An Outlier Detection-based Tree Selection Approach to Extreme Pruning of Random Forests

Khaled Fawagreh[a], Mohamed Medhat Gaber[a], Eyad Elyan[a]

[a]*School of Computing Science and Digital Media*
Robert Gordon University
{k.fawagreh, m.gaber1, e.elyan} @rgu.ac.uk
Riverside East, Garthdee Road,
Aberdeen, AB10 7GJ, UK

Abstract

Random Forest (RF) is an ensemble classification technique that was developed by Breiman over a decade ago. Compared with other ensemble techniques, it has proved its accuracy and superiority. Many researchers, however, believe that there is still room for enhancing and improving its performance in terms of predictive accuracy. This explains why, over the past decade, there have been many extensions of RF where each extension employed a variety of techniques and strategies to improve certain aspect(s) of RF. Since it has been proven empirically that ensembles tend to yield better results when there is a significant diversity among the constituent models, the objective of this paper is twofolds. First, it investigates how an unsupervised learning technique, namely, Local Outlier Factor (LOF) can be used to identify diverse trees in the RF. Second, trees with the highest LOF scores are then used to produce an extension of RF termed *LOFB-DRF* that is much smaller in size than RF, and yet performs at least as good as RF, but mostly exhibits higher performance in terms of accuracy. The latter refers to a known technique called ensemble pruning. Experimental results on 10 real datasets prove the superiority of our proposed extension over the traditional RF. Unprecedented pruning levels reaching as high as 99% have been achieved at the time of boosting the predictive accuracy of the ensemble. The notably high pruning level makes the technique a good candidate for real-time applications.

Keywords: Random Forest, Local Outlier Factor, Diversity, Clustering, Ensemble Pruning

1. Introduction

Ensemble classification is an application of ensemble learning to boost the accuracy of classification. Ensemble learning is a supervised machine learning paradigm where multiple models are used to solve the same problem [1] [2] [3]. Since single classifier systems have limited predictive performance [4] [1] [5] [2], ensemble classification was developed to yield better predictive performance [1] [5] [2]. In such an ensemble, multiple classifiers are used. In its basic mechanism, majority voting is then used to determine the class label for unlabeled instances where each classifier in the ensemble is asked to predict the class label of the instance being considered. Once all the classifiers have been queried, the class that receives the greatest number of votes is returned as the final decision of the ensemble.

Three widely used ensemble approaches could be identified, namely, boosting, bagging, and stacking. Boosting is an incremental process of building a sequence of classifiers, where each classifier works on the incorrectly classified instances of the previous one in the sequence. AdaBoost [6] is the representative of this class of techniques. However, AdaBoost is proned to overfitting. The other class of ensemble approaches is the Bootstrap Aggregating (Bagging) [7]. Bagging involves building each classifier in the ensemble using a randomly drawn sample of the data with replacement, having each classifier give an equal vote when labeling unlabeled instances. Bagging is known to be more robust than boosting against model overfitting. Random Forest (RF) is the main representative of bagging [8]. Stacking (sometimes called stacked generalization) extends the cross-validation technique that partitions the data set into a held-in data set and a held-out data set; training the models on the held-in data; and then choosing whichever of those trained models performs best on the held-out data. Instead of choosing among the models, stacking combines them, thereby typically getting performance better than any single one of the trained models [9]. Stacking has been successfully used in both supervised learning tasks (regression) [10], and unsupervised learning (density estimation) [11].

The ensemble method that is relevant to our work in this paper is RF. RF has been proved to be the state-of-the-art ensemble classification technique. Since RF algorithms typically build between 100 and 500 trees [12], it would be useful to reduce the number of trees participating in majority voting

and yet achieving better performance both in terms of accuracy and speed. In this paper, we propose an unsupervised learning approach to improve speed and accuracy of RF. For speed, our approach avoids having all trees participate in majority voting as only a small subset of the trees is selected. For accuracy, since it has been proven empirically that ensembles tend to yield better results when there is a significant diversity among the models [3] [13] [14] [15], our approach ensures that diverse trees in the ensemble are selected. We adopted Local Outlier Factor for tree diversification. Hence, the method is termed Local Outlier Factor Based Diversified Random Forest (both *LOFB-DRF* and *LOF-DRF* are used interchangeably) .

This paper is organized as follows. First we discuss related work in Section 2. This is followed by Section 3 where the motivation and an introduction to RF are covered. Section 4 describes the Local Outlier Factor that will be utilized in our proposed extension of RF. Section 5 formalizes our proposed method and corresponding algorithm. Experimental study demonstrating the superiority of the proposed technique over the traditional RF is detailed in Section 6. The paper is then concluded with a summary and pointers to future directions in Section 7.

2. Related Work

Several attempts have been made in recent years in order to produce a subset of an ensemble that performs as well as, or better than, the original ensemble. The purpose of ensemble pruning is to search for such a good subset. This is particularly useful for large ensembles that require extra memory usage, computational costs, and occasional decreases in effectiveness. Grigorios et al. [16] recently amalgamated a survey of ensemble pruning techniques where they classified such techniques into four categories: ranking based, clustering based, optimization based, and others. Ranking based methods, that are relevant to us in this paper, are conceptually the simplest. Since using the predictive performance to rank models is too simplistic and does not yield satisfying results [17] [18], ranking based methods employ an evaluation measure to rank models. Kappa statistic measure κ was used in [19] for pruning AdaBoost ensembles. For bagging ensembles, however, kappa has proven to be non-competitive [20]. For bagging ensembles, [21] developed an efficient and effective pruning method based on orientation ordering where the classifiers obtained from bagging are reordered and a subset is selected for aggregation.

An interesting issue that remains after ranking the models is to determine the models that will be chosen to form the pruned ensemble. For this, two approaches can be used. The first approach is to use a fixed user-specified amount or percentage of models. A second approach is to dynamically select the size based on the evaluation measure or the predictive performance of ensembles of different sizes. In this paper, the models will be ranked according to their Local Outlier Factor (LOF) values and the models with the top k (where k is a multiple of 5 ranging from 5 to 40) values will be selected to form the pruned ensemble.

