Comparing Boosting and Bagging Techniques With Noisy and Imbalanced Data

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

***Abstract*—This paper compares the performance of several boosting and bagging techniques in the context of learning from imbalanced and noisy binary-class data. Noise and class imbalance are two well-established data characteristics encountered in a wide range of data mining and machine learning initiatives. The learning algorithms studied in this paper, which include SMOTEBoost, RUSBoost, Exactly Balanced Bagging, and Roughly Balanced Bagging, combine boosting or bagging with data sampling to make them more effective when data are imbalanced. These techniques are evaluated in a comprehensive suite of experiments, for which nearly four million classification models were trained. All classifiers are assessed using seven different performance metrics, providing a complete perspective on the performance of these techniques, and results are tested for statistical significance via analysis-of-variance modeling. The experiments show that the bagging techniques generally outperform boosting, and hence in noisy data environments, bagging is the preferred method for handling class imbalance.**

***Index Terms*—Bagging, binary classification, boosting, class imbalance, class noise.**

I. Introduction

TWO COMMON aspects of data quality that can affect classification performance are class imbalance and noise. Class imbalance occurs when examples of one of the classes are severely outnumbered by those of the other class(es).[1] When data are imbalanced, traditional data mining algorithms tend to favor the overrepresented (majority or negative) class, resulting in high overall correct classification rates but unacceptably low detection rates with respect to the underrepresented (minority or positive) class. For example, when a model is trained on a binary data set with 1% of its examples from the minority class, a 99% accuracy rate can be achieved by classifying all examples as belonging to the majority class. While 99% accuracy is often considered excellent, such a model often has no practical value. The minority class is typically the class of interest, often carrying a much higher cost of misclassification than the majority class.

Data are said to be noisy if they contain erroneous data values. These erroneous values can occur in the independent (attribute noise) or dependent (class noise) variables in a data set. While real-world data often contain both types of noise, the latter one is generally more detrimental to classification performance [1]. Noisy data can confuse a learning algorithm, blurring the decision boundaries that separate the classes or causing models to overfit to accommodate incorrect data points. The impact of class noise on classification performance is determined by two factors: the level of noise and the distribution of noise in the data. The level of noise refers to the amount of noise in the data. Noise distribution (ND) refers to the type of noise in the data (the proportion of mislabeled examples that belong to the majority class and minority class). Research has shown that mislabeled minority class examples (i.e., the true class is positive, but the examples are incorrectly labeled as negative) have a greater impact on classification performance than mislabeled majority class examples [2]. While many studies have investigated class imbalance and class noise in isolation, very few have addressed their combined effects.

Several techniques have been proposed to deal with the problem of class imbalance. Data sampling [3], [4], a common method for dealing with class imbalance, balances the class distribution (CD) of the training data by either adding examples to the minority class (oversampling) or removing examples from the majority class (undersampling). Two common data sampling techniques that are relevant to this study are random undersampling (RUS) and the Synthetic Minority Oversampling Technique (SMOTE) [5]. Boosting [6] is another technique which is very effective when learning from imbalanced data [7]. Bagging, which is thought to be less effective than boosting when dealing with class imbalance [8], may outperform boosting when data contain noise [9] because boosting may attempt to build models to correctly classify noisy examples. In this paper, we investigate several hybrids of these techniques. SMOTEBoost [10] and RUSBoost [11] combine data sampling and boosting and significantly outperform both data sampling and boosting when applied individually [12]. Exactly Balanced Bagging (EBBag) and Roughly Balanced Bagging (RBBag) [13] combine undersampling with bagging and are also effective when data are imbalanced.

A. Research Objectives

The objective of this work is to present a comprehensive empirical evaluation and comparison of the boosting and bagging techniques SMOTEBoost, RUSBoost, EBBag, and RBBag in the context of learning from imbalanced and noisy data. Further explanation of the rationale used to select these techniques is provided in Section II. Both noise and class imbalance are important components of our experiments, as we seek to understand the relative performances of these techniques as

Manuscript received January 23, 2009; revised December 24, 2009; accepted May 24, 2010. Date of publication November 28, 2010; date of current version April 15, 2011. This paper was recommended by Associate Editor H. Pham.

