forests. In: Machine Learning: ECML 2004, vol. 3201, pp. 359–370. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
20. Saffari, A., Leistner, C., Santner, J., Godec, M., Bischof, H.: On-line Random Forests. pp. 1393–1400. IEEE (2009)
21. Tang, E.K., Suganthan, P.N., Yao, X.: An analysis of diversity measures. Machine Learning 65(1), 247–271 (Oct 2006)
22. Verikas, A., Gelzinis, A., Bacauskiene, M.: Mining data with random forests: A survey and results of new tests. Pattern Recognition 44(2), 330–349 (Feb 2011)
23. Wilcoxon, F.: Individual Comparisons by Ranking Methods. Biometrics Bulletin 1(6), 80 (Dec 1945)
24. Winham, S.J., Freimuth, R.R., Biernacka, J.M.: A weighted random forests approach to improve predictive performance. Statistical Analysis and Data Mining 6(6), 496–505 (Dec 2013)