An Empirical Study of Learning from Imbalanced Data Using Random Forest

Taghi M. Khoshgoftaar (taghi@cse.fau.edu)
Moiz Golawala (moizgolawala@hotmail.com)
Jason Van Hulse (jvanhulse@gmail.com)
Department of Computer Science and Engineering
Florida Atlantic University, Boca Raton, FL, USA

Abstract

This paper discusses a comprehensive suite of experiments that analyze the performance of the random forest (RF) learner implemented in Weka. RF is a relatively new learner, and to the best of our knowledge, only preliminary experimentation on the construction of random forest classifiers in the context of imbalanced data has been reported in previous work. Therefore, the contribution of this study is to provide an extensive empirical evaluation of RF learners built from imbalanced data. What should be the recommended default number of trees in the ensemble? What should the recommended value be for the number of attributes? How does the RF learner perform on imbalanced data when compared with other commonly-used learners? We address these and other related issues in this work.

1 Introduction

Weka [22] is an open source data mining and machine learning package implemented in JAVA at the University of Waikato. Many researchers and practitioners in the data mining and machine learning community commonly use Weka, and past work has provided no empirically proven recommendation on the appropriate default values for the ensemble size parameter $numTrees$ (number of trees in the ensemble) or the parameter controlling the number of attributes $numFeatures$ (number of attributes randomly selected to be considered at each node when building the trees) for the random forest classifier. We contend that default values of model parameters should be reasonable for experimentation. This work, as far as we know, is the first to conduct comprehensive experimentation with the RF learner in Weka and recommend empirically proven default values for the $numTrees$ and $numFeatures$ parameters.

Learning from imbalanced data is one of the most crucial problems encountered in data mining and knowledge discovery applications. In many domains, the inequality between the number of examples in each of the classes is severe. The degree of class imbalance can vary dramatically in real-world datasets, and in some application domains, such as fraud detection and diagnosis of rare diseases, the class of interest (the minority class) is often rare, less than 10% of the total cases. This is especially problamatic because the minority class is often the class of interest and has much higher misclassification costs than the majority class.

The random forest learner (Breiman, 2001) [6], as implemented in Weka, is an ensemble of unpruned classification trees that use majority voting to perform prediction. The RF learner gives accuracy that compares favorably with Adaboost [10] and is relatively robust against noise. The learner utilizes bagging [5] as well as the 'random subspace method' [12] in order to construct randomized decision trees. The outputs of ensembles of these randomized, unpruned decision trees are combined to produce the ultimate prediction. The RF learner has a number of interesting and useful properties, for example [6]:

- It has accuracy similar or better than Adaboost,
- It is relatively robust to outliers and noise,
- It is faster than bagging or boosting,
- As more trees are added to the forest, the tendency to overfit decreases.

Since the introduction of random forest in 2001 [6], there has been little empirical work on the topic [7, 14, 4, 24, 9, 13]. As far as we know, there is no related work that empirically recommend default settings for the $numTrees$ and $numFeatures$ parameters for the RF learner in the Weka tool. In [24], $numFeatures$ is varied to optimize the out-of-bag error rate, but no attention has been paid to varying the $numTrees$ parameter. Therefore, we present, in this work, a comprehensive empirical evaluation of learning from imbalanced data by varying the $numTrees$ and $numFeatures$ parameters. Our experimentation on the RF algorithm is done in Weka using 10 different datasets with varying degrees

1082-3409/07 $25.00 © 2007 IEEE
DOI 10.1109/ICTAI.2007.46

of class imbalance. The values tested for the $numTrees$ parameter are 10 (Weka's default), $50, 100, 150, 200$, and 500, and the values tested for the $numFeatures$ parameter are $1, 2, 3, M/10, M/5, M/3, M/2$, and $\lfloor \log_2 M + 1 \rfloor$(Weka's default), where 'M' is the total number of attributes. Statistical analysis using Analysis of Variance (ANOVA) models are used to determine reasonable default values for the $numTrees$ and $numFeatures$ parameters for all 10 datasets. In addition, we conduct experimentation to benchmark the RF learner (with the recommended $numTrees$ and $numFeatures$ values) with six different learners commonly-used in data mining and machine learning research. By varying the parameters $numTrees$ and $numFeatures$, this work is the first to compare the RF learner's performance on imbalanced data and recommend a default setting for each of the parameters.

The conclusions of our experimentation show that the default setting in Weka for $numTrees$ (10) is not an appropriate value, but the default value for the $numFeatures$ ($\lfloor \log_2 M + 1 \rfloor$) is reasonable. This work further shows the robustness of the RF learner implementation in Weka when compared to six other learners. Thorough experimentation makes our work extremely comprehensive and dramatically augments the reliability of our conclusions.