The authors are with 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.2010.2084081

[1] Our experiments only consider binary classification; hence, for the remainder of this paper, we only refer to two-class learning problems. Exploration of the multiclass setting is left as future work.

1083-4427/$26.00 © 2010 IEEE

both noise and class imbalance levels are varied. This objective is achieved by using simulated data derived from four real-world data sets (see Section III) and controlling the level of class imbalance and the distribution and level of class noise. Nearly four million models were trained and evaluated within our experiments, and to obtain a more complete perspective than prior studies, seven different performance metrics were used to evaluate model efficacy.

B. Paper Organization

The remainder of this paper is organized as follows. Section II presents related work and provides additional background regarding our research objectives. The experimental design is explained in Section III, and empirical results are provided in Section IV. Conclusions are presented in Section V.

II. Background and Related Work

Several studies have investigated the problem of class imbalance. Weiss [4], for example, provides an overview of class imbalance and describes several techniques for overcoming the problem. Japkowicz and Stephen [14] also investigate class imbalance, identifying particular aspects of imbalance that are most problematic for learning and comparing a few techniques for dealing with the problem.

Data sampling is a common technique for handling class imbalance. The goal of data sampling is to create a more balanced data set on which to train models. This can be done by either oversampling (adds new examples to the minority class) or undersampling (removes examples from the majority class). Both oversampling and undersampling can be performed randomly or intelligently. Random sampling involves removing random examples from the majority class (RUS) or duplicating random examples of the minority class (random oversampling). Other intelligent data sampling techniques have also been proposed. SMOTE [5], for example, creates new minority class examples by extrapolating between existing examples rather than simply duplicating them. Borderline-SMOTE [15] attempts to improve upon SMOTE by creating new examples based only on examples near the decision border. Wilson's Editing [16], [17] is an intelligent undersampling technique that tries to remove only noisy majority class examples. The various sampling techniques have been evaluated and compared in great detail [3], [18], [19].

In addition to data sampling, various algorithms derived from boosting methods have been proposed for handling class imbalance (or cost-sensitive problems), including AdaCost [20], CSB0, CSB1, and CSB2 [21], AdaC1, AdaC2, and AdaC3 [22], RareBoost [23], SMOTEBoost [10], and RUSBoost [11]. Our recent research has shown that, among these techniques, RUSBoost and SMOTEBoost perform the best when data are imbalanced [12], and hence, we consider only SMOTEBoost and RUSBoost in this work.

Boosting in the context of noisy data has been considered in previous work [24]–[26]. Karmaker and Kwek [25] present an alteration of AdaBoost called ORBoost which filters examples with weights that grow above a user-determined threshold. AveBoost2, another modification of AdaBoost to deal with label noise, is developed by Oza [26]. We restrict our evaluation to techniques specifically designed to deal with imbalanced data, and hence omit these boosting techniques. It was not the goal to evaluate all noise-handling techniques but to instead focus on the performance of boosting and bagging techniques proposed in the literature to deal with imbalanced data (which ORBoost and AveBoost2 are not) when the underlying training data are noisy. Note that none of the techniques, i.e., SMOTEBoost, RUSBoost, EBBag, and RBBag, are designed specifically to handle noise. A comprehensive comparison of all robust classification techniques is outside the scope of this work and is appropriate for future exploration.

Bagging, like boosting, is a metalearning technique that constructs an ensemble of models in order to improve classification performance. In this study, we investigate two versions of bagging which have been proposed to alleviate the problems associated with class imbalance. EBBag combines the entire minority class with random subsets of the majority class in such a manner that the CD is balanced when building each model in the ensemble [27]–[30]. RBBag [13] is very similar to EBBag but attempts to adhere more closely to the theory of bagging by allowing for variations in training data set CDs. Although simple bagging may not be expected to perform particularly well when dealing with class imbalance [8], our results show that the modified bagging techniques perform very well in this regard. RBBag and EBBag were selected for consideration in this work based on preliminary experimentation, showing that these techniques perform comparably with RUSBoost and SMOTEBoost when learning from imbalanced data.

