RUSBoost: A Hybrid Approach to Alleviating Class Imbalance

Article *in* IEEE Transactions on Systems Man and Cybernetics - Part A Systems and Humans · February 2010

DOI: 10.1109/TSMCA.2009.2029559 · Source: IEEE Xplore

4 authors, including:

Taghi Khoshgoftaar
Florida Atlantic University
403 PUBLICATIONS **7,936** CITATIONS

SEE PROFILE

Jason Van Hulse
Florida Atlantic University
69 PUBLICATIONS **2,002** CITATIONS

SEE PROFILE

RUSBoost: A Hybrid Approach to Alleviating Class Imbalance

Chris Seiffert, Taghi M. Khoshgoftaar, *Member, IEEE*, Jason Van Hulse, *Member, IEEE*, and Amri Napolitano

***Abstract*—Class imbalance is a problem that is common to many application domains. When examples of one class in a training data set vastly outnumber examples of the other class(es), traditional data mining algorithms tend to create suboptimal classification models. Several techniques have been used to alleviate the problem of class imbalance, including data sampling and boosting. In this paper, we present a new hybrid sampling/boosting algorithm, called RUSBoost, for learning from skewed training data. This algorithm provides a simpler and faster alternative to SMOTEBoost, which is another algorithm that combines boosting and data sampling. This paper evaluates the performances of RUSBoost and SMOTEBoost, as well as their individual components (random undersampling, synthetic minority oversampling technique, and AdaBoost). We conduct experiments using 15 data sets from various application domains, four base learners, and four evaluation metrics. RUSBoost and SMOTEBoost both outperform the other procedures, and RUSBoost performs comparably to (and often better than) SMOTEBoost while being a simpler and faster technique. Given these experimental results, we highly recommend RUSBoost as an attractive alternative for improving the classification performance of learners built using imbalanced data.**

***Index Terms*—Binary classification, boosting, class imbalance, RUSBoost, sampling.**

I. Introduction

CREATING an effective classification model can be a challenging endeavor if the data used to train the model are imbalanced. When examples of one class greatly outnumber examples of the other class(es), traditional data mining algorithms tend to favor classifying examples as belonging to the overrepresented (majority) class. Such a model would be ineffective at identifying examples of the minority class, which is frequently the class of interest (the positive class). Typically, it is the examples of this positive class that carry the highest cost of misclassification. Therefore, new techniques are required to ensure that a model can effectively identify these most important yet rarely occurring examples.

Many techniques have been proposed to alleviate the problem of class imbalance. Two such techniques are *data sampling* and *boosting* [1]. Data sampling balances the class distribution in the training data by either adding examples to the minority class (oversampling) or removing examples from the majority class (undersampling). Several techniques for performing undersampling and oversampling have been proposed. The simplest method for resampling a data set is random resampling. *Random oversampling* balances a data set by duplicating examples of the minority class until a desired class ratio is achieved. Similarly, *random undersampling* (RUS) removes examples (randomly) from the majority class until the desired balance is achieved. There are also "intelligent" methods for oversampling and undersampling.

Both undersampling and oversampling have their benefits and drawbacks. The main drawback associated with undersampling is the loss of information that comes with deleting examples from the training data [2]. It has the benefit, however, of decreasing the time required to train the models since the training data set size is reduced. Consider a data set with 20 000 examples, where 400 (2%) of the examples belong to the positive class, while 19 600 examples belong to the negative class. After performing undersampling to achieve a balanced (50 : 50) class distribution, the new training data set will contain only 800 examples (400 positive and 400 negative). The benefit of using undersampling to balance the class distribution in this data set is that the time required to train the classifier will be relatively short, but a smaller training data set also has its drawbacks—the information contained within the 19 200 deleted examples is lost.

On the other hand, oversampling results in no lost information. Every example in the original training data set also appears in the resampled training data set. Depending on the oversampling algorithm used, the resampled data set also includes either duplicate or new minority class examples. While no information is lost during oversampling, it is not without its drawbacks. When oversampling is performed by duplicating examples, it can lead to overfitting [3]. Whether duplicating examples or creating new ones, oversampling increases model training times. In our previous example, the data set contained only 400 minority class examples but 19 600 majority class examples. While undersampling (to achieve a 50 : 50 class ratio) results in a data set with only 800 examples, oversampling would create a training data set with 39 200 examples (19 600 positive and 19 600 negative). Training a model on such a large data set would obviously take much longer than if undersampling was used.

Another technique that can be used to improve classification performance is boosting. While many data sampling techniques are designed specifically to address the class imbalance problem, boosting is a technique that can improve the performance

Manuscript received April 18, 2008. First published October 30, 2009; current version published December 16, 2009. This paper was recommended by Associate Editor M. Dorneich.

C. Seiffert, deceased, was with the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: chrisseiffert@gmail.com).

T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano are with the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: taghi@cse.fau.edu; jvanhulse@gmail.com; amrifau@gmail.com).

Digital Object Identifier 10.1109/TSMCA.2009.2029559

of any weak classifier (regardless of whether the data are imbalanced). The most common boosting algorithm is *AdaBoost* [4], which iteratively builds an ensemble of models. During each iteration, example weights are modified with the goal of correctly classifying examples in the next iteration, which were incorrectly classified during the current iteration. Upon completion, all constructed models participate in a weighted vote to classify unlabeled examples. Such a technique is particularly effective at dealing with class imbalance because the minority class examples are most likely to be misclassified and therefore given higher weights in subsequent iterations. In fact, boosting has been referred to as a kind of advanced data sampling technique [1]. Boosting can be performed either by "reweighting" or "resampling" [5]. When boosting by reweighting, the modified example weights are passed directly to the base learner during each iteration. However, not all learning algorithms are designed to incorporate example weights in their decision-making processes. Therefore, in this paper, we utilize boosting by resampling, which resamples the training data according to the examples' assigned weights. It is this resampled training data set that is used to construct the iteration's model.

In this paper, we present a novel hybrid data sampling/boosting algorithm called RUSBoost, which is designed to improve the performance of models trained on skewed data. We compare the performance of RUSBoost to that of SMOTEBoost [6], which is another algorithm that combines boosting with data sampling. Both RUSBoost and SMOTEBoost introduce data sampling into the AdaBoost algorithm. SMOTEBoost does so using an oversampling technique (synthetic minority oversampling technique (SMOTE) [7]) which creates new minority class examples by extrapolating between existing examples. RUSBoost applies RUS, which is a technique that randomly removes examples from the majority class. RUS has been shown to perform very well despite its simplicity [8]. The motivations for introducing RUS into the boosting process are simplicity, speed, and performance.

SMOTE is a more complex and time-consuming data sampling technique than RUS. Therefore, SMOTEBoost (which uses SMOTE) is more complex and time consuming to perform than RUSBoost. Furthermore, SMOTE is an oversampling technique which carries the drawback of increased model training time. SMOTEBoost magnifies this drawback since boosting requires the training of an ensemble of models (many models with increased training times must be constructed). Consider our previous example data set, where only 400 of the 20 000 examples belong to the minority class. Using SMOTEBoost, n (where n is the number of boosting iterations to be performed) models would have to be built using data sets with as many as 39 988 examples. This number is larger than the previously mentioned 39 200 examples due to the application of boosting by resampling [5]. When SMOTEBoost is implemented using boosting by reweighting, training data sets with 39 200 examples are used.

On the other hand, RUS decreases the time required to construct a model, which is a key benefit particularly when creating an ensemble of models, which is the case in boosting. While both RUSBoost and SMOTEBoost would require that n models be trained, in our example data set, the n models built using RUSBoost would be trained using as few as two examples. This unlikely scenario would only occur under extreme conditions using boosting by resampling. Using boosting by reweighting, the number of examples used to train each model in the RUSBoost ensemble would be the previously mentioned 800 (compared to 39 200 examples using SMOTEBoost). RUSBoost, therefore, has a clear advantage over SMOTEBoost with respect to training time. The main drawback of RUS, which is the loss of information, is greatly overcome by combining it with boosting. While certain information may be absent during a given iteration of boosting, it will likely be included when training models during other iterations. Our results, which are based on four different performance metrics, demonstrate the strength of RUSBoost. Even using performance metrics where RUS performs poorly (due to loss of information), RUSBoost performs extremely well. Our results show that the simpler and quicker RUSBoost algorithm performs favorably when compared to SMOTEBoost, often resulting in significantly better classification performance.