The remainder of this work is organized as follows. Section 2 provides background on RF usage in different domains. Section 3 presents a description of the imbalanced datasets used in our experiments. We provide an explanation of the different learners in Section 4. Section 5 discusses the metrics used to evaluate learner performance. Section 6 describes the experimental design, and a discussion of the results is given in Section 7. Finally, Section 8 presents conclusions and future work.

2 Related Work

Breiman published his work on the RF learner in 2001 [6]. Since its introduction, the RF learner has been used in diverse domains such as fraud detection in online retailing [9], image data classification [4], and network intrusion detection [24] because of its robustness to noise and the strength of its predictions. Breiman's original paper on RF presents the algorithm and compares it to AdaBoost and Bagging, and discusses in detail how the strength and correlation of each tree has an impact on the overall forest. He discusses the $numFeatures$ for each tree using the CART algorithm and uses 100 trees in his experiments to show the strength of the forest. Breiman's work, however, does not experiment with many different values for the $numTrees$ parameter, and empirical evidence is not provided to substantiate the choice for $numTrees$ of 100. For the $numFeatures$ parameter, he recommends selecting $\lfloor \log_2 M + 1 \rfloor$ attributes in building each tree in the ensemble. However, only one other value is utilized in experimentation ($numFeatures = 1$) and there is no justification for using $numFeatures = \lfloor \log_2 M + 1 \rfloor$.

Zhang and Zulkernine [24] use the default values in Weka to build the ensembles in the context of network intrusion detection. They do vary the $numFeatures$ parameter and determine the optimal number of features (15 attributes out of a total of 38) to be selected in building the trees. Our experimentation shows that using a default value of 10 in Weka for the $numTrees$ parameter is not a reasonable choice, and that a larger ensemble is more appropriate. Most other studies [4, 7, 9, 13, 14, 24] either do not mention the $numTrees$ and $numFeatures$ parameter settings, or in most cases they use a value for $numTrees$ that is far too low.

RF has been discussed as a robust learner in several domains. However for imbalanced data, a comprehensive evaluation of the RF learner, with varying values for $numTrees$ and $numFeatures$, has not been performed. Researchers and practitioners who use Weka have no guidance on these settings and too often rely on the default values (10 and $\lfloor \log_2 M + 1 \rfloor$) in the Weka tool. Our work shows that the default value for $numTrees$ in Weka is not appropriate for experimentation with imbalanced data.

3 Experimental Datasets

The 10 imbalanced datasets utilized in our empirical study are listed in Table 1. The percentage of minority examples varies from 1.33% (highly imbalanced) to 30% (only slightly imbalanced). The imbalanced datasets come from a wide variety of application domains, including the UCI repository [3] and from the domain of software engineering measurements. We have also considered datasets with diversity in the number of attributes and datasets with both continuous and categorical attributes. The largest number of attributes is 61 (SpliceJunction-2) while the smallest is 19 (Vehicle-1). The smallest dataset has 846 examples (Vehicle-1), while the largest dataset (Satimage-4) contains over 6400 observations. Note that all imbalanced datasets have, or were transformed to have, a binary class. We only consider binary classification problems in this work.

4 Learners

4.1 RF Learner

Random forest classifiers induce an ensemble of unpruned classification trees. Randomness is accomplished first by selecting a bootstrap (sampling with replacement) sample from the training dataset to build the classifier. RF could be considered a special case of bagging, another ensemble learning algorithm that combines prediction from

Table 1. Empirical Datasets

Name	# total	# minority	% minority	# attr.
German Credit	1000	300	30.00%	21
Optdigits-8	5619	554	9.86%	65
Satimage-4	6434	626	9.73%	37
Satimage-5	2309	330	14.29%	20
SP1	3646	229	6.28%	43
SP2	3979	189	4.75%	43
SP3	3534	47	1.33%	43
SP4	3983	92	2.31%	43
SpliceJunc-2	3189	768	24.08%	61
Vehicle-1	846	212	25.06%	19

several unstable classifiers by using bootstrapping. The RF learner combines the predictions from classification trees induced using an algorithm similar to C4.5 (J48 in Weka). Bootstrapping is a process of random sampling with replacement from the training dataset. By using bootstrapping in the tree induction process, approximately $1/3$ of the instances in the training dataset are not used. These are referred to as out-of-bag (OOB) samples. Instances used in building the classifier are called in-the-bag instances. Another source of randomness is introduced by randomly selecting a subset of the attributes to consider at each node of each decision tree. The number of classifiers in the ensemble is determined by the $numTrees$ parameter. For each tree in the ensemble, the $numFeatures$ parameter determines the number of attributes (features) that will randomly be selected to determine the splitting for each node in the tree. Below is the algorithm for RF:

1. Build bootstrapped sample $\mathcal{B}_i$ from the original dataset $\mathcal{D}$, where $\mid \mathcal{B}_i \mid = \mid \mathcal{D} \mid$ and examples are chosen at random with replacement from $\mathcal{D}$.
2. Construct a tree $\mathcal{T}_i$ using $\mathcal{B}_i$ as the training dataset using the standard decision tree algorithm with the following modifications:
   (a) At each node in the tree, restrict the set of candidate attributes to a randomly selected subset $(x_1, x_2, \ldots, x_k)$, where $k = numFeatures$.
   (b) Do not prune the decision tree.
3. Repeat Steps 1 and 2 for $i = 1, \ldots, numTrees$, creating a forest of trees $\mathcal{T}_i$, derived from different bootstrap samples.
4. When classifying an example x, aggregate the decisions (votes) over all trees $\mathcal{T}_i$ in the forest. If $\mathcal{T}_i(x)$ is the class of x as determined by tree $\mathcal{T}_i$, then the predicted class of x is the class that occurs most often in the ensemble, i.e., the class with the majority of votes.

By default, Weka uses $numFeatures = \lfloor \log_2 M + 1 \rfloor$, where M is the number of attributes in the dataset (this value was suggested by Breimen in his original work). In our experimentation we vary the $numFeatures$ from $1, 2, 3, M/10, M/5, M/3, M/2$, and $\lfloor \log_2 M + 1 \rfloor$ for all imbalanced datasets, and our experimental results show that $\lfloor \log_2 M + 1 \rfloor$ is an appropriate default value. Our experimentation also considers different values for $numTrees$, specifically $10, 50, 100, 150, 200$, and 500 (the default value in Weka is 10). Our experiments demonstrate that 10 is not an appropriate value, and in particular 100 is more reasonable.

The classification error rate of the RF learner depends on the strength and the correlation between individual tree classifiers in the ensemble. RF relies on randomness in each tree classifier for its robustness. If all trees in the ensemble were identical, an ensemble would perform exactly as any single tree within the ensemble. Therefore bootstrapping and randomly selecting attributes determined by $numFeatures$, as mentioned earlier are crucial in lowering the correlation between individual trees in the RF learner. The strength of the RF learner also depends on the strength of each individual tree. A classification tree with robust performance increases the strength of the forest. Having a diverse collection of robust trees lowers the overall error rate and enhances the overall strength of the RF learner.

4.2 Naive Bayes

Naive Bayes [15] is a simple classifier that utilizes Bayes' rule of conditional probability. Naive Bayes (NB) classifiers are very quick to learn and apply. The classifier is termed 'naive' because it assumes that all predictor variables are conditionally independent. Though the independence assumption is typically inaccurate for real-world data, the classifier has been shown to often perform well even in the presence of strong attribute dependencies [8]. The parameters for Naive Bayes were left at the default values in our experiments.

4.3 Support Vector Machines

A support vector machine [20] is a linear classifier which attempts to learn the maximum margin hyperplane separating two classes in the data. This maximum margin hyperplane is usually determined by a small subset of instances called support vectors. In order to learn nonlinear decision boundaries, the data can be transformed by a kernel function. The linear maximum margin hyperplane then found for the transformed data can represent a nonlinear partitioning of the original feature space. SMO is the implementation of support vector machines in Weka. Two changes to the default parameters were made: the complexity constant 'c' was changed from 1.0 to 5.0, and the '$buildLogisticModels$' parameter, which allows proper

probability estimates to be obtained, was set to '$true$' [22]. By default the linear kernel is used.

4.4 K Nearest Neighbors

K nearest neighbors [1], abbreviated kNN, is called a 'lazy learning' technique because little effort goes into building the classifier and most of the work is performed at the time of classification. All examples in the training dataset are stored in a case library and a prediction for an instance is calculated based on the classes of k nearest neighbors of the instance found in the case library. The nearest neighbors are the instances in the case library that are the closest in the feature-space to the instance being classified based on some distance measure, usually Euclidean distance.

For the kNN classifier (denoted Ibk in Weka), we set the kNN parameter at 2 (referred to as $2NN$) and at 5 (referred to as $5NN$), thereby creating two separate classifiers. One parameter setting was modified from its default value; the $distanceWeighting$ parameter was set to 'Weight by $1/distance$' to use inverse distance weighting in determining how to classify an instance.