Given this background of relevant prior work, we restrict our analysis to the four most promising metalearning techniques that have been proposed for learning from imbalanced data: SMOTEBoost, RUSBoost, EBBag, and RBBag. To our knowledge, this is the first comprehensive empirical comparison of the performance of these techniques. We do not consider any of the individual sampling techniques in this work for a few reasons. First, we investigate hybrids of two of the stronger data sampling techniques (RUS and SMOTE) with boosting, and our recent work has shown that simple boosting outperforms data sampling [31], while RUSBoost and SMOTEBoost outperform both individual boosting and sampling [7], [11]. In addition, our previous research [32] has already evaluated data sampling in the context of noisy and imbalanced data.

Another characteristic of training data which can cause problems for learning algorithms is data noise. Noise can be categorized as either attribute noise or class noise. Several techniques have been proposed for dealing with class noise, including robust classifiers [33], noise filtering [34], [35], and noise correction [36], [37]. A classifier is said to be robust to noise if its performance does not deteriorate significantly in the presence of noise. Noise filters attempt to identify and remove noisy examples from the training data, while noise correctors attempt to correct noisy data values. This study does not focus on noise handling but instead identifies which techniques for handling class imbalance (of RUSBoost, SMOTEBoost, EBBag, and RBBag) are most robust to class noise in the training data.

While many studies have investigated the problems of class imbalance and noise in isolation, very few have studied their combined effects. Van Hulse *et al.* [2] present an analysis of the impact of class noise in imbalanced data. Specifically, our previous work investigates the impact of the level and distribution of class noise on 11 different learners. We extend our previous work in a few important ways. The baseline data sets used in this work were acquired from the University of California, Irvine (UCI) repository, while our previous work used data sets from the software quality classification domain that were preprocessed to remove noise. Second, the experiments in this work are expanded to include variations of the CD, and third, the effects of these three factors (CD, noise level (NL), and ND) are analyzed using several boosting and bagging techniques designed specifically for learning from imbalanced data. Anyfantis *et al.* [38] investigate a few cost-sensitive learning techniques in a noisy and imbalanced environment. Unlike our study, their work considers only two levels of noise and assumes a uniform distribution of noise in the data sets. Furthermore, unlike their work, we vary the level of class imbalance in the data. Our work is also different in that we investigate boosting and bagging techniques designed for imbalanced data, while they investigate cost-sensitive learning techniques.

Why is it important to consider both class noise and class imbalance? We contend that both of these issues are pervasive in real-world data mining applications, so understanding the performance of learning techniques in such an environment is critical to both researchers and practitioners. Based on the results of these experiments, future research can be directed toward improving the noise handling of techniques which perform poorly when data are noisy. One theory proposed in related research [4], [14] is that class imbalance by itself is not the primary cause of poor classifier performance, but instead, other issues such as small disjuncts, overlapping classes, concept complexity, or absolute rarity are much more important. Class noise can be a cause of all of these issues, for example, noisy data generally increase concept complexity and can create small disjuncts in the data. Therefore, conducting experiments which explicitly consider class noise provides valuable insight into the general problem of class imbalance.

III. Experiments

A. Data Sets

The experiments in this work are based on four data sets from the UCI repository [39]. Each of the data sets originally contained multiple class values, which were transformed into binary-class problems as described in the following. The data sets used in these experiments, as well as information about their size and CDs, can be found in Table I. The row labeled "Size" indicates the total number of examples available in the given data set. The rows "#min" and "%min" identify the number of minority class examples and the percentage of all examples belonging to the minority class, respectively. We selected these four data sets due to both the total number of examples and the number of minority class examples. These characteristics facilitated our experimental design, as described in Section III-E. The most important, however, is that these data sets are relatively clean. That is, models constructed on these data sets resulted in nearly perfect classification performance prior to noise injection.