We present a comprehensive investigation of RUSBoost, comparing its performance to that of SMOTEBoost, as well as AdaBoost, RUS, and SMOTE, which are the main components of RUSBoost and SMOTEBoost. Using four different learners and four performance metrics, we evaluate the performance of each of these algorithms using 15 skewed data sets from various application domains. Our results show that SMOTEBoost and RUSBoost both result in significantly better performance than their components and that RUSBoost typically performs as well as or better than SMOTEBoost despite its relative simplicity and efficiency.

The remainder of this paper is organized as follows. Section II presents related works, and Section III describes the RUSBoost algorithm. Section IV provides the details of our experiments, and the experimental results are provided in Section V. We conclude in Section VI.

II. Related Work

Much research has been performed with respect to the class imbalance problem. Weiss [1] provides a survey of the class imbalance problem and techniques for reducing the negative impact imbalance that has on classification performance. The study identifies many methods for alleviating the problem of class imbalance, including data sampling and boosting, which are the two techniques investigated in this paper. Japkowicz [9] presents another study addressing the issue of class imbalance, including an investigation of the types of imbalance that most negatively impact classification performance, and a small case study comparing several techniques for alleviating the problem.

Data sampling has received much attention in research related to class imbalance. Data sampling attempts to overcome imbalanced class distributions by adding examples to (oversampling) or removing examples from (undersampling) the data set. The simplest form of undersampling is RUS. RUS randomly removes examples from the majority class until a desired class distribution is found. While there is no universally accepted optimal class distribution, a balanced (50 : 50) distribution is often considered to be near optimal [10]. However, when

examples from the minority class are very rare, a ratio closer to 35 : 65 (minority:majority) may result in better classification performance [11].

In addition to random data sampling techniques, several more "intelligent" algorithms for resampling data have been proposed. Barandela *et al.* [12] and Han *et al.* [13] examine the performance of some of these "intelligent" data sampling techniques, such as SMOTE, borderline SMOTE, and Wilson's editing. Van Hulse *et al.* [8] examine the performance of seven different data sampling techniques (both "intelligent" and random) using a large number of different learning algorithms and experimental data sets, finding both RUS and SMOTE to be very effective data sampling techniques.

Another technique for dealing with class imbalance is boosting. While boosting is not specifically designed to handle the class imbalance problem, it has been shown to be very effective in this regard [14]. The most commonly used boosting algorithm is AdaBoost [4], which has been shown to improve the performance of any weak classifier, provided that the classifier results in better performance than random guessing. Several variations have been proposed to make AdaBoost cost sensitive [15]–[17] or to improve its performance on imbalanced data [18]–[20]. One of the most promising of these techniques is SMOTEBoost [6]. SMOTEBoost combines an intelligent oversampling technique (SMOTE) with AdaBoost, resulting in a highly effective hybrid approach to learning from imbalanced data. Our proposed technique, which is RUSBoost, is based on the SMOTEBoost algorithm but provides a faster and simpler alternative for learning from imbalanced data with performance that is usually as good (and often better) than that of SMOTEBoost.

Closely related to the issue of class imbalance is cost-sensitive learning. Weiss *et al.* [21] compare the performances of oversampling, undersampling, and cost-sensitive learning when dealing with data that have both an imbalanced class distribution and unequal error costs. Sun *et al.* [17] present an in-depth examination of cost-sensitive boosting. Chawla *et al.* [22] perform a detailed evaluation of a wrapper-based sampling approach to minimize misclassification cost. A detailed evaluation of RUSBoost as a cost-sensitive learning technique, including a comparison to existing methodologies, is left as a future work.

A. Background

This paper compares the performance of RUSBoost to that of its components (RUS and AdaBoost), showing that RUSBoost results in better classification performance than either of its components alone. In addition, we compare the performance of RUSBoost to that of its predecessor, which is SMOTEBoost, as well as SMOTE. The following sections briefly describe these techniques. In addition, we present the results for each baseline learner without using either sampling or boosting and label the results as "None." For simplicity, throughout this paper, we refer to all of these techniques (and RUSBoost) as "sampling techniques."

1) AdaBoost: Boosting is a metalearning technique designed to improve the classification performance of weak learners by iteratively creating an ensemble of weak hypotheses which are combined to predict the class of unlabeled examples. This paper uses AdaBoost, which is a well-known boosting algorithm shown to improve the classification performance of weak classifiers. We present a brief synopsis of AdaBoost. For the complete details of the AdaBoost algorithm, please refer to Freund and Schapire's work [4].

Initially, all examples in the training data set are assigned with equal weights. During each iteration of AdaBoost, a weak hypothesis is formed by the base learner. The error associated with the hypothesis is calculated, and the weight of each example is adjusted such that misclassified examples have their weights increased while correctly classified examples have their weights decreased. Therefore, subsequent iterations of boosting will generate hypotheses that are more likely to correctly classify the previously mislabeled examples. After all iterations are completed, a weighted vote of all hypotheses is used to assign a class to the unlabeled examples. In this paper, all boosting algorithms (AdaBoost, SMOTEBoost, and RUSBoost) are performed using ten iterations. Preliminary experiments with AdaBoost using the same data sets as in this paper showed no significant improvement between 10 and 50 iterations.

Since boosting assigns higher weights to misclassified examples and minority class examples are those most likely to be misclassified, it stands to reason that minority class examples will receive higher weights during the boosting process, making it similar in many ways to cost-sensitive classification [23] (a technique for alleviating the class imbalance problem that is not discussed in this paper). In effect, boosting alters the distribution of the training data (through weighting, not altering the number of examples); thus, it can also be thought of as an advanced data sampling technique [10]. For simplicity, we refer to boosting as one of the five "sampling techniques" investigated in this paper.

2) RUS: Data sampling techniques attempt to alleviate the problem of class imbalance by adjusting the class distribution of the training data set. This can be accomplished by either removing examples from the majority class (undersampling) or adding examples to the minority class (oversampling). One of the most common data sampling techniques (largely due to its simplicity) is RUS. Unlike more complex data sampling algorithms, RUS makes no attempt to "intelligently" remove examples from the training data. Instead, RUS simply removes examples from the majority class *at random* until a desired class distribution is achieved. In this paper, we use three postsampling class distributions: 35, 50, and 65. These numbers represent the percentage of examples in the postsampling data set, which belong to the minority class. The same three values are used when performing SMOTE, SMOTEBoost, and RUSBoost.

3) SMOTE: Chawla *et al.* [7] proposed an intelligent oversampling method called SMOTE. SMOTE adds new artificial minority examples by extrapolating between preexisting minority instances rather than simply duplicating original examples. The newly created instances cause the minority regions of the feature space to be fuller and more general. The technique first finds the k nearest neighbors of each minority example (this paper recommends $k = 5$). The artificial examples are

Algorithm RUSBoost

Given: Set S of examples $(x_1, y_1), ..., (x_m, y_m)$ with minority class $y^r \in Y$, $|Y| = 2$

Weak learner, $WeakLearn$

Number of iterations, T

Desired percentage of total instances to be represented by the minority class, N

1 Initialize $D_1(i) = \frac{1}{m}$ for all i.

2 Do for $t = 1, 2, ..., T$

a Create temporary training dataset S'_t with distribution D'_t using random undersampling

b Call $WeakLearn$, providing it with examples S'_t and their weights D'_t.

c Get back a hypothesis $h_t : X \times Y \rightarrow [0, 1]$.

d Calculate the pseudo-loss (for S and D_t):
$$\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y)).$$

e Calculate the weight update parameter:
$$\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t}.$$

f Update D_t:
$$D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}.$$

g Normalize D_{t+1}: Let $Z_t = \sum_i D_{t+1}(i)$.
$$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}.$$

3 Output the final hypothesis:
$$H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}.$$

Fig. 1. RUSBoost algorithm.

then generated in the direction of some or all of the nearest neighbors, depending on the amount of oversampling desired. If 200% oversampling is specified, then synthetic examples are generated randomly along the line segments connecting each minority example to two of its five nearest neighbors. If x_i is the minority example being examined, x_j is one of the selected nearest neighbors of x_i, and x_n is the new example being added to the data set, then $x_n = u(x_j - x_i) + x_i$, where u is a uniform random number between zero and one. This sampling process causes a classifier to learn a larger and more general decision region in the feature space, ideally alleviating the problem caused by class imbalance.