4.5 C4.5

C4.5 is a benchmark decision tree algorithm proposed by Quinlan [18]. This learner is quite robust and has become the de facto standard against which other learning algorithms are compared. The decision tree is built using an entropy-based splitting criterion stemming from information theory. C4.5 improves Quinlan's older ID3 [17] decision tree algorithm by adding support for tree pruning and dealing with missing values and numeric attributes. The Weka version of C4.5 is called J48. Two different versions of the C4.5 classifiers were used in this study, denoted C4.5D and C4.5N. C4.5D uses the default parameter settings in Weka, while C4.5N uses no pruning and Laplace smoothing [21].

5 Metrics

Two measures of classifier performance are presented in this work, the *area under* the ROC *curve* (AUC) and the Kolmogorov-Smirnov (KS) [11] statistic. KS measures the maximum difference between the cumulative distribution functions of the predicted probabilities of examples in each class. The class c_i distribution function $F_{c_i}(t)$, is estimated by the proportion of class c_i examples with posterior probabilities $\leq t, 0 \leq t \leq 1$:

$$F_{c_i}(t) = \frac{\#\text{ class } c_i \text{ instances with posterior probability} \leq t}{\#\text{ class } c_i \text{ instances}}.$$

Then the KS statistic is defined as:

$$KS = \max_{t \in [0,1]} \mid F_{c_1}(t) - F_{c_2}(t) \mid . \quad (1)$$

The larger the distance between the two distribution functions, the better the learner is able to separate examples in the two classes. The maximum possible value for KS is one (perfect separation). It has a minimum of zero (no separation). In other words, a KS of zero implies that $F_{c_1}(t) = F_{c_2}(t) \; \forall \; t$, and the classifier is not able to differentiate minority and majority class examples.

Receiver Operating Characteristic curves, or *ROC curves*, graphs the true positive rate on the y-axis versus the false positive rate on the x-axis. The ROC curve illustrates the performance of a classifier across the entire range of decision thresholds, and accordingly does not assume any particular misclassification costs or class prior probabilities. For a single numeric measure, the AUC is often used, which gives a general idea of the predictive potential of the classifier. A higher AUC is better, as it indicates that, in general, the ROC curve obtains higher true positive and lower false negative rates. An in-depth explanation of ROC curves and their potential use for creating optimal classifiers is provided in Provost and Fawcett [16].

We chose to use these two metrics because they measure the general ability of the learner to separate the two classes and are independent of the choice of decision threshold. Other performance measures can also be used, and we leave a consideration of alternative measures to future work.

6 Experimental Design

The design of our experiments can be summarized in two phases. In the first phase, for each of the 10 imbalanced datasets, 20 different runs of five-fold cross validation (CV) were executed. For all iterations of CV, the training datasets consisted of four folds, and the remaining fold served as a test dataset. Each of the 48 RF learners (built by varying six settings for $numTrees$ and eight settings for $numFeatures$) were built using the training datasets (four folds), and evaluated with the test datasets (remaining one fold). The performance of the RF learners is compared, and appropriate values for $numTrees$ and $numFeatures$ were selected as discussed in Section 7.

In the second phase of experimentation, six learners (2NN, 5NN, C4.5N, C4.5D, SVM, and NB) were built and compared with the selected RF learner using the same 10 imbalanced datasets. Once again, we built each learner using the training datasets and evaluated it on the test datasets (based on CV).

In total 20 five-fold CV runs times 10 imbalanced datasets is $1,000$ different training datasets. In phase

one, we experimented with a combination of six ensemble sizes and eight different values for $numFeatures$ (total of $6 \times 8 = 48$) for RF. Therefore we built $48,000$ $(1,000 \times 48)$ RF learners. In phase two, we have an additional six learners built on each of the 20 five fold CV runs of 10 datasets resulting in $6,000$ learners. Therefore, a total of $54,000$ classifiers were constructed and evaluated in this study.