TABLE I
Characteristics of the Four Initial UCI Data Sets Prior to Noise Simulation

	LetterA	Nursery3	OptDigits8	Splice2
Size	20000	12960	5620	3190
#min	789	328	554	768
%min	3.95	2.53	9.86	24.08

The LetterA data set is based on a letter recognition database originally containing 26 different classes (one for each letter). The class attribute was transformed into a binary class by selecting letter "A" to be the positive class and the remaining letters to be the negative class. The Nursery3 data set is based on a data set used to rank nursery schools. The data set originally contained five classes, one of which ("very recommended") was selected to be the positive class, while the remaining class values were grouped together to make up the negative class. The OptDigits8 data set is used to train models to recognize handwritten digits from zero to nine. Digit "8" was selected as the positive class, and the remaining digits make up the negative class. The Splice2 data set is a molecular biology data set used to identify different "splice junctions" on a deoxyribonucleic acid sequence. Class "ie" (known as "acceptors") was selected as the positive class, while classes "ei" ("donors") and "Neither" were combined to form the negative class.

B. Boosting and Bagging Techniques

Two different implementations of each of the four boosting and bagging techniques SMOTEBoost, RUSBoost, EBBag, and RBBag are considered in this work, for a total of eight procedures. Both of the bagging techniques EBBag and RBBag are implemented by performing bagging with replacement (EBBagR and RBBagR) and without replacement (EBBagN and RBBagN). The boosting techniques SMOTEBoost and RUSBoost also have two different implementations. SMOTEBoostW and RUSBoostW use "boosting by reweighting," while SMOTEBoostS and RUSBoostS use "boosting by resampling." Each of these techniques was implemented by our research group within the Weka [40] environment, an open-source data mining suite. Space considerations preclude a full description of these algorithms within this work; however, further details can be obtained by the interested reader by consulting the included references.

Throughout this paper, the term "technique" will be used to categorize all eight of these bagging and boosting techniques. The term "version" will distinguish between EBBag and RBBag or SMOTEBoost and RUSBoost. The term "implementation" will distinguish between bagging with replacement and without replacement or boosting by resampling and boosting by reweighting. Both boosting and bagging were performed using $n = 10$ iterations. Preliminary testing with more iterations did not yield significantly different results.

C. Learners

The four learners used in this study are all provided by Weka [40]. The first two learners, denoted C4.5D and C4.5N, are based on the standard C4.5 decision tree algorithm called J48 in Weka. C4.5D uses the default parameter settings in Weka, while C4.5N disables pruning and enables Laplace smoothing [41]. The third learner is *Naive Bayes* (NB), which uses the default Weka parameters. *RIPPER* is a rule-based learner implemented as JRip in Weka. Our experiments use the default Weka parameters for RIPPER.

D. Model Evaluation

Model performance is measured using seven different commonly used learning metrics [42]: area under the receiver operating characteristic curve (AROC), area under the precision–recall curve (APRC), Kolmogorov–Smirnov statistic (KS), F-measure (FM), geometric mean (GM), best FM (BFM), and best GM (BGM).

E. Experimental Design

While the four data sets described in Section III-A are the basis for our experiments, our experiments are performed using subsets of those data sets with controlled CDs, NLs, and NDs. Experiments are performed using ten-fold cross validation, and for each fold of cross validation, a new subset of the original data is sampled from the training folds with 1500 examples and a CD according to our experimental parameters. These training subsets are then corrupted to include the desired level and distribution of class noise. CD, NL, and ND are described in this section.

1) Experimental Factor—CD: The first experimental factor CD indicates the percentage of examples in the derived data set belonging to the minority class. The experiments in this work consider six levels of CD, i.e., $CD = \{1, 2, 4, 6, 8, 10\}$, where the value of CD indicates the percentage of examples in the training data that belong to the minority class. For example, a derived data set with $CD = 4$ has 4% of its examples from the minority class and 96% of its examples from the majority class. Note that this is the ratio prior to noise injection (discussed in the following sections). Depending on the ND, the final data set may contain a different CD.