2.1. Diversity Creation Methods

Because of the vital role diversity plays on the performance of ensembles, it had received a lot of attention from the research community. G. Brown et al. [13] summarized the work done to date in this domain from two main perspectives. The first is a review of the various attempts that were made to provide a formal foundation of diversity. The second, which is more relevant to this paper, is a survey of the various techniques to produce diverse ensembles. For the latter, two types of diversity methods were identified: implicit and explicit. While implicit methods tend to use randomness to generate diverse trajectories in the hypothesis space, explicit methods, on the other hand, choose different paths in the space deterministically. In light of these definitions, bagging and boosting in the previous section are classified as implicit and explicit respectively.

G. Brown et al. [13] also categorized ensemble diversity techniques into three categories: starting point in hypothesis space, set of accessible hypotheses, and manipulation of training data. Methods in the first category use different starting points in the hypothesis space, therefore, influencing the convergence place within the space. Because of their poor performance of achieving diversity, such methods are used by many authors as a default benchmark for their own methods [5]. Methods in the second category vary the set of hypotheses that are available and accessible by the ensemble. For different ensembles, these methods vary either the training data used or the architecture employed. In the third category, the methods alter the way space is traversed. Occupying any point in the search space, gives a particular hypothesis. The type of the ensemble obtained will be determined by how the space of the possible hypotheses is traversed.

In this paper, we propose a new diversity creation method based on unsupervised learning. The method utilizes an existing unsupervised learning

technique that, to the best of our knowledge, has not been used before in the production of pruned ensembles.

2.2. Diversity Measures

Regardless of the diversity creation technique used, diversity measures were developed to measure the diversity of a certain technique or perhaps to compare the diversity of two techniques. Tang et al. [15] presented a theoretical analysis on six existing diversity measures: disagreement measure [22], double fault measure [23], KW variance [24], inter-rater agreement [25], generalized diversity [26], and measure of difficulty [25]. The goal was not only to show the underlying relationships between them, but also to relate them to the concept of margin, which is one of the contributing factors to the success of ensemble learning algorithms.

We suffice to describe the first two measures as the others are outside the scope of this paper. The disagreement measure is used to measure the diversity between two base classifiers h_j and h_k, and is calculated as follows:

$$dis_{j,k} = \frac{N^{10} + N^{01}}{N^{11} + N^{10} + N^{01} + N^{00}}$$

where

- N^{10}: means number of training instances that were correctly classified by h_j, but are incorrectly classified by h_k
- N^{01}: means number of training instances that were incorrectly classified by h_j, but are correctly classified by h_k
- N^{11}: means number of training instances that were correctly classified by h_j and h_k
- N^{00}: means number of training instances that were incorrectly classified by h_j and h_k

The higher the disagreement measure, the more diverse the classifiers are. The double fault measure uses a slightly different approach where the diversity between two classifiers is calculated as:

$$DF_{j,k} = \frac{N^{00}}{N^{11} + N^{10} + N^{01} + N^{00}}$$

The above two diversity measures work only for binary classification (AKA binomial) where there are only two possible values (like Yes/No) for the class label, hence, the objects are classified into exactly two groups. They do not work for multiclass (AKA multinomial) classification where the objects are classified into more than two groups.

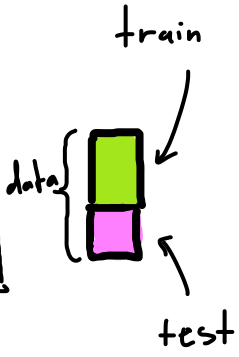
3. Preliminaries

3.1. Motivation

As mentioned before, RF algorithms tend to build between 100 and 500 trees [12]. Our research aims at producing child RFs that are significantly smaller in size and yet, have accuracy performance that is at least as good as that of the parent RF from which they were derived. The classification speed of each child is guaranteed to be much faster than that of the parent RF because 1) it has much fewer trees and 2) any tree used in the child is also in the parent (i.e., no new trees were introduced in the child).

3.2. Random Forest

RF is an ensemble learning method used for classification and regression. Developed by Breiman [8], the method combines Breiman's bagging sampling approach [7], and the random selection of features, introduced independently by Ho [27] [28] and Amit and Geman [29], in order to construct a collection of decision trees with controlled variation. Using bagging, each decision tree in the ensemble is constructed using a sample with replacement from the training data. Statistically, the sample is likely to have about 64% of instances appearing at least once in the sample. Instances in the sample are referred to as in-bag-instances, and the remaining instances (about 36%), are referred to as out-of-bag instances. Each tree in the ensemble acts as a base classifier to determine the class label of an unlabeled instance. This is done via majority voting where each classifier casts one vote for its predicted class label, then the class label with the most votes is used to classify the instance. Algorithm 1 below depicts the RF algorithm [8] where N is the number of training samples and S is the number of features in data set.



4. Local Outlier Factor

The Local Outlier Factor (LOF) algorithm was developed by Breunig et al. [30] to measure the outlierness of an object. The higher the LOF value

Algorithm 1 Random Forest Algorithm

{User Settings}
input N, S
{Process}
Create an empty vector $\overrightarrow{RF}$
for $i = 1 \rightarrow N$ **do**
  Create an empty tree T_i
  repeat
    Sample S out of all features F using Bootstrap sampling
    Create a vector of the S features $\overrightarrow{F_S}$
    Find Best Split Feature $B(\overrightarrow{F_S})$
    Create A New Node using $B(\overrightarrow{F_S})$ in T_i
  until No More Instances To Split On
  Add T_i to the $\overrightarrow{RF}$
end for
{Output}
A vector of trees $\overrightarrow{RF}$

assigned to an object, the more isolated the object is with respect to its neighbors. It is considered a very powerful anomaly detection technique in machine learning and classification. Earlier work on outlier detection was investigated in [31] [32] [33] [34], however, the work was limited by treating an outlier as a binary property to classify an object as an outlier or not, without assigning it a value to measure its outlierness as was done in [30].