4) SMOTEBoost: SMOTEBoost, which was proposed by Chawla *et al.* [6], combines the SMOTE algorithm with AdaBoost, resulting in a hybrid sampling/boosting algorithm that outperforms both SMOTE and AdaBoost. As is the case with RUSBoost, SMOTEBoost is based on the AdaBoost.M2 algorithm. Prior to constructing the weak hypothesis during each round of boosting, SMOTE is applied to the training data to achieve a more balanced training data set. Therefore, the algorithm for SMOTEBoost is very similar to that of RUSBoost, which is presented in Section III. Fig. 1 can be modified to represent the SMOTEBoost algorithm by changing step 2a to

Create temporary training data set S'_t with distribution D'_t using SMOTE

The application of SMOTE at this point has two drawbacks that RUSBoost is designed to overcome. First, it increases the complexity of the algorithm. SMOTE must find the k nearest neighbors of the minority class examples and extrapolate between them to make new examples. RUS, on the other hand, simply deletes the majority class examples at random. Second, since SMOTE is an oversampling technique, it results in longer model training times. This effect is compounded by SMOTEBoost's use of boosting, since many models must be built using larger training data sets. On the other hand, RUS results in smaller training data sets and, therefore, shorter model training times.

III. RUSBoost

Fig. 1 shows the RUSBoost algorithm. RUSBoost is based on the SMOTEBoost algorithm [6], which is, in turn, based on the AdaBoost.M2 algorithm [4]. SMOTEBoost improves upon AdaBoost by introducing an intelligent oversampling technique (SMOTE [7]), which helps to balance the class distribution, while AdaBoost improves the classifier performance using these balanced data. RUSBoost achieves the same goal but uses RUS rather than SMOTE. The result is a simpler algorithm with faster model training times and favorable performance.

Let x_i be a point in the feature space X and y_i be a class label in a set of class labels Y. Each of the m examples in the data set (S) can be represented by the tuple (x_i, y_i). Let t be an iteration between one and the maximum number of iterations T (number of classifiers in the ensemble), h_t be the weak hypothesis (trained using some classification algorithm, *WeakLearn*) trained on iteration t, and $h_t(x_i)$ be the output of hypothesis h_t, for instance, x_i (which may be a numeric confidence rating). Let $D_t(i)$ be the weight of the ith example on iteration t (the weights are typically normalized to sum to one).

In step 1, the weights of each example are initialized to $1/m$, where m is the number of examples in the training data set. In step 2, T weak hypotheses are iteratively trained, as shown in steps 2a–2g. In step 2a, RUS is applied to remove the majority class examples until $N\%$ of the new (temporary) training data set S'_t belongs to the minority class. For example, if the desired class ratio is 50 : 50, then the majority class examples are randomly removed until the numbers of majority and minority class examples are equal. As a result, S'_t will have a new weight distribution D'_t. In step 2b, S'_t and D'_t are passed to the base learner, *WeakLearn*, which creates the weak hypothesis h_t (step 2c). The pseudoloss ϵ_t (based on the original training data set S and weight distribution D_t) is calculated in step 2d. In step 2e, the weight update parameter α is calculated as $\epsilon_t/(1 - \epsilon_t)$. Next, the weight distribution for the next iteration D_{t+1} is updated (step 2f) and normalized (step 2g). After T iterations of step 2, the final hypothesis $H(x)$ is returned as a weighted vote of the T weak hypotheses (step 3).

IV. Experiments

A. Data Sets

The performance of various imbalance-handling techniques (including our new algorithm, RUSBoost) is evaluated using 15 data sets from various application domains. Table I provides the characteristics of these data sets, including the number of examples (Size) in each data set, the number of minority class

TABLE I
DATA SET CHARACTERISTICS

Dataset	Size	# min	% min	# attr
SP3	3541	47	1.33	43
MAMMOGRAPHY	11183	260	2.32	7
SOLARFLAREF	1389	51	3.67	13
CAR3	1728	69	3.99	7
CCCS12	282	16	5.67	9
SP1	3649	229	6.28	43
PC1	1107	76	6.87	16
GLASS3	214	17	7.94	10
CM1	505	48	9.50	16
PENDIGITS5	10992	1055	9.60	17
SATIMAGE4	6435	626	9.73	37
ECOLI4	336	35	10.42	8
SEGMENT5	2310	330	14.29	20
CONTRA2	1473	333	22.61	10
VEHICLE1	846	212	25.06	19

examples (#min), the percentage of all examples belonging to the minority class (%min; commonly referred to as the *level of imbalance*), and the number of attributes in the data sets, including the dependent attribute (#attr). These data sets represent a wide variety of data set sizes, imbalance levels, and application domains. The size of the data sets ranges from 214 examples (Glass3) to 11 183 examples (Mammography). The data sets have anywhere from 7 to 43 attributes. In Table I, the data sets are sorted from the most imbalanced (SP3 with only 1.33% examples belonging to the minority class) to the least imbalanced (Vehicle1 with 25.06% examples from the minority class).

Three of the data sets (SP1, SP3, and CCCS12) are proprietary software project data sets obtained from Nortel Networks and the Department of Defense. The remaining 12 data sets are publicly available. PC1 and CM1 can be downloaded from the National Aeronautics and Space Administration Metric Data Program website [24]. The Mammography data set is from the medical diagnosis domain and was generously provided by Dr. N. Chawla [7]. The remaining data sets were obtained from the popular University of California–Irvine repository [25], and they represent various application domains. Since this paper considers only the binary classification problem, it was necessary to transform some of the data sets to have a binary class. In the case of multiclass data sets, a single class was selected to be the positive class, while the remaining classes were combined to make up the negative class. In the case of data sets with a continuous dependent variable, a domain-specific threshold was applied to divide the examples into two classes. Each of the techniques evaluated in this paper, however, can be applied in a multiclass environment. AdaBoost can be applied to multiclass data, and data sampling (RUS or SMOTE) can be performed to achieve any class distribution, regardless of the number of classes.

B. Learners

This paper uses four learners, all of which are implemented in Weka [26], which is an open-source data mining suite. C4.5 [27] is a decision tree learner that uses an entropy-based splitting criterion stemming from information theory [28]. Two versions of C4.5 are used in our experiments. C4.5D uses the default Weka parameters, while C4.5N disables pruning and uses Laplace smoothing [10]. The Weka implementation of C4.5 is J48. Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [29] is a rule-based learner that modifies the incremental reduced error pruning algorithm [30] to improve accuracy without sacrificing efficiency. Naive Bayes (NB) [31] utilizes Bayes's rule of conditional probability to classify examples. It is "naive" in that it assumes that all predictor variables are conditionally independent. The default Weka parameters were used for both RIPPER and NB.

C. Performance Metrics

This paper uses four different performance metrics to evaluate the models constructed for our experiments, all of which are more suitable than the overall accuracy when dealing with class imbalance. While many studies select just one metric, we recognize that there is no universally accepted metric for evaluating learner performance when data are imbalanced. Depending on the selection of performance metric, very different results can be observed. Although an evaluation of different performance metrics is beyond the scope of this paper, we include these four metrics to illustrate the power of RUSBoost, which performs well even when its components (RUS and AdaBoost) do not.

In binary classification problems, classifications can be grouped into one of four categories. These four categories are used to derive the four performance metrics used in this paper. If x_i is an example from a data set D with class c_j, $j = 0$ or 1, and if $c(x_i)$ and $\hat{c}(x_i)$ are the actual and predicted classes of x_i, then

$$x_i \text{ is a } true\ positive \text{ (tpos) if } c(x_i) = c_1 = \hat{c}(x_i)$$
$$x_i \text{ is a } true\ negative \text{ (tneg) if } c(x_i) = c_0 = \hat{c}(x_i)$$
$$x_i \text{ is a } false\ positive \text{ (fpos) if } c(x_i) = c_0 \neq \hat{c}(x_i)$$
$$x_i \text{ is a } false\ negative \text{ (fneg) if } c(x_i) = c_1 \neq \hat{c}(x_i).$$

Based on the aforementioned definitions, several basic performance metrics can be defined, which are the basis for the four metrics that are used in this paper. The total number of instances in a data set that are false negatives is denoted by $\#fneg$ and, similarly for $\#fpos$, $\#tpos$ and $\#tneg$. If N_{c_j} denotes the number of instances in class c_j, then let $N = N_{c_0} + N_{c_1}$. Three basic measures of classifier performance can be defined as

$$\text{true positive rate (or recall)} = \frac{\#tpos}{N_{c_1}} \tag{1}$$

$$\text{false positive rate} = \frac{\#fpos}{N_{c_o}} \tag{2}$$

$$\text{precision} = \frac{\#tpos}{\#tpos + \#fpos}. \tag{3}$$

1) ROC Curves: One of the most popular methods for evaluating the performance of learners built using imbalanced data is *receiver operating characteristic* [32], or *ROC*, curves. *ROC* curves graph the true positive rate on the y-axis versus the false positive rate on the x-axis. The resulting curve illustrates the tradeoff between detection and false alarm rates. The *ROC*

curve illustrates the performance of a classifier across the complete range of possible decision thresholds and accordingly, does not assume any particular misclassification costs or class prior probabilities. The area under the ROC curve (A-ROC) is used to provide a single numerical metric for comparing model performances.