2) Experimental Factor—NL: The second factor NL determines the quantity of noisy examples in the training data. The selected data sets are relatively clean, so NL is varied by artificially injecting noise into the data set. This is accomplished by swapping the class value of some of the examples. The number of examples with their classes swapped is a function of NL and the number of minority examples in the derived data set.

While many works involving noise injection often inject noise by simply selecting $x\%$ of the examples and corrupting their class, this technique may be inappropriate when dealing with imbalanced data sets. For example, if a data set contains only 1% of its examples belonging to the minority class and as little as 10% of its examples are corrupted (injected with noise), the minority class will become overwhelmed by noisy examples from the majority class. Instead, we corrupt a percentage of the examples based on the number of minority examples in the data set.

In our experiments, we use five levels of noise $NL = \{10, 20, 30, 40, 50\}$, where NL determines the number of examples, based on the size of the minority class, that will be injected with noise. The actual number of noisy examples will be

$$2 \times \frac{NL}{100} \times P$$

where P is the number of positive (minority) examples in the data set. For example, a data set with $CD = 10$ and $NL = 20$ will have 150 minority examples (10% of 1500) and $2 \times 0.2 \times 150 = 60$ noisy examples. Note that this does not indicate which examples (minority or majority) will be corrupted. That is determined by the final experimental factor: ND.

3) Experimental Factor—ND: The final experimental factor ND determines the type of noise that will be injected into the data. When dealing with binary-class data sets (the only kind considered in this work), there are two possible noise types: P → N and N → P. Noise of type N → P occurs when an example is mistakenly labeled as "positive" while the true label is "negative." Conversely, P → N noise occurs when an example is mistakenly labeled as "negative" while the true label is "positive." In other words, the left-hand side of the arrow represents the true class of the instance, and the arrow signifies the corruption of the class by some mechanism to a noisy (incorrect) value on the right-hand side. When constructing classification models, the learners are only "aware" of the noisy or incorrect class value for those instances that are injected with noise.

The experiments in this work use five levels of ND, i.e., $ND = \{0, 25, 50, 75, 100\}$, where the value of ND indicates the percentage of noisy examples that are of type P → N. For example, if a derived data set is to contain 60 noisy examples and $ND = 25$, then 25% (15) of those noisy examples will be minority ("positive") class examples that have their labels changed to "negative" $(\mathrm{P} \rightarrow \mathrm{N})$, while the remaining 75% (45) of the noisy examples will be of type N → P. Due to the definition of ND, the combination of $ND = 100$ and $NL = 50$ cannot be used since the resulting data set would have zero minority examples.

F. Parameter Selection

Each of the bagging and boosting algorithms implemented in this paper attempts to alleviate the problem of class imbalance within the training data. This is accomplished by undersampling (both versions of bagging and RUSBoost) or oversampling (SMOTEBoost). These techniques, therefore, require a user-specified parameter that indicates how much sampling should be done. In our implementation, this parameter specifies the percentage of examples in the postsampling data set that will belong to the minority class.

Two values of this parameter were used for our experiments: 35% and 50%. That is, models were trained to achieve CDs (minority:majority) of approximately 1:2 $(param = 35)$ and 1:1 $(param = 50)$. In some previous work, sampling is performed

TABLE II
COMPARISON OF PARAMETER SETTINGS FOR THE BOOSTING AND BAGGING TECHNIQUES. EACH PERFORMANCE METRIC IS AVERAGED OVER ALL DATA SETS, LEARNERS, CDs, NDs, AND NLs