The LOF can be used as a method to achieve diversity. It was one of 3 strategies used to obtain diversity when constructing an ensemble for the KDDCup 1999 dataset [35]. Schubert et al. [36] proposed methods for measuring similarity and diversity of methods for building advanced outlier detection ensembles using LOF variants and other algorithms.

Formally, Breunig et al. [30] introduced the concept of reachability distance in order to calculate the LOF. If the distance of object A to the k nearest neighbor is denoted by k-distance(A), where the k nearest neighbors is denoted by N_k(A), the following equation defines the reachability distance (rd):

$$rd_k(A, B) = max\{k{-}distance(B), d(A, B)\} \quad (1)$$

where $d(A, B)$ is the distance between objects A and B. The local reachability density of object A is then defined by

$$lrd(A) = \frac{\sum_{B \in N_k(A)} rd_k(A, B)}{|N_k(A)|} \tag{2}$$

Using the local reachability density of object A as defined in the previous equation, the LOF for object A is given by:

$$LOF_k(A) = \frac{\sum_{B \in N_k(A)} \frac{lrd(B)}{lrd(A)}}{|N_k(A)|} \tag{3}$$

5. LOF-Based Diverse Random Forest (LOFB-DRF)

In this section, we propose an extension of RF called LOFB-DRF that spawns a child RF that is 1) much smaller in size than the parent RF and 2) has an accuracy that is at least as good as that of the parent RF. In this extension, we use the LOF discussed in Section 4. As shown in Figure 1, each tree predictions on the training dataset (denoted by the vector $C(t_i, T)$) is assigned an LOF value that indicates the degree of its outlierness. The top k (k=5,10,...,40) trees corresponding to these predictions with the highest weighted LOF values (to be discussed next) are then selected to become members of the resulting LOFB-DRF. In the remainder of this paper, we will refer to the parent/original traditional Random Forest as *RF*, and refer to the resulting child RF based on our method as *LOFB-DRF*.

Based on Figure 1, we formalize the LOFB-DRF algorithm as shown in Algorithm 2 where T is the training set. The constant k refers to the number of trees that will have the highest weighted LOF values as will be discussed later. The domain of this constant is multiple of 5 in the range 5 to 40. This way and as we shall see in the experiments section, we can compare the performance RF with an LOFB-DRF of different sizes.

It is important to remember that the size of the resulting LOFB-DRF is determined by the constant k. For example, if k is 5, then the resulting LOFB-DRF will have size 5, and so on.

5.1. Selection of Trees

With reference to Algorithm 2, the selection of trees in RF that will become members of LOFB-DRF proceeds as follows. First, predictions of each

Algorithm 2 LOFB-DRF Algorithm

{User Settings}
input T, k
{Process}
Create an empty vector $\overrightarrow{treesPredictions}$
Create an empty vector $\overrightarrow{LOFB-RF}$
Using T, call Algorithm 1 above to create the parent RF
for $i = 1 \rightarrow RF.getNumTrees()$ **do**
$\overrightarrow{treesPredictions} \Leftarrow \overrightarrow{treesPredictions} \cup$ C(RF.tree(i), T)
end for
For each instance in $\overrightarrow{treesPredictions}$, assign an LOF value
Select the top k instances in $\overrightarrow{treesPredictions}$ with highest weighted LOF values
Select the corresponding trees from RF and add them to $\overrightarrow{LOFB-DRF}$
{Output}
A vector of trees $\overrightarrow{LOFB-DRF}$

tree on the training dataset T is computed as a vector and added to the vector $\overrightarrow{treesPredictions}$. At the conclusion of the **for** loop, $\overrightarrow{treesPredictions}$ becomes a super vector containing vectors where each vector stores the predictions of each tree. Each instance in $\overrightarrow{treesPredictions}$ is then assigned a normalized LOF value between 0 and 1. This way, each normalized value describes the probability of the instance being an outlier [35]. Then we assign to each instance a weight that is the product of the normalized LOF value and the accuracy rate of the corresponding tree on the training data. Formally, let c_i be an instance in the super vector $\overrightarrow{treesPredictions}$, NormalizedLOF($c_i$) be the normalized LOF value assigned to this instance, and AccuracyRate(Tree(c_i),T) be the accuracy rate of Tree(c_i) on the training dataset T where Tree(c_i) is the tree that corresponds to the instance c_i. The weight assigned to this instance is given by:

$$weight = NormalizedLOF(c_i) \times AccuracyRate(Tree(c_i), T) \quad (4)$$

The instances are then sorted in descending order by this weight and the corresponding top k trees are then selected.