2) PRC Curves: The *precision–recall characteristic* (PRC) [33] curve has also been used to evaluate the performance of models built using skewed data. Like ROC curves, the PRC curve plots the *recall*, or true positive rate, on the y-axis. However, the difference between the ROC and PRC curves is that the PRC curve plots the *precision* on the x-axis. Precision is the proportion of examples predicted as belonging to the positive class to examples that actually belong to the positive class. The implications of this difference have been investigated by Davis and Goadrich [34]. As with the ROC curves, the area under the PRC curve (A-PRC) is used as a single metric for comparing the performances of different models.

3) K–S Statistic: The Kolmogorov–Smirnov (K–S) statistic measures the maximum difference between the empirical distribution functions of the posterior probabilities of instances in each class. In other words, let $F_{c_i}(t) = P(p(x) \leq t \,|\, c_i)$, where $0 \leq t \leq 1$. $F_{c_i}(t)$ is estimated by the proportion of class c_i instances $\leq t$

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

Then, the K–S statistic is defined as

$$K\!-\!S = \max_{t \in [0,1]} \left| F_{c_1}(t) - F_{c_2}(t) \right|.$$

The larger the distance between the two distribution functions, the better the learner is able to separate the two classes. The maximum possible value for K–S is one (perfect separation), with a minimum of zero (no separation). The K–S statistic is a commonly used metric of classifier performance in the credit scoring application domain [35].

4) $F\!-\!Measure$*:* The final performance metric used in this paper is the $F\!-\!measure$. Like the PRC curve, this metric is derived from *recall* and *precision*. It is the only metric used in this paper that is decision threshold dependent. A default decision threshold of 0.5 is used. The formula for $F\!-\!measure$, which is given in (4), uses a tunable parameter β to indicate the relative importance of recall and precision. That is, one can modify β to place more emphasis on either recall or precision. Typically, $\beta = 1$ is used (as is the case in our study)

$$F\!-\!measure = \frac{(1+\beta^2) \times recall \times precision}{\beta^2 \times recall + precision}. \quad (4)$$

D. ANOVA Analysis

The results presented in this paper are tested for statistical significance at the $\alpha = 5\%$ level using a one-factor analysis of variance (ANOVA) [36]. An ANOVA model can be used to test the hypothesis that the classification performances for each level of the main factor(s) are equal against the alternative hypothesis that at least one is different. In this paper, we use a one-factor ANOVA model, which can be represented as

$$\psi_{jn} = \mu + \theta_j + \epsilon_{jn}$$

where

- ψ_{jn} response (i.e., A-ROC, K–S, A-PRC, or $F\!-\!measure$) for the nth observation of the jth level of experimental factor θ;
- μ overall mean performance;
- θ_j mean performance of level j for factor θ;
- ϵ_{jn} random error.

The main factor θ is the sampling technique. That is, we are testing to see if the average performances of the six levels (RUSBoost, SMOTEBoost, RUS, SMOTE, AdaBoost, and None) of θ are equal. If the alternative hypothesis (that at least one level of θ is different) is accepted, numerous procedures can be used to determine which are different. This involves a pairwise comparison of each sampling technique's mean performance, with the null hypothesis that they are equal. In this paper, we use Tukey's honestly significant difference (HSD) test to identify which levels of θ are significantly different.

E. Experimental Design Summary

Each of the four learners, as well as AdaBoost, is implemented within the Weka data mining suite [26]. Weka was extended by our research group to include the four data sampling techniques (SMOTE, SMOTEBoost, RUS, and RUSBoost) and to provide the four performance metrics used to evaluate the classification performance. A statistical analysis, via ANOVA modeling, was performed using SAS [37].

All experiments are performed using tenfold cross-validation. Each data set is split into ten partitions, nine of which are used to train the model, while the remaining (hold-out) partition is used to test the model. This process is repeated ten times so that each partition acts as test data once. In addition, ten independent runs of this procedure are performed to eliminate any biasing that may occur during the random partitioning process. Therefore, each data set results in $10 \times 10 = 100$ experimental data sets. We use 15 data sets from various application domains with various levels of imbalance and size, resulting in a total of $15 \times 100 = 1500$ derived experimental data sets.

Using each of these experimental data sets, four base learners are used to train the models in conjunction with each of the six techniques: RUSBoost, SMOTEBoost, RUS, SMOTE, AdaBoost, and None. Four of the techniques (RUSBoost, SMOTEBoost, RUS, and SMOTE) are each performed three times, each with a different "target class distribution" (35%, 50%, or 65%). AdaBoost and None (no sampling) are each performed once. Therefore, $4 \times 3 + 2 = 14$ sampling technique/parameter combinations are used. Using four base learners, there are $4 \times 14 = 56$ combinations of sampling technique and learner evaluated. In total, 1500 data sets $\times$ 56 combinations $= 84\,000$ models are evaluated to formulate the results presented in this paper. All boosting algorithms were performed using ten

TABLE II
AVERAGE PERFORMANCES OF THE SAMPLING TECHNIQUES ACROSS ALL LEARNERS AND DATA SETS

	A-ROC		K-S		A-PRC		F-measure	
Technique	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD
RUSBoost	0.8704	A	0.7325	A	0.5629	A	0.4971	A
SMOTEBoost	0.8674	A	0.7284	A	0.5707	A	0.4976	A
AdaBoost	0.8394	B	0.6813	B	0.5253	B	0.4506	C
RUS	0.8243	C	0.6507	D	0.3916	E	0.4228	D
SMOTE	0.8199	C	0.6633	C	0.4776	C	0.4755	B
None	0.7670	D	0.5355	E	0.4308	D	0.4117	E

iterations. Experiments performed using more iterations did not result in significantly different results.

V. EMPIRICAL RESULTS

This section presents the results of our experiments with RUSBoost, SMOTEBoost, and their components (RUS, SMOTE, and AdaBoost). For comparison purposes, the performance of the baseline learner without either sampling or boosting is included and denoted by "None." The results are presented in a top–down fashion. That is, we begin by comparing the average performances of these sampling techniques across all 15 data sets and all four learners in Section V-A. We then narrow our view, examining the performance of the techniques on a per-learner basis in Section V-B. Section V-C investigates the performance of each technique for each combination of data set and learner. Finally, Section V-D directly compares RUSBoost and SMOTEBoost.

A. Overall Results

Table II presents the performance of each sampling technique averaged across all learners and data sets. This table gives a general view of the performance of each technique using each of the four performance metrics. Table II provides both the numerical average performance (Mean) and the results of Tukey's HSD test. If two techniques have the same letter in the column labeled "HSD," then their mean performances were not significantly different according to the HSD test at the 95% confidence level. For example, using A-ROC to measure the performance of the learners, the performances of RUSBoost and SMOTEBoost are not significantly different (they are both in group A), but both RUSBoost and SMOTEBoost perform significantly better than AdaBoost (which belongs to group B). Although it does not occur in this table, it is possible for a technique to belong to more than one group.

For a more detailed description of the entire distribution of classification performance across all data sets and learners, two boxplot comparisons of the six methodologies for handling imbalance are shown in Figs. 2 and 3 (we exclude the boxplots for A-ROC and $F-measure$ since the boxplot of A-ROC is similar to that of K–S and the boxplot of $F-measure$ is similar to that of A-PRC). Boxplots allow for an easy comparison of the techniques at various points in the distribution, not just the mean as in Table II. For example, it is clear that the distribution of the A-PRC values for RUSBoost is much more favorable than that of RUS—the 75th, median, and 25th percentiles of the distribution of the A-PRC values for RUSBoost are much larger than that of RUS.