7 Experimental Results

7.1 Phase 1: Selecting an Appropriate RF Learner

In this study, for each RF learner and 10 imbalanced datasets, a two-factor analysis of variance (ANOVA) model [2] was constructed, where the first factor was the $numTrees$, and the second factor was the $numFeatures$. A two factor, full factorial ANOVA model can be written as:

$$AUC_{ijk} = \mu + \theta_i + \psi_j + (\theta\psi)_{ij} + \epsilon_{ijk} \quad (2)$$

where:

- μ is the overall mean AUC performance
- θ, ψ are the two main factors; θ is $numTrees$, and ψ is $numFeatures$
- θ_i, ψ_j are the treatment effects of ith and jth levels of the experimental factors θ, ψ. i is an index on $numTrees$, $i = 1, \ldots, 6$, while j is an index on the $numFeatures$, $j = 1, \ldots, 8$. For example, θ_1 corresponds to $numTrees = 10$, while θ_2 corresponds to $numTrees = 50$.
- $(\theta\psi)_{ij}$ are two-way interaction terms between the main effects
- AUC_{ijk} is the AUC of the k^{th} observation of the i^{th} level of θ and j^{th} level of ψ. Note that since there are 10 imbalanced datasets each with 20 different cross validation runs, $k = 1, \ldots, 1000$. In other words, $AUC_{ij1}, \ldots, AUC_{ij1000}$ are the $1,000$ AUC values obtained by the RF learner with parameters θ_i and ψ_j. For example, $i = 3$ and $j = 6$ would correspond to the RF learner with $numTrees = 100$ and $numFeatures = M/3$.
- ϵ_{ijk} is the random error.

An ANOVA model with the KS as the performance measure can be defined in an analogous manner. ANOVA models are useful because the variability in performance can be factored among the different experimental effects (in this case, the number of trees and number of features for the RF learner) allowing the effects which significantly contribute to changes in the performance to be determined.

Table 2. ANOVA Table

Effect(AUC)	DF	SS	MS	F	p-value
$numTrees(\theta)$	5	46.64	9.33	246.79	.0001
$numFeatures(\psi)$	7	1.11	0.16	4.19	.0001
$\theta \times \psi$	35	1.19	0.03	0.90	.6386
Effect(KS)	DF	SS	MS	F	p-value
$numTrees(\theta)$	5	97.88	19.58	264.87	.0001
$numFeatures(\psi)$	7	4.32	0.62	8.34	.0001
$\theta \times \psi$	35	3.32	0.09	1.28	.1209

Table 3. Analysis of Effects

Level	AUC	**HSD**	**Level**	**KS**	**HSD**
500	**0.9135**	A	500	**0.7286**	A
200	0.9109	AB	200	0.7218	AB
150	0.9096	AB	150	0.7180	AB
100	0.9071	B	100	0.7110	B
50	0.8999	C	50	0.6944	C
10	0.8561	D	10	0.6042	D
Level	AUC	**HSD**	**Level**	**KS**	**HSD**
$M/5$	**0.9027**	A	$M/3$	**0.7040**	A
$M/3$	0.9025	A	$M/5$	0.7039	A
$\lfloor \log_2 M + 1 \rfloor$	0.9023	A	$M/2$	0.7031	AB
$M/2$	0.9018	A	$\lfloor \log_2 M + 1 \rfloor$	0.7029	AB
$M/10$	0.9009	A	$M/10$	0.6991	AB
3	0.9002	AB	3	0.6962	B
2	0.8977	AB	2	0.6898	CB
1	0.8936	B	1	0.6774	C

Further, the ANOVA model can be used to test the hypothesis that the mean AUC or KS of the levels of a factor are equal against the alternative hypothesis that at least the mean of one level is different. If the alternative hypothesis (i.e., that at least one mean is different) is accepted, a number of procedures can be used to determine which of the means are significantly different from the others. This involves the comparison of two means with the null hypothesis that the means are equal. Tukey's Honestly Significant Difference (HSD) test [19] is a statistical test comparing the mean value of the performance measure (AUC or KS) for the different levels of each factor (either $numTrees$ of $numFeatures$). Two levels of a factor with the same block letter are not statistically different with 95% statistical confidence. All the statistical tests in this work use 95% confidence level.

The ANOVA table using both the AUC and KS is provided in Table 2. The first column lists the three experimental factors (two main factors and one interaction), followed by the degrees of freedom (DF), sum of squares (SS), mean square (MS), F-statistic, and the p-value of the F-statistic. The main effects individually are significant with p-values far less then 5%. However, the interaction between the main effects is not significant as the p-value is greater than 5%.

Table 3 presents an analysis of effects, i.e., a detailed examination of the two statistically significant main fac-

tors in the ANOVA model. The results are provided for both the AUC and KS metrics across six different values for $numTrees$ (factor θ) and eight different values for $numFeatures$ (factor ψ). The highest mean achieved by the RF learners for each of the measures is in bold. As expected, an RF learner with 500 trees (which we denote for simplicity as RF-500) has achieved the highest average AUC and KS measures across all the learners when comparing different $numTrees$. Considering the HSD grouping, however, RF-500 might not be the preferred default choice. Levels of each factor with means that are significantly different from one another share the same HSD label. Notice that the average AUC for RF-500 is not statistically different from RF-200 or RF-150 since they are labeled with the letter 'A' in the HSD column. In other words, since the HSD value for these three levels contains a letter 'A', the HSD test concludes that there is no statistically significant difference between the mean AUC of these three values. Likewise, since the HSD label for RF-50 ('C') is different than the HSD label of RF-10 ('D'), the average AUC and KS measures for RF-50 are statistically different from the RF-10 learner.