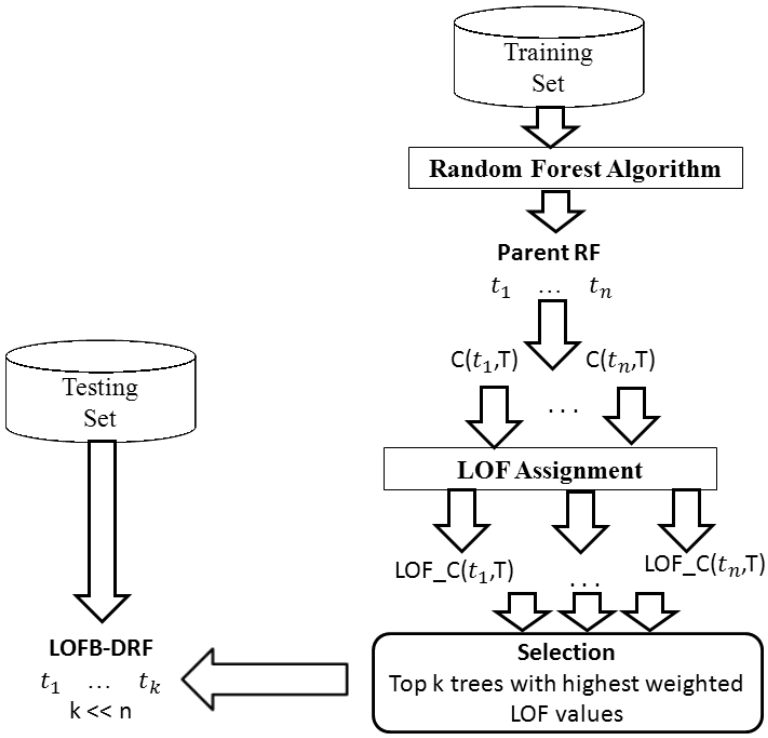


Figure 1: LOFB-DRF Approach

5.2. Diversity Measure

Here we propose a simple diversity measure to measure the diversity of classifiers that works with binary and multiclass classification. Given two classifiers h_j and h_k and a training set T of size n. Let $C(t_l,s_i)$ denotes the class label obtained after having t_l classify the sample s_i in the training set T. The diversity between the two classifiers can be measured by:

$$diversity_{j,k} = \frac{\sum_{i=1}^{n} \delta(C(t_j, c_i), C(t_k, c_i))}{n} \tag{5}$$

where

$$\delta(x_j, y_j) = \begin{cases} 0, & \text{if } x_j = y_j \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

The higher the number of discrepancies between the two classifiers, the higher the diversity is. For example, assume that we have a training set consisting of 10 training samples T={s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9,s_{10}}, and two classifiers t_1 and t_2. Assume also that there are 3 possible values for the class label {a,b,c}. Let C(t_1,T)=<a,a,b,c,c,a,b,c,b,b> and C(t_2,T)=<a,a,b,b,a,a,b,c,c,c>. According to 5 above, the diversity between the two classifiers is therefore 4/10 or 40%.

6. Experiments

For our experiments, we have used 10 real datasets with varying characteristics from the UCI repository [37]. To use the holdout testing method, each dataset was divided into 2 sets: training and testing. Two thirds (66%) were reserved for training and the rest (34%) for testing. Each dataset consists of input variables (features) and an output variable. The latter refers to the class label whose value will be predicted in each experiment. For the RF in Figure 1, the initial RF to produce the LOFB-DRF had a size of 500 trees, a typical upper limit setting for RF [12].

The LOFB-DRF algorithm described above was implemented using the Java programming language utilizing the API of Waikato Environment for Knowledge Analysis (WEKA) [38]. We ran this algorithm 10 times on each dataset where a new RF was created in each run. We then calculated the average of the 10 runs for each resulting LOFB-DRF to produce the average for a variety of metrics including accuracy rate, minimum accuracy rate, maximum accuracy rate, standard deviation, FMeasure, and AUC as shown in Table 5. For the RF, we just calculated the average accuracy rate, FMeasure, and AUC as shown in the last 3 columns of the table.

6.1. Results

Table 5 compares the performance of LOFB-DRF and RF on the 10 datasets used in the experiment. To show the superiority of LOFB-DRF, we have highlighted in boldface the average accuracy rate of LOFB-DRF when it is greater than that of RF. With the exception of the *audit* and *vote* datasets (last 2 datasets), we find that LOFB-DRF performed at least as good as RF. Interestingly enough, of the 10 datasets, LOFB-DRF, regardless of its size, completely outperformed RF on 3 of the datasets, namely, *squash-stored*, *eucalyptus*, and *sonar*. While LOFB-DRF lost to RF on only 2 datasets

(*audit* and *vote*), the difference was by a very small negligible fraction of less than 1% (in the case of *audit*), and less than 1.2% (in the case of *vote*)!

6.2. Pruning Level

In ensemble pruning, a pruning level refers to the reduction ratio between the original ensemble and the pruned one. For example, if the size of the original ensemble is 500 trees and the pruned one is of size 50, then $100\% - \frac{50}{500} \times 100\% = 90\%$ is the pruning level that was achieved in the pruned ensemble. This means that the pruned ensemble is 90% smaller than the original one. Table 1 shows the pruning levels where the first column shows the maximum possible pruning level for an LOFB-DRF that has outperformed RF, and the second column shows the pruning level of the best performer LOFB-DRF. We can see that with extremely healthy pruning levels ranging from 95% to 99%, our technique outperformed RF. This makes LOFB-DRF a natural choice for real-time applications, where fast classification is an important desideratum. In most cases, 100 times faster classification can be achieved with the 99% pruning level, as shown in the table. In the worst case scenario, only 16.67 times faster classification with 95% pruning level in the *squash-unstored* dataset. Such estimates are based on the fact that the number of trees traversed in the RF is the dominant factor in the classification response time. This is especially true, given that RF trees are unpruned bushy trees.

Note that the *audit* and *vote* datasets were not listed in the table as the RFs for these datasets (refer to the last 2 datasets in Table 5) outperformed all LOFB-DRFs, however, by a very small amount as shown in Table 2.

Table 1: Maximum Pruning Level with Best Possible Performance

Dataset	Maximum Pruning Level	Best Performance Pruning Level
breast-cancer	97%	95%
squash-unstored	95%	93%
squash-stored	99%	98%
eucalyptus	99%	99%
soybean	98%	97%
diabetes	96%	96%
car	99%	99%
sonar	99%	99%

6.3. Analysis

By showing the number of datasets each was superior on, Figure 2 compares the accuracy rate of RF and LOFB-DRF using different sizes of LOFB-DRF. For sizes 10, 15, 20, and 25, the figure clearly shows that LOFB-DRF indeed performed at least as good as RF. As shown in Table 2 below, for the cases (size 5, 30, 35, and 40) where RF outperformed LOFB-DRF, the difference was very small, considering the pruning level that was achieved.