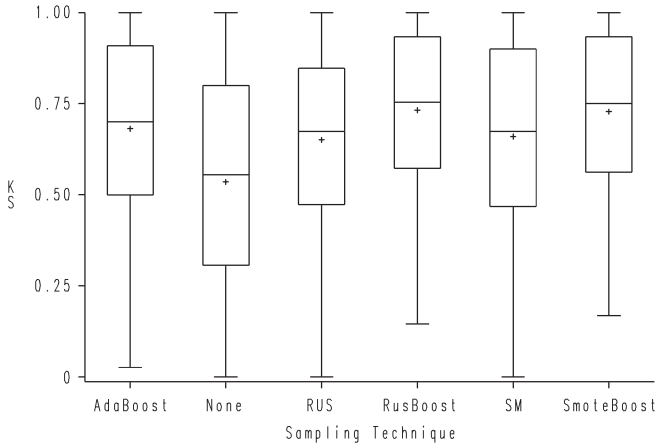

Fig. 2. K–S boxplot.

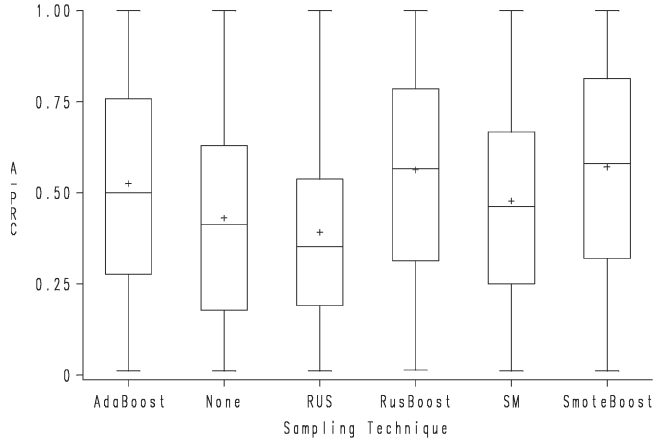

Fig. 3. A-PRC boxplot.

The following observations are derived from the data in Table II and Figs. 2 and 3.

1) According to the HSD test, there is no significant difference between the performances of RUSBoost and SMOTEBoost, regardless of the performance metric, when averaging across all data sets and learners. In all cases, both SMOTEBoost and RUSBoost belong to group A. The boxplots of RUSBoost and SMOTEBoost are also very similar.
2) Using A-ROC and K–S to measure performance, RUSBoost has a slightly better performance (presented in the Mean column), but the difference is not significant. A similar conclusion holds for SMOTEBoost using A-PRC and $F-measure$.

TABLE III
PERFORMANCES OF THE SAMPLING TECHNIQUES AVERAGED ACROSS ALL DATA SETS

		C45D		C45N		NB		RIPPER	
Metric	Technique	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD
A-ROC	RUSBoost	0.8729	A	0.8839	A	0.8468	A	0.8778	A
	SMOTEBoost	0.8683	A	0.8806	A	0.8485	A	0.8722	A
	AdaBoost	0.8492	B	0.8485	C	0.8125	C	0.8475	B
	RUS	0.8098	C	0.8436	C	0.8332	B	0.8108	C
	SMOTE	0.7780	D	0.8641	B	0.8366	B	0.8009	D
	None	0.7049	E	0.8466	C	0.8344	B	0.6820	E
K-S	RUSBoost	0.7384	A	0.7531	A	0.6935	A	0.7449	A
	SMOTEBoost	0.7335	A	0.7467	A	0.6933	A	0.7399	A
	AdaBoost	0.7017	B	0.6995	B	0.6293	C	0.6946	B
	RUS	0.6434	C	0.6749	C	0.6594	B	0.6253	D
	SMOTE	0.6298	C	0.7105	B	0.6612	B	0.6518	C
	None	0.4425	D	0.6698	C	0.6608	B	0.3691	E
A-PRC	RUSBoost	0.5859	AB	0.6029	A	0.4842	A	0.5787	B
	SMOTEBoost	0.5929	A	0.6024	A	0.4862	A	0.6013	A
	AdaBoost	0.5737	B	0.5708	B	0.3954	D	0.5612	C
	RUS	0.3527	E	0.4180	D	0.4444	C	0.3512	E
	SMOTE	0.4613	C	0.5297	C	0.4612	B	0.4583	D
	None	0.3896	D	0.5148	C	0.4527	BC	0.3662	E
F-Measure	RUSBoost	0.5222	A	0.5296	A	0.4236	A	0.5130	B
	SMOTEBoost	0.5131	AB	0.5171	A	0.4275	A	0.5328	A
	AdaBoost	0.4804	C	0.4835	B	0.3853	B	0.4530	C
	RUS	0.4343	D	0.4338	D	0.3825	B	0.4406	C
	SMOTE	0.5006	B	0.4979	B	0.3881	B	0.5152	B
	None	0.3948	E	0.4630	C	0.3767	B	0.4122	D

3) Both RUSBoost and SMOTEBoost perform significantly better than AdaBoost, RUS, and SMOTE. In other words, the application of hybrid sampling/boosting is better than boosting or sampling alone.
4) AdaBoost performs significantly better than RUS, SMOTE, and None for three of the four performance metrics, confirming the results in a previous work [14], which claimed that boosting frequently outperforms data sampling when the training data are skewed.
5) While there is no significant difference between the performances of SMOTE and RUS using A-ROC to measure performance, SMOTE significantly outperforms RUS for the other three metrics. In fact, when using A-PRC to measure performance, RUS results in a significantly worse performance than if no sampling technique was applied at all (None).
6) While RUS is usually outperformed by SMOTE, it is not the case that RUSBoost is outperformed by SMOTEBoost. Instead, the performances of SMOTEBoost and RUSBoost are very similar. This indicates that the combination of RUS with boosting indeed helps overcome the main drawback of RUS (the loss of information), which is evident when measuring performance with A-PRC and $F-measure$. However, SMOTE's main drawback (increased model training time due to larger training data sets) is amplified by its combination with boosting. Given the similar performance between SMOTEBoost and RUSBoost, one would prefer the simpler and faster technique: RUSBoost.

B. Results by Learner

This section investigates the performance of the five sampling techniques by learner when classification models are trained using C4.5D, C4.5N, NB, and RIPPER. Table III shows the performance of each learner/sampling technique pair for each of the four performance measures. As in Table II, this table presents the average performance (across all 15 data sets) and the results of HSD testing. When two sampling techniques have the same letter in the HSD column, there is no statistically significant difference in their performances.

Many of the trends are similar to what was previously observed in Section V-A; however, we summarize the following new findings here.

1) With RIPPER as the base learner and using the $F-measure$ and A-PRC to measure performance, SMOTEBoost is preferred over RUSBoost. In all other situations, RUSBoost and SMOTEBoost are not significantly different from one another, as determined by the HSD test.
2) In almost all situations, RUSBoost and SMOTEBoost are significantly better than RUS, SMOTE, and AdaBoost (the exceptions are C4.5D with A-PRC and $F-measure$, and RIPPER with $F-measure$). That is, the improvements obtained by hybrid sampling/boosting are not specific to any single learner or performance measure.
3) AdaBoost does not perform well with the NB classifier. Relative to the other three learners, a comparison between AdaBoost and sampling (either RUS or SMOTE) shows that RUS or SMOTE is significantly better than AdaBoost in only three of the 12 scenarios. In other words, with the exception of NB, AdaBoost generally performs similar to, or better than, sampling.
4) For the NB classifier, SMOTE, RUS, and AdaBoost do not significantly improve the performance of the baseline learner and, in some cases, can hurt performance. Despite

TABLE IV
RESULTS FOR ALL DATA SETS USING C4.5D (A-ROC)