For both KS and AUC measures RF-100 is statistically superior to RF-50 and RF-10 and is not statistically different from RF-150 and RF-200 (RF-100 shares the same HSD label 'B' with RF-150 and RF-200). We consider default values to be reasonable values that provide adequate performance for a learner. The RF learner with $numTrees = 100$ qualifies as a reasonable default choice. Though slightly inferior to RF-500 (RF-500 an HSD label of 'A' while RF-100 has an HSD label of 'B'), RF-100 outperforms RF-10 and RF-50 and is comparable with RF-150 and RF-200. Therefore, we recommend $numTrees = 100$ for a default value for building RF learners from imbalanced data in the Weka tool. The same analysis holds true for the KS measure. The ANOVA analysis shows that RF with $numTrees$ of 100 would be the reasonable default value. The current default value in Weka of 10 trees is the lowest performer on both AUC and KS measures and thus is considered inadequate.

Considering $numFeatures$, the highest mean AUC is achieved when $numFeatures$ equals to $M/5$ (i.e., randomly select 20% of the features when splitting at each node) while setting the $numFeatures$ to $M/3$ (one-third of the features) produced the highest KS mean. The HSD test shows that there is no significant difference between $\lfloor \log_2 M + 1 \rfloor$ (the default value in Weka) and the setting which achieved the best performance ($M/5$) for AUC (both values have the HSD label as 'A'). Similar results hold for the KS measure, where $\lfloor \log_2 M + 1 \rfloor$ is not statistically different than the setting which resulted in the best performance ($M/3$). Therefore, our recommendation is that the default value of $\lfloor \log_2 M + 1 \rfloor$ for the $numFeatures$ parameter is a good choice in the Weka tool.

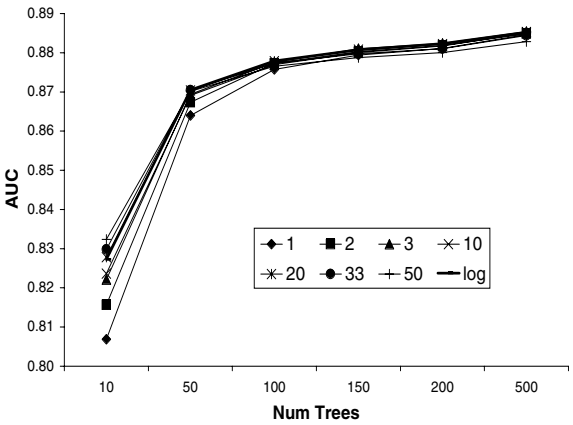

Figure 1. AUC Interaction Of Main Effects

Note that further optimization of the RF learner may be obtained by changing the $numTrees$ and $numFeatures$ settings that we have recommended. However, our intention in this study was not to optimize the learner but merely to analyze its performance under reasonable choices (default values).

Figure 1 displays the interaction between the two main factors, $numTrees$ and $numFeatures$, for the AUC measure. Qhen $numTrees = 10$, there is a wide dispersion of average AUC measures for the different $numFeatures$. However the values for the different $numFeatures$ converge as the $numTrees$ value approaches 50 trees. This concurs with the p-value in the ANOVA table, which indicates that the interaction between the main factors is not statistically significant. Indeed, the trend lines from Figure 1 are reasonably parallel to one another, indicating a very weak interaction between the two factors. The importance of statistical models like ANOVA is that the statistical significance of the interaction can be precisely determined.

Similar results were obtained for the KS measure, but due to space restrictions an analysis of the interaction for the KS measure is not included here. Therefore, we conclude that the default values of 100 for $numTrees$ and $\lfloor \log_2 M + 1 \rfloor$ for $numFeatures$ are reasonable. We continue with the second phase of our experimentation using the RF-100 learner with the default $numFeatures$ parameter value of $\lfloor \log_2 M + 1 \rfloor$ and compare it to six commonly-used learners.