Table 2: Outperformance Range of RF Over LOFB-DRF

LOFB-DRF Size	5	30	35	40
Range	0.31% - 4.12%	0.08% - 2.78%	0.05% - 1.45%	0.31% - 3.33%
Pruning Level	99%	94%	93%	92%

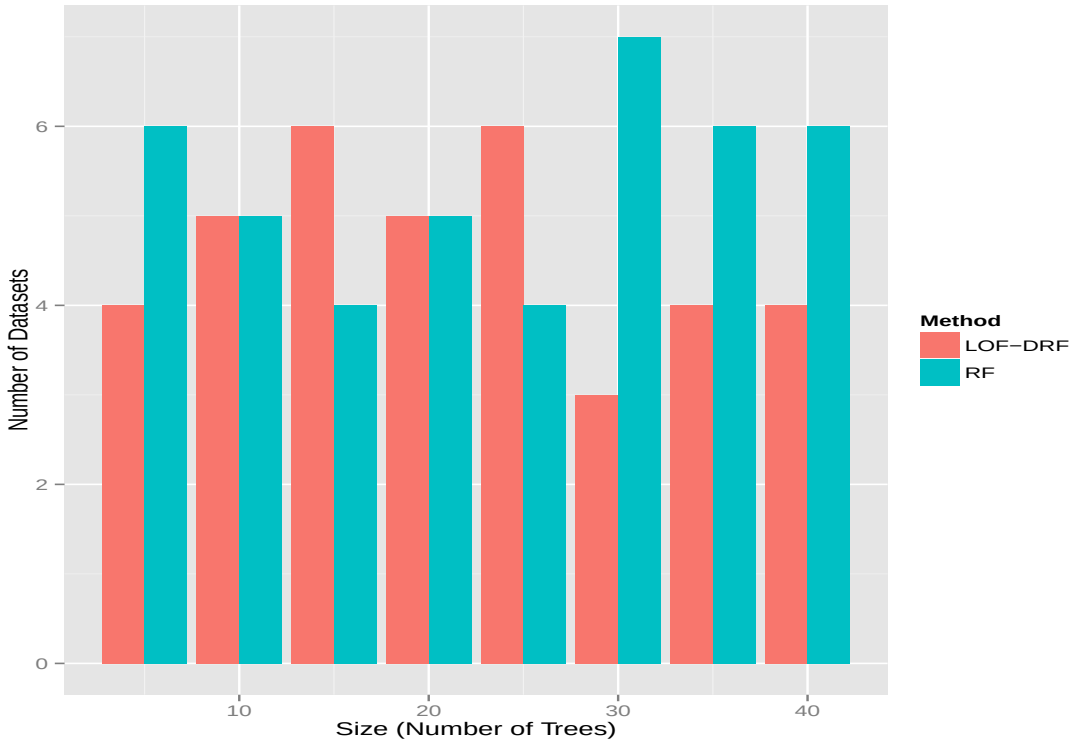


Figure 2: Accuracy Rate Comparison of RF & LOFB-DRF

6.4. Bias/Variance Analysis

Bias and variance are measures used to estimate the accuracy of a classifier [39]. The bias measures the difference between the classifier's predicted class value and the true value of the class label being predicted. The variance, on the other hand, measures the variability of the classifier's prediction as a result of sensitivity due to fluctuations in the training set. If the prediction is always the same regardless of the training set, it equals zero. However, as the prediction becomes more sensitive to the training set, the variance tends to increase. For a classifier to be accurate, it should maintain a low bias and variance.

There is a trade-off between a classifier's ability to minimize bias and variance. Understanding these two types of measures can help us diagnose classifier results and avoid the mistake of over- or under-fitting. Breiman et al. [40] provided an analysis of complexity and induction in terms of a trade-off between bias and variance. In this section, we will show that LOFB-DRF can have a bias and variance comparable to and even better than RF. Starting with bias, the first column in Table 3 shows the pruning level of LOFB-DRF that performed the best relative to RF, and the second column shows the pruning level of the smallest LOFB-DRF that outperformed RF. As demonstrated in the table, LOFB-DRF has outperformed RF on all datasets. On the other hand, Table 4 shows similar results but variance-wise. Once again, LOFB-DRF has outperformed RF on all datasets. Although looking at bias in isolation of variance (and vice versa) provides only half of the picture, our aim is to demonstrate that with a pruned ensemble, both bias and/or variance can be enhanced. We attribute this to the high diversity our ensemble exhibits.

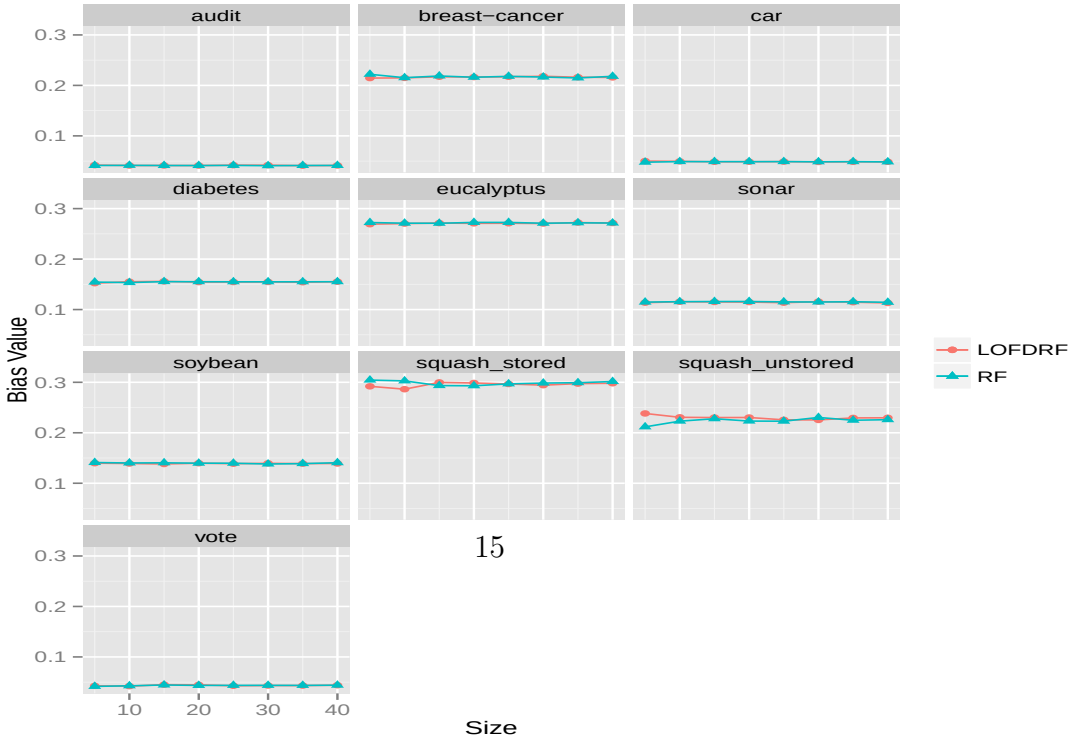
We have also conducted experiments to compare the bias and variance between LOFB-DRFs and Random Forests of identical size. Figure 3 compares the bias and Figure 4 compares the variance. Both figures show that LOFB-DRF in most cases can have bias and variance equal to or better than Random Forest.