Technique	$param$	AROC	APRC	KS	FM	GM	BFM	BGM
EBBagN	35	**.945**	**.725**	**.792**	**.563**	**88.07**	**.709**	**89.55**
	50	.943	.713	.789	.448	85.85	.703	89.40
EBBagR	35	**.942**	**.721**	**.783**	**.585**	85.67	**.697**	**89.09**
	50	.940	.713	.779	.496	**87.17**	.695	88.91
RBBagN	35	**.945**	**.726**	**.792**	**.562**	**88.08**	**.709**	**89.55**
	50	.944	.714	.791	.446	85.83	.704	89.49
RBBagR	35	**.942**	**.720**	**.782**	**.583**	85.81	**.700**	**89.05**
	50	.940	.711	.779	.489	**87.02**	.693	88.91
RUSBoostS	35	**.927**	**.677**	**.740**	**.574**	76.86	**.654**	**86.94**
	50	.915	.635	.732	.449	**84.51**	.637	86.51
RUSBoostW	35	**.937**	**.695**	**.767**	**.572**	**85.76**	**.678**	**88.30**
	50	.922	.646	.751	.425	84.43	.654	87.45
SMOTEBoostS	35	.921	.683	.734	**.508**	64.71	.659	86.59
	50	**.925**	**.698**	**.741**	.508	**67.70**	**.670**	**86.96**
SMOTEBoostW	35	**.924**	**.673**	**.743**	**.531**	68.22	**.656**	**87.06**
	50	.920	.667	.736	.517	**70.21**	.650	86.66

to achieve a balanced 1:1 CD [41]. However, our previous studies have shown that the 1:2 ratio may be better, particularly when data are severely imbalanced [43].

Table II presents the overall results (for all of the experiments described in this section) using each of the eight boosting/bagging techniques and each of the seven performance metrics. The results, averaged across all other experimental parameters, using $param = 35$ and $param = 50$ are presented. The values in **bold** print indicate which of the parameters resulted in better performance for the given combination of metric and technique.

For 46 of the 56 technique/metric combinations, $param = 35$ outperforms $param = 50$. Seven of the eight techniques obtain better performance using $param = 35$ for at least six of the seven metrics. The only technique that frequently performed better using $param = 50$ was SMOTEBoostS. In general, however, the difference between $param = 50$ and $param = 35$ for this technique was relatively small. For four of the seven metrics, the difference was less than 1%, and only one metric (GM) had a difference of more than 2%. Since the majority of the techniques yielded better results using $param = 35$ regardless of the performance metric, all results presented in this study are based on $param = 35$.

G. Design Summary

All experiments are performed using tenfold cross validation. That is, the original data sets (with the original size and CD as described in Section III-A) are divided into ten equal-sized partitions. Nine of those partitions will be used to create a training data set, while the remaining partition will be used as test data and remains unsampled and uncorrupted. During each fold, examples are sampled from the nine training partitions to create training subsets $T(CD)$ with 1500 examples and each of the desired CDs. For example, $T(1)$ contains 1485 majority class examples and 15 (1%) minority class examples, while $T(2)$ contains 1470 majority class examples and 30 (2%) minority class examples. Furthermore, the 30 minority class examples in $T(2)$ contain all 15 of the examples from $T(1)$ plus 15 additional minority examples. Similarly, $T(1)$ contains all 1470 of the majority class examples from $T(2)$ plus 15 additional majority examples. Each of these training subsets is then injected with class noise to create 24 new data sets [from each $T(CD)$] $T_{ND}^{NL}(CD)$, with the desired levels of class noise based on the factors NL and ND. As with CD, increased levels of NL and ND corrupt the same examples as the lesser levels, plus additional examples. This entire process is repeated so that each of the ten partitions of the original data set will be used as test data once.

In total, during a single run of tenfold cross validation, 6 CDs and 24 combinations of NL and ND are used, creating $6 \times 24 \times 10 = 1440$ training subsets from each of the four original training data sets (resulting in $4 \times 1440 = 5760$ in all). Furthermore, we perform ten independent runs of tenfold cross validation to alleviate any biasing that may occur during the random partitioning and noise injection processes. Therefore, the total number of training subsets created for our experiments is $10 \times 5760 = 57\,600$.

Each of the 57 600 training subsets is used to train four learners (C4.5D, C4.5N, NB, and RIPPER) using eight boosting and bagging techniques, each of which is performed using two parameter values (described in Section III-F). Experiments are also performed without boosting or bagging, denoted NONE. Therefore, a total of $4 \times (8 \times 2 + 1) = 68$ models are training on each of the 57 600 training subsets. The total number of models trained for this set of experiments is $57\,600 \times 68 = 3\,916\,800$. Each of these models is evaluated using seven different performance metrics.