7.2 Phase 2: Comparison of RF-100 to Other Learners

This phase of our study compares RF-100 with the default $numFeatures$ parameter value of $\lfloor \log_2 M + 1 \rfloor$ (labeled in Figure 2 and 3 as RF100Log) with six different learners: 2NN, 5NN, C4.5D, C4.5N, NB, and SVM. Figure 2 shows the box plot for the AUC measures across all

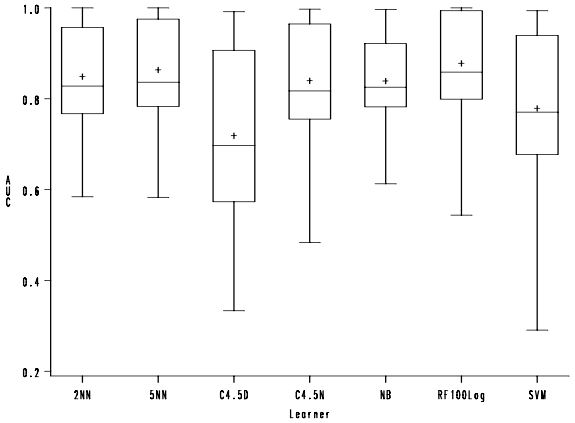

Figure 2. AUC Box Plot - All Learners

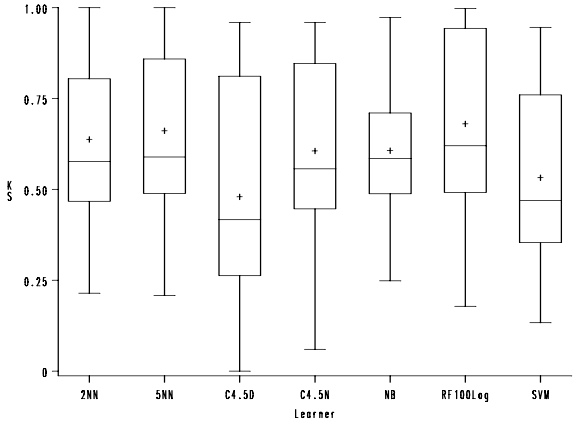

Figure 3. KS Box Plot - All Learners

the learners. From the AUC box plot we can see the C4.5D learner shows the highest variability in results (indicated by the larger box) with the lowest values for median and lower quartile. We also note that 2NN, 5NN, and C4.5N have an AUC performance that is almost identical, with C4.5N having a slightly worse lower quartile and median value. The smallest box, indicating the tightest range between upper quartile and lower quartile values, belongs to the NB learner. Overall 2NN, 5NN, NB, and C4.5N have similar median values. The RF learner has the highest mean, upper quartile, and lower quartile of all the learners. These findings indicate that the RF-100 learner with the default $numFeatures$ setting in Weka provides superior results on imbalanced data when compared with other learners.

Figure 3 compares the six learners with RF-100 and the default attribute selection value using the KS measure, with similar results as the AUC measure in Figure 2. NB shows the tightest range between the upper and lower extremes and upper and lower quartile values (the smallest box). C4.5D is clearly the worst performer with the lowest median, upper quartile, and lower quartile values in addition to the largest variability between the quartiles, indicating the largest variation in results. The medians for 2NN, 5NN, NB, and C4.5N are similar. The RF learner once again performs the best with the highest median, lower quartile, and upper quartile values. Similar to the AUC measure, the KS measure also indicates that the RF-100 learner with the default $numFeatures$ setting in the Weka tool provides superior results when compared with these six learners, and we highly recommend the use of RF when learning from imbalanced data.

7.3 Threats to Validity

Experimental research commonly includes a discussion of two different types of threat to validity [23]. Threats to internal validity relate to unaccounted influences that may impact the empirical results. The benchmark Weka data mining tool [22] was used for all learners, and all of the parameter settings have been included in this work to allow the experiments to be repeated. The parameters for the RF learners were chosen to ensure good performance in many different circumstances and to be reasonable for the imbalanced datasets. Experimentation with different settings for the $numTrees$ parameter provides guidance to the research community as to the recommended value for that parameter. ANOVA models were constructed using the SAS GLM procedure [19], and all assumptions for constructing valid ANOVA models were verified. To normalize the data and stabilize the variances, before constructing the ANOVA models, the target variable was transformed $\widehat{AUC} = \arcsin\sqrt{AUC}$ as suggested in Berenson et al. [2]. All output was validated for accuracy by members of our research group, giving us confidence in the reliability of the results.

Threats to external validity consider the generalization of the results outside the experimental setting and what limits, if any, should be applied. The comprehensive scope of our experiments, which utilize 10 real-world imbalanced datasets, greatly enhances the reliability of our conclusions. Performing numerous repetitions of cross validation greatly reduces the likelihood of anomalous results due to selecting a lucky or unlucky partition of the data. Building $54,000$ learners in these experiments allows us to be confident in the reliability of our experimental conclusions.