Table 3: Pruning Level for LOFB-DRF Bias

Dataset	Best Performer	Smallest LOFB-DRF Outperforming RF
breast-cancer	99%	99%
squash-unstored	94%	95%
squash-stored	98%	99%
eucalyptus	99%	99%
soybean	97%	97%
diabetes	99%	99%
car	94%	97%
sonar	92%	99%
audit	93%	98%
vote	98%	99%

Table 4: Pruning Level for LOFB-DRF Variance

Dataset	Best Performer	Smallest LOFB-DRF Outperforming RF
breast-cancer	95%	99%
squash-unstored	99%	99%
squash-stored	97%	97%
eucalyptus	93%	98%
soybean	94%	94%
diabetes	98%	98%
car	99%	99%
sonar	99%	99%
audit	97%	97%
vote	92%	99%



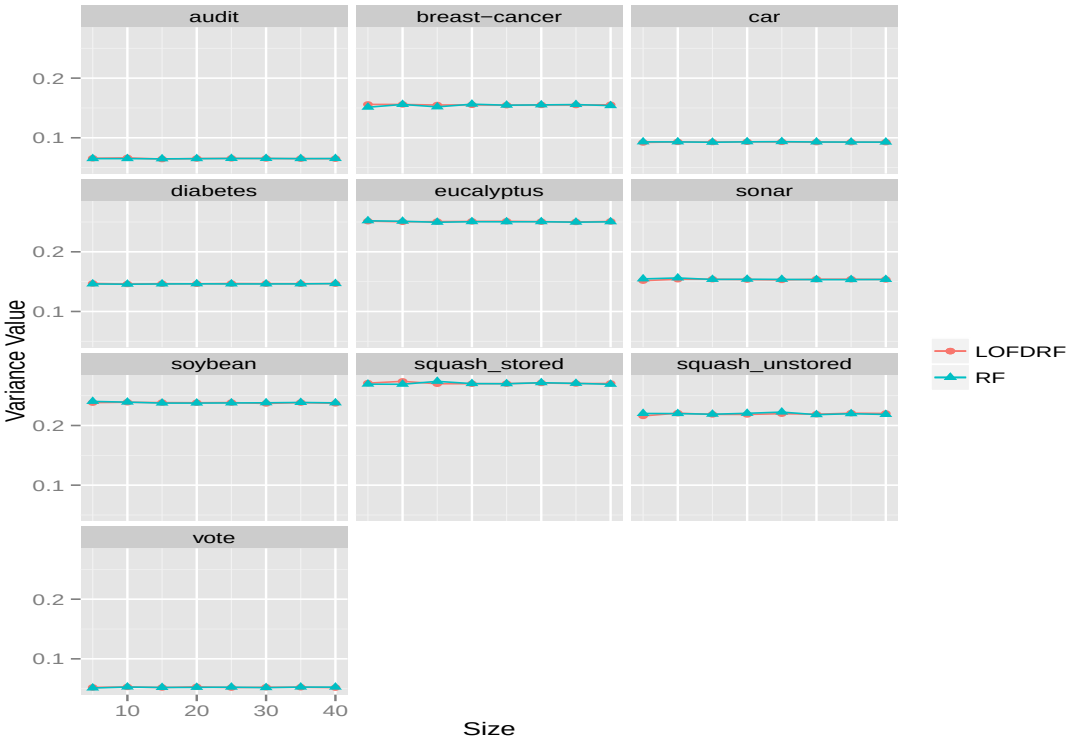


Figure 4: Variance Comparison of LOFB-DRF and Random Forest

7. Conclusion and Future Directions

Research conducted in this paper was based on how diversity in ensembles tends to yield better results [3] [13] [14] [15]. We adopted the Local Outlier Factor method to select diverse trees in an RF and then used these trees to form a pruned ensemble of the original ensemble. The selection was based on both LOF value and predictive accuracy of each tree. Experimental results have shown the potential of this method with extreme pruning of Random Forests that can outperform the original population of trees with values reaching 99% pruning level. This makes the pruned ensemble a suitable candidate for real-time applications.

We have selected trees that correspond to the instances with the top k weighted LOF values. Another interesting variation would be to use a hybrid approach that combines LOF with clustering to boost diversity up. Using this approach, we first create clusters of trees then from each cluster, we select

a representative that corresponds to the instance with the highest weighted LOF value. The current implementation also gives equal importance to the peculiarity of the tree as measured by the LOF score and the predictive accuracy, represented by the percentage of correctly classified instances for the tree. However, tuning this significance can play an important role in enhancing the classifier. From one hand, choosing trees with higher predictive accuracy can lead to model overfitting, and on the other hand, using LOF only can lead to leaving out trees that are most representative of the dataset. Balancing between the two can result in an ensemble that is diverse enough to boost the accuracy.