IV. EMPIRICAL RESULTS

This section presents the results of our extensive suite of experiments. We begin by analyzing the main factors in our experiments using an analysis-of-variance (ANOVA) model, namely, the boosting or bagging technique (Tech), the base learner (Lrn), the level of imbalance (CD), the level of noise (NL), and the distribution of class noise within the data (ND). We examine the impact of Tech and Lrn on classification performance, in general, in Section IV-A, before focusing on the boosting and bagging techniques in Section IV-B.

TABLE III
F-STATISTICS AND p-VALUES FOR THE FIVE MAIN FACTORS AND ALL TWO- AND THREE-WAY INTERACTIONS FOR ALL SEVEN PERFORMANCE METRICS

		AROC		APRC		KS		FM		GM		BFM		BGM	
Factor	DF	F-stat	p	F-stat	p	F-stat	p	F-stat	p	F-stat	p	F-stat	p	F-stat	p
Lrn	3	14987.6	0	4032.71	0	10847.4	0	531.38	0	5712.18	0	3401.84	0	8436.75	0
Tech	8	25106.4	0	3983.84	0	13165	0	1624.69	0	29002.4	0	2849.04	0	14233.1	0
CD	5	24847.1	0	13027	0	24361.4	0	10211.5	0	17446.1	0	12863.6	0	14138.3	0
NL	4	18249.8	0	12569.7	0	19943.1	0	10206.6	0	12602.1	0	12842	0	11198.5	0
ND	4	7455.69	0	4446.95	0	6477.61	0	2381.84	0	11140.3	0	4587.43	0	4067.42	0
Lrn × Tech	24	6017.02	0	887.98	0	3020.81	0	314.48	0	966.03	0	577.66	0	4539.11	0
Lrn × CD	15	299.77	0	42.44	0	120.68	0	51.93	0	24.54	0	37.17	0	166.69	0
Tech × CD	40	123.26	0	64.99	0	96.86	0	31.84	0	547.26	0	65.94	0	190.86	0
Lrn × NL	12	253.1	0	39.15	0	219.7	0	40.73	0	77.49	0	66.23	0	191.58	0
Tech × NL	32	197.2	0	117.28	0	146.32	0	125.58	0	333.2	0	108.85	0	141.44	0
CD × NL	20	233.77	0	41.15	0	85.44	0	6.96	0	8.93	0	27.02	0	40.68	0
Lrn × ND	12	445.62	0	288.67	0	322.74	0	473.35	0	240.8	0	290.27	0	278.33	0
Tech × ND	32	228.59	0	41.72	0	163.46	0	271.31	0	869.72	0	52	0	226.32	0
CD × ND	20	159.68	0	48.26	0	88.16	0	20.77	0	24.02	0	45.93	0	45.81	0
NL × ND	15	1843.67	0	983.25	0	1570.64	0	635.58	0	1356.64	0	942.29	0	909.95	0
Lrn × Tech × CD	120	57.51	0	8.94	0	50.37	0	18.13	0	26.59	0	12.22	0	139.74	0
Lrn × Tech × NL	96	27.25	0	4.78	0	16.93	0	7.16	0	12.58	0	4.37	0	50.69	0
Lrn × CD × NL	60	8.42	0	8.35	0	10.35	0	4.58	0	3.35	0	7.68	0	7.24	0
Tech × CD × NL	160	16.38	0	4.3	0	11.72	0	5.06	0	10.6	0	4.25	0	9.68	0
Lrn × Tech × ND	96	82.07	0	10.11	0	64.65	0	12.64	0	27.6	0	17.5	0	119.68	0
Lrn × CD × ND	60	6.41	0	1.44	.0152	8.81	0	9.43	0	22.65	0	3.15	0	8.94	0
Tech × CD × ND	160	21	0	6.4	0	14.38	0	3.22	0	14.43	0	5.49	0	13.94	0
Lrn × NL × ND	45	44.78	0	12.83	0	24.67	0	10.56	0	15.98	0	13.12	0	21.88	0
Tech × NL × ND	120	16.79	0	3.74	0	10.43	0	17.08	0	60.44	0	3.86	0	15.41	0
CD × NL × ND	75	44.19	0	8.4	0	20.71	0	1.67	.0003	4.87	0	7.42	0	13.54	0