	RUSBoost		SMOTEBoost		AdaBoost		RUS		SMOTE		None	
Dataset	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD
CAR	0.9964	A	0.9937	A	0.9940	A	0.9420	C	0.9731	B	0.5000	D
CCCS12	0.9579	A	0.9582	A	0.9477	A	0.8786	B	0.8498	B	0.7826	C
CM1	0.7543	A	0.7364	A	0.7168	AB	0.6751	CB	0.6473	C	0.5613	D
CONTRACEPTIVE	0.6982	A	0.6770	CDB	0.6605	D	0.6898	AB	0.6651	CD	0.6845	CAB
ECOLI	0.9320	A	0.9308	A	0.9143	A	0.8618	B	0.8468	B	0.7726	C
GLASS	0.7620	AB	0.8341	A	0.7764	AB	0.7003	B	0.7508	B	0.7479	B
MAMMOGRAPHY	0.9356	A	0.9103	B	0.8846	C	0.8804	C	0.8408	D	0.8274	D
PC1	0.8466	A	0.8705	A	0.8419	A	0.7747	B	0.7224	C	0.6724	D
PENDIGITS	0.9996	A	0.9996	A	0.9993	A	0.9745	C	0.9798	B	0.9713	C
SATIMG	0.9471	A	0.9506	A	0.9399	A	0.8199	B	0.7818	C	0.7506	D
SEGMENT	0.9964	A	0.9970	A	0.9959	A	0.9583	B	0.9622	B	0.9410	C
SOLARFLARE	0.8824	A	0.8442	B	0.8311	B	0.8651	AB	0.7651	C	0.5136	D
SP1	0.7790	A	0.7658	AB	0.7514	B	0.6773	C	0.6141	D	0.5895	E
SP3	0.7704	A	0.7149	B	0.6611	C	0.6957	BC	0.5163	D	0.5027	D
VEHICLE	0.8359	A	0.8409	A	0.8224	A	0.7535	B	0.7544	B	0.7557	B
Average / # A's	0.8729	15	0.8683	11	0.8492	10	0.8098	2	0.7780	0	0.7049	1

this, both RUSBoost and SMOTEBoost do significantly improve the performance of this learner.

5) The results using C4.5D and RIPPER with the A-ROC performance measure agree with the results from a previous study [14], where AdaBoost was shown to significantly outperform RUS, which, in turn, significantly outperformed SMOTE.
6) The performance using A-PRC and $F-measure$ is considerably different compared to that of A-ROC or K–S, particularly where RUS is concerned. In general, RUS (which is typically one of the best sampling techniques when evaluated with A-ROC or K–S) performs very poorly using A-PRC and $F-measure$. In most cases, RUS does not improve the baseline learner and often results in significantly worse performance. Although RUS is shown to be a poorly performing sampling technique using these two measures, RUSBoost performs quite well, once again demonstrating how the combination of boosting and RUS can help alleviate the main drawback (information loss) obtained using RUS alone.

C. Results by Data Set and Learner

In this section, we expand upon the results presented in Section V-B by data set. That is, we no longer consider the average performance across all data sets but instead investigate the performance of the various sampling techniques on a per-data-set basis. Although the results for all learners and all performance metrics cannot be included, we include details for a few of the learner/metric combinations and summarize the results for the remaining combinations.

Table IV shows the results of evaluating C4.5D models using A-ROC. This table presents the average of the ten runs of tenfold cross-validation for each of the 15 data sets using each sampling technique. The HSD values in this table are meant to be read horizontally, rather than vertically, as was the case in the previous tables. That is, if two techniques have the same letter in a given row, then the performance of those techniques was not statistically different according to the HSD test with 95% confidence for the given data set (each row represents a data set). The final row of this table presents the average performance across all 15 data sets (as in Table III) and the number of data sets where the given technique was in group A (#A's), which is the best performing group.

RUSBoost is in group A for all 15 of the data sets in our experiments, which means that it did not perform significantly worse than any other technique in any of the 15 data sets (using C4.5D and A-ROC). SMOTEBoost was in group A for 11 of the 15 data sets. There were four data sets where RUSBoost significantly outperformed SMOTEBoost (Contraceptive, Mammography, SolarFlare, and SP3), and conversely, there were no data sets where SMOTEBoost (or any other sampling technique) significantly outperformed RUSBoost. AdaBoost performed nearly as well as SMOTEBoost, being in group A for ten of the 11 data sets. RUS was in group A for two data sets (Contraceptive and SolarFlare), while SMOTE was not in group A for any data set. None was in group A for the data set Contraceptive, indicating that no technique significantly improved the A-ROC of C4.5D for this data set. This moderately sized (1473 examples) data set has only slight imbalance (22.61% of the samples belong to the minority class).

Note that the number of A's only indicates the number of data sets where the given technique was in group A. It does not accurately reflect the relative performance of the techniques where neither technique was in group A. For example, SMOTEBoost is in group A 11 times, while AdaBoost is in group A ten times. That does not mean that SMOTEBoost only outperformed AdaBoost on one data set. To compare the performances of these techniques, one must look at the other letter assignments as well. In the case of SMOTEBoost and AdaBoost, SMOTEBoost significantly outperformed AdaBoost on two data sets (Mammography and SP3) even though SMOTEBoost was not in group A for either of these data sets. For the data set SP1, SMOTEBoost was in both groups A and B, while AdaBoost was only in group B. Even though SMOTEBoost was in group A for this data set, it did not significantly outperform AdaBoost, which was not in group A (they are both in group B). In other words, the "number of A's" only indicates how many times the given technique was not significantly outperformed by any other technique.

TABLE V
RESULTS FOR ALL DATA SETS USING RIPPER (A-PRC)

	RUSBoost		SMOTEBoost		AdaBoost		RUS		SMOTE		None	
Dataset	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD
CAR	0.8383	B	0.9952	A	0.7906	B	0.3229	C	0.7863	B	0.3425	C
CCCS12	0.7011	A	0.7428	A	0.6990	A	0.3271	C	0.5132	B	0.5020	B
CM1	0.3299	A	0.3315	A	0.3387	A	0.1765	B	0.2040	B	0.1188	C
CONTRACEPTIVE	0.4666	A	0.4567	A	0.4318	B	0.3462	C	0.3532	C	0.3164	D
ECOLI	0.7111	A	0.7438	A	0.7253	A	0.4457	C	0.5396	B	0.5144	BC
GLASS	0.4060	B	0.5758	A	0.4103	B	0.1484	C	0.3654	B	0.1274	C
MAMMOGRAPHY	0.7094	A	0.6982	A	0.6575	B	0.3462	E	0.5814	C	0.5085	D
PC1	0.4623	A	0.4580	A	0.4593	A	0.2142	C	0.3117	B	0.1865	C
PENDIGITS	0.9976	A	0.9977	A	0.9967	A	0.9099	D	0.9566	B	0.9445	C
SATIMG	0.7437	B	0.7688	A	0.7378	B	0.4605	D	0.4903	C	0.4453	D
SEGMENT	0.9835	A	0.9866	A	0.9815	A	0.7581	D	0.8599	B	0.8231	C
SOLARFLARE	0.2987	A	0.2686	A	0.2548	A	0.1672	B	0.1931	B	0.0962	C
SP1	0.2677	A	0.2316	B	0.2287	B	0.1521	C	0.1531	C	0.1086	D
SP3	0.1040	A	0.0915	AB	0.0697	CB	0.0422	ED	0.0577	CD	0.0232	E
VEHICLE	0.6598	AB	0.6724	A	0.6354	B	0.4508	D	0.5095	C	0.4364	D
Average / # A's	0.5787	12	0.6013	14	0.5612	7	0.3512	0	0.4583	0	0.3662	0

TABLE VI
NUMBER OF DATA SETS WHERE EACH TECHNIQUE WAS RANKED IN GROUP A

		RUSBoost	SMOTEBoost	AdaBoost	RUS	SMOTE	None
A-ROC	C4.5D	15	11	10	2	0	1
	C4.5N	15	14	4	3	7	3
	NB	14	15	4	12	11	12
	RIPPER	14	11	7	0	1	0
	Total	58	51	25	17	19	16
K-S	C4.5D	15	12	10	2	1	0
	C4.5N	14	13	5	2	6	2
	NB	15	15	3	13	12	13
	RIPPER	13	12	6	0	1	0
	Total	57	52	24	17	20	15
A-PRC	C4.5D	14	14	12	0	1	0
	C4.5N	15	14	8	0	3	1
	NB	14	14	3	10	11	11
	RIPPER	12	14	7	0	0	0
	Total	55	56	30	10	15	12
F-Measure	C4.5D	15	11	8	7	9	3
	C4.5N	15	12	7	4	8	3
	NB	12	13	10	9	10	11
	RIPPER	12	12	4	6	8	2
	Total	54	48	29	26	35	19
All	C4.5D	59	48	40	11	11	4
	C4.5N	59	53	24	9	24	9
	NB	55	57	20	44	44	47
	RIPPER	51	49	24	6	10	2
	Total	224	207	108	70	89	62

Table V shows similar details as Table IV using RIPPER to build models and A-PRC to evaluate them. This is the learner/metric combination where RUSBoost resulted in its worst performance. Its worst performance, however, is still very good. Table V shows that RUSBoost is in group A for 12 of the 15 data sets used in our study. There are only three data sets (Car, Glass, and SatImg) where RUSBoost is not in group A, which means that, for these three data sets, some other procedure (in these cases, SMOTEBoost) significantly outperformed RUSBoost. Instead, for each of these data sets, RUSBoost is in group B, which is the second best group. SMOTEBoost, on the other hand, is in group A for all but one of the experimental data sets. Using A-PRC to measure the performance of RIPPER models, SMOTEBoost has a slight edge over RUSBoost. AdaBoost is in group A for seven data sets, while SMOTE, RUS, and None are always outperformed by at least one of the other techniques (they are never in group A).