Table 5: Performance Metrics of LOFB-DRF & RF

LOFB-DRF Size	AVG	MIN	MAX	SD	Fmeasure	AUC	AVG FMeasure		AUC
breast-cancer									
5	67.01	61.86	74.23	3.16	0.65	0.57	71.13	0.65	0.58
10	67.22	64.95	69.07	1.71	0.66	0.58			
15	**71.34**	67.01	76.29	3.12	0.65	0.58			
20	69.48	67.01	73.20	2.62	0.66	0.58			
25	**71.86**	69.07	74.23	1.46	0.65	0.58			
30	70.41	68.04	72.16	1.53	0.65	0.58			
35	70.62	65.98	73.20	1.91	0.65	0.58			
40	69.18	64.95	72.16	2.14	0.65	0.58			
squash-unstored									
5	58.89	44.44	83.33	12.47	0.58	0.66	61.11	0.52	0.64
10	54.44	33.33	66.67	9.56	0.56	0.66			
15	60.56	50.00	83.33	8.77	0.55	0.65			
20	60.00	50.00	66.67	5.98	0.54	0.66			
25	**63.33**	55.56	77.78	7.93	0.54	0.65			
30	58.33	44.44	77.78	8.70	0.53	0.65			
35	**67.22**	50.00	83.33	10.08	0.54	0.66			
40	57.78	50.00	66.67	6.19	0.53	0.65			
squash-stored									
5	**56.67**	38.89	66.67	9.56	0.57	0.59	55.56	0.51	0.56
10	**59.44**	44.44	66.67	7.05	0.54	0.58			
15	**58.33**	50.00	66.67	4.48	0.54	0.58			
20	**58.33**	50.00	61.11	3.73	0.55	0.58			
25	**58.33**	50.00	66.67	5.12	0.53	0.57			
30	**56.67**	55.56	61.11	2.22	0.52	0.56			
35	**56.11**	55.56	61.11	1.67	0.52	0.57			
40	**56.11**	55.56	61.11	1.67	0.52	0.56			
eucalyptus									
5	**25.80**	11.20	40.40	8.73	0.26	0.60	19.92	0.21	0.57
10	**21.00**	12.40	28.40	4.70	0.24	0.59			
15	**24.32**	14.80	32.00	5.01	0.24	0.58			
20	**24.48**	15.60	29.60	4.55	0.23	0.58			
25	**24.68**	21.20	29.60	2.35	0.23	0.58			
30	**24.80**	14.80	33.60	5.13	0.23	0.58			
35	**23.96**	20.00	34.40	4.20	0.23	0.58			
40	**21.16**	15.20	28.00	3.69	0.22	0.57			
soybean									
5	77.28	60.78	85.78	6.80	0.79	0.88	77.59	0.73	0.85
10	**78.45**	70.69	85.34	5.46	0.75	0.87			
15	**79.57**	72.84	83.62	3.50	0.76	0.87			
20	76.85	74.57	78.88	1.26	0.74	0.86			
25	76.90	74.14	79.31	1.88	0.74	0.86			
30	76.85	72.41	81.47	2.43	0.74	0.86			
35	77.33	71.98	82.33	3.66	0.73	0.86			
40	76.59	71.98	81.03	2.59	0.73	0.85			
diabetes									
5	80.80	74.71	84.29	3.53	0.72	0.68	81.26	0.71	0.67

Continued on next page

Table 5 – continued from previous page

LOFB-DRF Size	AVG	MIN	MAX	SD	Fmeasure	AUC	AVG FMeasure	AUC
10	81.15	74.71	84.29	3.56	0.71	0.68		
15	79.85	77.39	83.14	1.96	0.71	0.67		
20	**81.42**	79.31	83.14	1.24	0.71	0.67		
25	80.96	78.93	82.76	1.31	0.71	0.67		
30	80.88	78.54	82.76	1.14	0.71	0.67		
35	79.81	77.39	81.99	1.40	0.71	0.67		
40	**81.38**	80.08	83.14	0.94	0.71	0.67		
car								
5	**64.17**	62.41	67.52	1.33	0.56	0.78	62.26 0.56	0.78
10	**63.01**	61.56	64.29	0.75	0.56	0.78		
15	**62.36**	60.71	64.29	1.12	0.56	0.78		
20	**62.35**	61.22	63.78	0.82	0.56	0.78		
25	**62.69**	60.88	63.95	0.85	0.56	0.78		
30	62.18	61.05	63.10	0.82	0.56	0.78		
35	61.96	60.88	63.61	0.72	0.56	0.78		
40	61.99	61.05	62.59	0.54	0.55	0.78		
sonar								
5	**12.25**	7.04	18.31	3.34	0.26	0.00	0.14 0.29	0.00
10	**9.15**	0.00	16.90	5.20	0.28	0.00		
15	**6.34**	0.00	14.08	4.47	0.29	0.00		
20	**3.38**	0.00	8.45	2.76	0.29	0.00		
25	**3.10**	0.00	7.04	2.42	0.28	0.00		
30	**1.83**	0.00	4.23	1.27	0.28	0.00		
35	**3.38**	0.00	4.23	1.29	0.28	0.00		
40	**3.38**	0.00	9.86	2.69	0.28	0.00		
audit								
5	95.63	94.26	96.47	0.72	0.91	0.89	96.31 0.90	0.88
10	95.74	95.00	96.18	0.35	0.90	0.88		
15	95.99	95.29	96.47	0.35	0.90	0.88		
20	96.06	95.29	96.76	0.39	0.90	0.88		
25	96.22	95.88	96.47	0.25	0.91	0.89		
30	96.03	95.59	96.47	0.25	0.90	0.88		
35	96.26	95.88	96.47	0.18	0.90	0.88		
40	96.00	95.59	96.47	0.27	0.90	0.87		
vote								
5	96.82	95.27	97.97	0.80	0.96	0.98	97.97 0.95	0.97
10	97.09	95.27	97.97	0.86	0.96	0.97		
15	97.57	96.62	97.97	0.45	0.95	0.97		
20	97.43	96.62	97.97	0.51	0.95	0.97		
25	97.57	96.62	97.97	0.45	0.95	0.97		
30	97.70	97.30	97.97	0.33	0.95	0.97		
35	97.64	96.62	97.97	0.45	0.95	0.97		
40	97.64	96.62	97.97	0.45	0.95	0.97		

References

[1] R. Polikar, Ensemble based systems in decision making, Circuits and Systems Magazine, IEEE 6 (2006) 21–45.