8 Conclusions

This work presented a comprehensive and systematic experimental analysis of the performance of the random forest learner with different $numTrees$ and $numFeatures$ parameters. In the first phase of our experimentation, $numTrees$ parameter was varied in Weka across six different ensemble sizes $(10, 50, 100, 150, 200, 500)$. Exper-

imentation also considered eight different $numFeatures$ values $(1, 2, 3, M/10, M/5, M/3, M/2,$ and $\lfloor \log_2 M + 1 \rfloor)$, thus creating 48 unique RF learners. We constructed learners using 10 real-world imbalanced datasets from a variety of application domains. The objective of the first phase of our experimentation was to provide practical guidance to data mining and machine learning researchers and practitioners on the $numTrees$ and $numFeatures$ parameters when using Weka to build RF learners with imbalanced data. Based on the results presented in this work, we suggest a default value of 100 for $numTrees$ and $\lfloor \log_2 M + 1 \rfloor$ for $numFeatures$ for binary classification in general and imbalanced datasets in particular.

The second phase of our experimentation compared the AUC and KS performance of the RF-100 learner to six different learners using 10 imbalanced datasets. Unfortunately due to space limitations, we can only present a fraction of our experimental results; however, the box plots for AUC and KS demonstrate that the performance of the RF-100 learner with the recommended parameter settings is superior to the six other learners presented here. Therefore, we highly recommend the use of random forest classifiers when learning from imbalanced data.

Much of the related work on RF has *not* focused on selecting reasonable values for the two important parameters of RF classifiers, $numTrees$ and $numFeatures$. In most previous studies, researchers and practitioners simply utilize a particular value for $numTrees$ without much supporting evidence. Our analysis provides guidelines to researchers and practitioners in data mining and machine learning to select a default value of 100 for $numTrees$ and Weka's default value for $numFeatures$ when building RF learners. To our knowledge, there has been no previous study to empirically recommend $numTrees$ and $numFeatures$ parameter values for RF learners in the Weka tool. Future work may consider the utilization of data sampling, a commonly-used solution to dealing with class imbalance, with respect to the RF learner. RF learner performance can also be compared to cost sensitive learning in future work.

References

[1] D. W. Aha. *Lazy learning*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[2] M. L. Berenson, D. M. Levine, and M. Goldstein. *Intermediate Statistical Methods and Applications: A Computer Package Approach*. Prentice-Hall, Inc., 1983.

[3] C. Blake and C. Merz. UCI repository of machine learning databases. *http://www.ics.uci.edu/ mlearn/ MLRepository.html*, 1998. Department of Information and Computer Sciences, University of California, Irvine.

[4] P. Boinee, A. D. Angelis, and G. L. Foresti. Ensembling classifiers - an application to image data classification from cherenkov telescope experiment. In *IEC (Prague)*, pages 394–398, 2005.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] C. Chen, A. Liaw, and L. Breiman. Using random forest to learn imbalanced data. *Technical Report 666, Statistics Department, University of California at Berkeley, Available at http://www.stat.berkeley.edu/users/chenchao/666.pdf*, 2004.

[8] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 2-3(29):103–130, 1997.

[9] J. D. E. Altendrof, P. Brende and L. Lessard. Fraud detection for online retail using random forests. *Technical Report*, 2005.

[10] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[11] D. J. Hand. Good practice in retail credit scorecard assessment. *Journal of the Operational Research Society*, 56:1109–1117, 2005.

[12] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[13] F. Livingston. Implementation of breiman's random forest machine learning algorithm. *Machine Learning Journal Paper*, Fall 2005.

[14] Y. Ma, L. Guo, and B. Cukic. A statistical framework for the prediction of fault-proneness. *Advances in Machine Learning Applciations in Software Engineering*, pages 237–263, 2007.

[15] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.

[16] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[17] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[18] J. R. Quinlan. *C4.5: Programs For Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[19] SAS Institute. *SAS/STAT User's Guide*. SAS Institute Inc., 2004.

[20] B. Schlkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, Massachusetts, 1999.

[21] G. M. Weiss and F. Provost. Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, (19):315–354, 2003.

[22] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, California, 2nd edition, 2005.

[23] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Boston, MA, 2000.

[24] J. Zhang and M. Zulkernine. Network intrusion detection using random forests. *Proc. Of the Third Annual Conference on Privacy, Security and Trust (PST)*, pages 53–61, October 2005.