Note that, in most cases, RUS is the worst of the five techniques for handling class imbalance when A-PRC is used to measure the performance of RIPPER models. There are only four data sets where RUS performs significantly better than the baseline learner RIPPER, yet RUSBoost significantly outperforms None on all data sets, further demonstrating RUSBoost's ability to negate the drawbacks of RUS.

A detailed view of the results, as presented in Tables IV and V, cannot be included for all 16 combinations of learner and performance metric. However, a summary of these 16 combinations can be found in Table VI. This table indicates the number of data sets (out of 15) that the given sampling

technique was assigned group A for the given learner and performance metric. For example, using A-ROC to evaluate C4.5N models, RUSBoost was in group A for all 15 data sets, while using K–S to measure the performance of C4.5N models, RUSBoost was in group A for 14 of the 15 data sets. The rows labeled "Total" present the sums of these values for each performance metric (up to a maximum of 60). For example, using A-ROC to measure performance, RUSBoost was in group A for 58 of the 60 learner/data set combinations. At the bottom of Table VI (the section labeled "All"), the results for each learner/sampling technique combination (summed across the four performance measures) are presented. For example, using the C4.5D learner, RUSBoost is in group A for 59 of the 60 data set/metric combinations (15 using A-ROC + 15 using K–S + 15 using $F-measure$ + 14 using A-PRC).

The results show that the performance of RUSBoost is very favorable when compared to the other techniques, particularly using A-ROC or K–S to measure performance. Even using A-PRC and $F-measure$, its performance is very competitive, usually performing at least as well as SMOTEBoost. SMOTEBoost performs slightly better than RUSBoost using NB, but for the other three learners, RUSBoost is in group A more frequently than SMOTEBoost. Summing the number of times each sampling technique is assigned group A for all four learners using the A-ROC metric, RUSBoost is in group A for 58 of the 60 possible data set/learner combinations, while SMOTEBoost is in group A for 51 of these combinations. In other words, RUSBoost was significantly outperformed by at least one other technique, as measured by A-ROC, for two of the 60 scenarios. On the other hand, SMOTEBoost was outperformed by at least one other technique in nine of the 60 scenarios. A-PRC is the only performance metric where the per-data-set analysis favors SMOTEBoost. Of the 60 data set/learner combinations, SMOTEBoost beats RUSBoost by one data set (56 data sets to 55 data sets).

Overall, of the 240 combinations of learner, metric, and data set used in this study, RUSBoost is in group A for 224 (93.33%) of the combinations. SMOTEBoost, which is the next best technique, is in group A for 207 (86.25%) of these 240 combinations. Even using RIPPER, where RUSBoost performed worst (when evaluated using A-PRC), RUSBoost is in group A for more (51) experiments than SMOTEBoost (49). NB is the only learner where SMOTEBoost is in group A more frequently than RUSBoost, but examining the data in Tables IV–VI, in a relative sense, there is not as much of a difference in the performances between all five techniques using NB.

D. Comparing RUSBoost and SMOTEBoost

Finally, we directly compare RUSBoost and SMOTEBoost in Table VII. The previous sections have used the HSD statistical test to perform multiple pairwise comparisons between the mean performances of the different techniques. The HSD test has the advantage that adjustments are made to reduce the likelihood of obtaining a type I error (i.e., incorrectly rejecting the null hypothesis of equality between two means) of the entire experiment. An alternative is to perform multiple pairwise t-tests; however, as the number of comparisons increases, the likelihood of obtaining a type I error of the entire experiment increases. Since it is more conservative, HSD is often preferred for multiple mean comparisons; however, it has the drawback that it has a higher likelihood to make a type II error, i.e., failing to correctly reject the null hypothesis of equality between two means. In this section, we analyze only RUSBoost and SMOTEBoost—therefore, we revert to computing a standard t-statistic to compare the means of these two techniques to obtain a more precise comparison.

TABLE VII
t-TEST COMPARISON OF RUSBOOST AND SMOTEBOOST

	A-ROC	K-S	F-Measure	A-PRC	Total
RUSBoost	17	16	17	8	58
SMOTEBoost	9	9	6	7	31
Neither	34	35	37	45	151

Table VII compares only RUSBoost and SMOTEBoost, with a two-sample t-statistic calculated for each learner and data set, presented by a performance metric. Therefore, each column totals to 60, and in total, 240 pairwise comparisons between RUSBoost and SMOTEBoost were performed, each with a 95% confidence level. The first row represents the number of times that RUSBoost significantly outperformed SMOTEBoost, the second row is the number of times that SMOTEBoost significantly outperformed RUSBoost, and the final row represents the cases with no significant difference between SMOTEBoost and RUSBoost. Overall, RUSBoost is preferred over SMOTEBoost, particularly relative to the A-ROC, K–S, and $F-measure$ performance metrics.

VI. CONCLUSION

Many domains are affected by the problem of class imbalance. That is, when examples of one class greatly outnumber examples of the other class(es), it can be challenging to construct a classifier that effectively identifies the examples of the underrepresented class. Several techniques have been proposed for dealing with the problem of class imbalance. In this paper, we have proposed a new algorithm, called RUSBoost, for alleviating the problem of class imbalance. We compare the performance of RUSBoost with that of several of the most popular and most effective techniques for learning from skewed data, finding that RUSBoost performs favorably when compared to these other techniques.

RUSBoost presents a simpler, faster, and less complex alternative to SMOTEBoost for learning from imbalanced data. SMOTEBoost combines a popular oversampling technique (SMOTE) with AdaBoost, resulting in a hybrid technique that surpasses the performance of its components. SMOTEBoost, however, has two main drawbacks: complexity and increased model training time. By using RUS instead of SMOTE, RUSBoost achieves results that are comparable to (and often better than) SMOTEBoost using an algorithm that is simpler to implement and that results in faster model training times.

This paper compares the performances of RUSBoost and SMOTEBoost, as well as their individual components: RUS, SMOTE, and AdaBoost. Models were constructed using four different base learners, including two versions of C4.5 decision

trees, RIPPER and NB. Experiments were conducted using 15 data sets of various sizes with various levels of imbalance from many different application domains. In addition, as there is no universally agreed upon performance metric for measuring classification performance when data are imbalanced, we evaluate all models using four different performance metrics: ROC curves, PRC curves, K–S statistic, and $F-measure$.

An interesting result presented in this paper relates to the use of different performance measures and the performances of RUSBoost, SMOTEBoost, SMOTE, and RUS using these different metrics. Using a performance metric such as A-ROC, RUS significantly improved the classification performance, despite its relative simplicity and its inherent drawback (loss of information). Conversely, applying RUS can result in worse classification performance using A-PRC. If the objective of the model construction is to maximize A-PRC, it is often significantly better to build models without using sampling than to apply RUS. On the contrary, SMOTE results in an improved performance regardless of performance metric selection. Therefore, one might expect SMOTEBoost (which uses SMOTE) to result in better performance than RUSBoost (which uses RUS) when evaluating models using A-PRC. However, we find that there is no significant difference between the performances of RUSBoost and SMOTEBoost using this performance measure (as shown in Table III). SMOTEBoost does outperform RUSBoost using one learner (RIPPER) when A-PRC is used to measure performance; however, as shown in Table VI, the performances of RUSBoost and SMOTEBoost are very similar, even when using this metric. In general, as shown in Table VI, RUSBoost obtained the best performance (of the five techniques used in our study) more often than any of the other techniques investigated in this paper.