[2] L. Rokach, Ensemble-based classifiers, Artificial Intelligence Review 33 (2010) 1–39.

[3] L. I. Kuncheva, C. J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine learning 51 (2003) 181–207.

[4] W. Yan, K. F. Goebel, Designing classifier ensembles with constrained performance requirements, in: Defense and Security, International Society for Optics and Photonics, pp. 59–68.

[5] R. Maclin, D. Opitz, Popular ensemble methods: An empirical study, Journal Of Artificial Intelligence Research 11 (1999) 169–198.

[6] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of computer and system sciences 55 (1997) 119–139.

[7] L. Breiman, Bagging predictors, Machine learning 24 (1996) 123–140.

[8] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[9] D. H. Wolpert, Stacked generalization, Neural networks 5 (1992) 241–259.

[10] L. Breiman, Stacked regressions, Machine learning 24 (1996) 49–64.

[11] P. Smyth, D. Wolpert, Linearly combining density estimators via stacking, Machine Learning 36 (1999) 59–83.

[12] G. Williams, Use R: Data Mining with Rattle and R: the Art of Excavating Data for Knowledge Discovery, Springer, 2011.

[13] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: a survey and categorisation, Information Fusion 6 (2005) 5–20.

[14] J. J. G. Adeva, U. Beresi, R. Calvo, Accuracy and diversity in ensembles of text categorisers, CLEI Electronic Journal 9 (2005).

[15] E. K. Tang, P. N. Suganthan, X. Yao, An analysis of diversity measures, Machine Learning 65 (2006) 247–271.

[16] G. Tsoumakas, I. Partalas, I. Vlahavas, An ensemble pruning primer, in: Applications of supervised and unsupervised ensemble methods, Springer, 2009, pp. 1–13.

[17] D. Partridge, W. B. Yates, Engineering multiversion neural-net systems, Neural Computation 8 (1996) 869–893.

[18] Y. Yang, K. Korb, K. M. Ting, G. I. Webb, Ensemble selection for superparent-one-dependence estimators, in: AI 2005: Advances in Artificial Intelligence, Springer, 2005, pp. 102–112.

[19] D. D. Margineantu, T. G. Dietterich, Pruning adaptive boosting, in: ICML, volume 97, Citeseer, pp. 211–218.

[20] G. Martinez-Muoz, D. Hernández-Lobato, A. Suárez, An analysis of ensemble pruning techniques based on ordered aggregation, Pattern Analysis and Machine Intelligence, IEEE Transactions on 31 (2009) 245–259.

[21] G. Martínez-Muñoz, A. Suárez, Pruning in ordered bagging ensembles, in: Proceedings of the 23rd international conference on Machine learning, ACM, pp. 609–616.

[22] D. B. Skalak, The sources of increased accuracy for two proposed boosting algorithms, in: Proc. American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop, volume 1129, Citeseer, p. 1133.

[23] G. Giacinto, F. Roli, Design of effective neural network ensembles for image classification purposes, Image and Vision Computing 19 (2001) 699–707.

[24] R. Kohavi, D. H. Wolpert, et al., Bias plus variance decomposition for zero-one loss functions, in: ICML, pp. 275–283.

[25] J. L. Fleiss, B. Levin, M. C. Paik, Statistical methods for rates and proportions, John Wiley & Sons, 2013.

[26] D. Partridge, W. Krzanowski, Software diversity: practical statistics for its measurement and exploitation, Information and software technology 39 (1997) 707–717.

[27] T. K. Ho, Random decision forests, in: Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on, volume 1, IEEE, pp. 278–282.

[28] T. K. Ho, The random subspace method for constructing decision forests, Pattern Analysis and Machine Intelligence, IEEE Transactions on 20 (1998) 832–844.

[29] Y. Amit, D. Geman, Shape quantization and recognition with randomized trees, Neural computation 9 (1997) 1545–1588.

[30] M. M. Breunig, H.-P. Kriegel, R. T. Ng, J. Sander, Lof: identifying density-based local outliers, in: ACM Sigmod Record, volume 29, ACM, pp. 93–104.

[31] A. Arning, R. Agrawal, P. Raghavan, A linear method for deviation detection in large databases., in: KDD, pp. 164–169.

[32] I. Ruts, P. J. Rousseeuw, Computing depth contours of bivariate point clouds, Computational Statistics & Data Analysis 23 (1996) 153–168.

[33] E. M. Knox, R. T. Ng, Algorithms for mining distance-based outliers in large datasets, in: Proceedings of the International Conference on Very Large Data Bases, Citeseer.

[34] E. M. Knorr, R. T. Ng, Finding intensional knowledge of distance-based outliers, in: VLDB, volume 99, pp. 211–222.

[35] H.-P. K. P. K. Erich, S. A. Zimek, Interpreting and unifying outlier scores, in: 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ.

[36] E. Schubert, R. Wojdanowski, A. Zimek, H.-P. Kriegel, On evaluation of outlier rankings and outlier scores, in: Proceedings of the 12th SIAM International Conference on Data Mining (SDM), Anaheim, CA, 2012, pp. 1047–1058.

[37] K. Bache, M. Lichman, Uci machine learning repository, 2013.

[38] G. H. B. P. P. R. I. H. W. Mark Hall, Eibe Frank, The WEKA Data Mining Software: An Update, volume 11, SIGKDD Explorations, 2009.

[39] R. Kohavi, et al., A study of cross-validation and bootstrap for accuracy estimation and model selection, in: IJCAI, volume 14, pp. 1137–1145.

[40] B. Leo, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and regression trees, Wadsworth International Group (1984).