This paper has presented a comprehensive empirical investigation comparing the performances of several techniques for improving classification performance when data are imbalanced, including our proposed RUSBoost algorithm. Considering 240 different combinations of learner, performance metric, and data set, we find that our proposed RUSBoost algorithm most frequently achieves the best performance. Although the overall performances of RUSBoost and SMOTEBoost are not significantly different (when averaged across all four learners and 15 data sets), RUSBoost significantly outperforms SMOTEBoost about twice as often as SMOTEBoost outperforms RUSBoost. This superior performance is achieved while reducing the model training times. It is this combination of simplicity, speed, and excellent performance that makes RUSBoost an attractive algorithm for learning from skewed training data.

Future work will continue to investigate the performance of RUSBoost. RUSBoost will be evaluated using additional learners, and its effectiveness in application domains such as bioinformatics and life sciences will be assessed. In addition, the effects of parameter selection on RUSBoost (the number of boosting iterations and the level of sampling) will be evaluated in an attempt to identify potentially favorable parameter values. Finally, the performance of RUSBoost will be compared to that of additional techniques for dealing with imbalanced data, including, but not limited to, additional boosting algorithms.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the Associate Editor for the constructive evaluation of this paper and also the various members of the Data Mining and Machine Learning Laboratory, Florida Atlantic University, for the assistance with the reviews.

REFERENCES

[1] G. M. Weiss, "Mining with rarity: A unifying framework," *SIGKDD Explor.*, vol. 6, no. 1, pp. 7–19, Jun. 2004.
[2] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor.*, vol. 6, no. 1, pp. 20–29, Jun. 2004.
[3] C. Drummond and R. C. Holte, "C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling," in *Proc. Int. Conf. Mach. Learn. Workshop Learn. From Imbalanced Data Sets II*, 2003, pp. 1–8.
[4] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156.
[5] Y. Freund and R. Schapire, "A short introduction to boosting," *J. Jpn. Soc. Artif. Intell.*, vol. 14, no. 5, pp. 771–780, Sep. 1999.
[6] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proc. Principles Knowl. Discov. Databases*, 2003, pp. 107–119.
[7] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "SMOTE: Synthetic minority oversampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.
[8] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, Jun. 2007, pp. 935–942.
[9] N. Japkowicz, "Learning from imbalanced data sets: A comparison of various strategies," in *Proc. AAAI Workshop Learn. From Imbalanced Data Sets*, 2000, pp. 10–15.
[10] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *J. Artif. Intell. Res.*, vol. 19, pp. 315–354, 2003.
[11] T. M. Khoshgoftaar, C. Seiffert, J. Van Hulse, A. Napolitano, and A. Folleco, "Learning with limited minority class data," in *Proc. 6th ICMLA*, 2007, pp. 348–353.
[12] R. Barandela, R. M. Valdovinos, J. S. Sanchez, and F. J. Ferri, "The imbalanced training sample problem: Under or over sampling?" in *Proc. Joint IAPR Int. Workshops SSPR/SPR*," vol. 3138, *Lecture Notes in Computer Science*, 2004, pp. 806–814.
[13] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. ICIC*, vol. 3644, *Lecture Notes in Computer Science*, New York, 2005, pp. 878–887.
[14] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Building useful models from imbalanced data with sampling and boosting," in *Proc. 21st Int. FLAIRS Conf.*, May 2008, pp. 306–311.
[15] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: Misclassification cost-sensitive boosting," in *Proc. 16th Int. Conf. Mach. Learn.*, 1999, pp. 97–105.
[16] K. M. Ting, "A comparative study of cost-sensitive boosting algorithms," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 983–990.
[17] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, Dec. 2007.
[18] M. V. Joshi, V. Kumar, and R. C. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2001, pp. 257–264.
[19] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: The DataBoost-IM approach," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 30–39, Jun. 2004.
[20] D. Mease, A. J. Wyner, and A. Buja, "Boosted classification trees and class probability/quantile estimation," *J. Mach. Learn. Res.*, vol. 8, pp. 409–439, May 2007.
[21] G. Weiss, K. McCarthy, and B.Zabar, "Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs?" in *Proc. Int. Conf. Data Mining*, 2007, pp. 35–41.

[22] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi, "Automatically countering imbalance and its empirical relationship to cost," *Data Mining Knowl. Discov.*, vol. 17, no. 2, pp. 225–252, Oct. 2008.
[23] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. 17th Int. Conf. Mach. Learn.*, 2001, pp. 239–246.
[24] NASA/WVU IV&V Facility, Metrics data program. [Online]. Available: http://mdp.ivv.nasa.gov
[25] C. Blake and C. Merz, UCI Repository of Machine Learning Databases, Irvine, CA: Dept. Inform. Comput. Sci., Univ. California1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/~MLRepository.html
[26] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*., 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.
[27] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[28] J. Aczel and J. Daroczy, *On Measures of Information and Their Characterizations*. New York: Academic, 1975.
[29] W. W. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 115–123.
[30] J. Furnkranz and G. Widmer, "Incremental reduced error pruning," in *Proc. Int. Conf. Mach. Learn.*, 1994, pp. 70–77.
[31] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[32] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Mach. Learn.*, vol. 42, no. 3, pp. 203–231, Mar. 2001.
[33] T. Landgrebe, P. Paclik, and R. Duin, "Precision–recall operating characteristic (P-ROC) curves in imprecise environments," in *Proc. 18th Int. Conf. Pattern Recog.*, 2006, vol. 4, pp. 123–127.
[34] J. Davis and M. Goadrich, "The relationship between precision–recall and ROC curves," in *Proc. 23rd ICML*, 2006, pp. 233–240.
[35] D. J. Hand, "Good practice in retail credit scorecard assessment," *J. Oper. Res. Soc.*, vol. 56, no. 9, pp. 1109–1117, 2005.
[36] M. L. Berenson, D. M. Levine, and M. Goldstein, Intermediate Statistical Methods and Applications: A Computer Package Approach, Englewood Cliffs, NJ: Prentice-Hall1983.
[37] SAS Inst. Inc., SAS/STAT User's Guide, Cary, NC, 2004.

Chris Seiffert received the B.S. degree in computer and information science from the University of Florida, Gainesville, in 2002 and the M.S. and Ph.D. degrees in computer engineering from the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, in 2005 and 2008, respectively.

His research interests included data mining, machine learning, and knowledge discovery with imbalanced data. He passed away unexpectedly in October 2008.

Taghi M. Khoshgoftaar (M'86) is currently a Professor with the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, where he is the Director of the Empirical Software Engineering Laboratory and the Data Mining and Machine Learning Laboratory. He has served as a North American Editor of the *Software Quality Journal*, was on the editorial board of the *Empirical Software Engineering Journal*, and is on the editorial boards of the journals *Software Quality*, *Fuzzy Systems*, and *Knowledge and Information Systems*. His research interests are in software engineering, software metrics, software reliability and quality engineering, computational intelligence, computer performance evaluation, data mining, machine learning, and statistical modeling. He has published more than 350 refereed papers in these areas.

Prof. Khoshgoftaar is a member of the IEEE Computer Society and the IEEE Reliability Society. He was the Program Chair and the General Chair of the IEEE International Conference on Tools with Artificial Intelligence in 2004 and 2005, respectively, and was the Program Chair of the 20th International Conference on Software Engineering and Knowledge Engineering in 2008. He is the General Chair of the 21st International Conference on Software Engineering and Knowledge Engineering (in 2009). He has served on technical program committees of various international conferences, symposia, and workshops.

Jason Van Hulse (M'05) received the B.S. degree in mathematics from the University at Albany, Albany, NY, in 1997, the M.A. degree in mathematics from Stony Brook University, Stony Brook, NY, in 2000, and the Ph.D. degree in computer engineering from Florida Atlantic University, Boca Raton, in 2007.

Since 2000, he has been with First Data Corporation, Greenwood Village, CO, working in the data mining and predictive modeling fields, where he is currently the Vice President of Decision Science. He is also a Research Assistant Professor of Computer Science and Engineering at Florida Atlantic University. His research interests include data mining and knowledge discovery, machine learning, computational intelligence, and statistics. He has published numerous peer-reviewed research papers in various conferences and journals.

Dr. Van Hulse is a member of the IEEE Computer Society and the Association for Computing Machinery.

Amri Napolitano received the B.S. degree in computer and information science from the University of Florida, Gainesville, in 2004 and the M.S. degree in computer science from Florida Atlantic University, Boca Raton, in 2006.

He is currently with the Department of Computer Science and Engineering, Florida Atlantic University. His research interests include data mining and machine learning, evolutionary computation, and artificial intelligence.