A. Impact of the Five Experimental Factors

Table III presents the ANOVA [44] table analyzing each of the main factors (Lrn, Tech, CD, NL, and ND), as well as their two- and three-way interactions. The table shows that each of the five factors, as well as all of their two- and three-way interactions, is statistically significant at the $\alpha = 5\%$ level for all seven performance metrics ($p = 0$ implies that the p-value is less than 0.0001). ANOVA was performed using SAS [45], and the assumptions required for valid ANOVA were satisfied by our experimental data.

ANOVA can be used to test the hypothesis that the average performance for each level of the main factors (learner, technique, CD, NL, and ND) is equal against the alternative hypothesis that at least one is different. If the alternative hypothesis (i.e., at least one average performance is different) is accepted, numerous procedures can be used to determine which of the values are significantly different from the others. This involves a comparison of two average performance values, with the null hypothesis that they are equal. A type-I error occurs when the null hypothesis is incorrectly rejected. In this study, we use Tukey's Honestly Significant Difference (HSD) test [45], which controls the type-I experimentwise error rate [44].

Since each of the main factors is shown to be statistically significant, we perform the HSD test, indicating which levels of these factors result in significantly different performances when compared to the other levels of that factor. For example, Table IV provides the results of the HSD test for the main factor Lrn. This factor has four levels (the four different learners used in our experiments), each of which is assigned to a group (indicated by a letter) based on its average performance (across all of the other factors, as well as ten runs of tenfold cross validation). If multiple levels (learners) are assigned to the same group (i.e., they have the same letter in a column in this table), then their average performances were not significantly different based on the performance metric indicated in that column. Although the actual values for the average performances are not provided, the assigned letters provide information on their relative performances. Learners in group A performed better than those in group B, which performed better than those in group C, and so forth.

TABLE IV
TUKEY'S HSD TEST FOR THE MAIN FACTOR: LEARNER (LRN)

Learner	AROC APRC BFM	KS	FM	GM	BGM
C4.5D	C	B	B	C	B
C4.5N	A	A	A	B	A
NB	B	A	D	A	A
RIPPER	D	C	C	D	B

TABLE V
TUKEY'S HSD TEST FOR THE MAIN FACTOR: TECHNIQUE (TECH)

Tech	AROC	APRC	KS	FM	GM	BFM	BGM
EBBagN	A	A	A	C	A	A	A
EBBagR	B	B	B	A	B	B	B
NONE	G	G	G	F	F	F	F
RBBagN	A	A	A	C	A	A	A
RBBagR	B	B	B	A	B	B	B
RUSBoostS	D	E	E	B	C	E	D
RUSBoostW	C	C	C	B	B	C	C
SMOTEBoostS	F	D	F	E	E	D	E
SMOTEBoostW	E	F	D	D	D	E	D

The results of Tukey's HSD test for the factors learner and technique are presented in Tables IV and V, respectively. Table IV shows that there is a significant difference in performance depending on the selected learner. Since AROC, APRC, and BFM have identical HSD test results, they are combined into a single column in this table. In general, C4.5N performs the best overall in our experiments, being in group A for all metrics except GM, for which it is second only to NB. NB also performed quite well for most metrics, being in group A for three of the seven metrics and in group B for three others. Using FM to measure performance, however, NB is the worst of the four learners. With the exception of the FM performance measure, C4.5D and RIPPER do not perform as well as C4.5N or NB, with C4.5D usually outperforming RIPPER. Recall, however, that these results are based on the average performance across all CDs, NLs, and NDs, as well as using all eight boosting/bagging techniques and NONE. While comparing the learners is not the focus of this paper, it is worth noting that these relative performance rankings can change as the other factors are changed.

Table V shows the relative performance of the boosting and bagging techniques, averaged across all of the other experimental parameters (including the four learners). Regardless of the performance metric, these results are relatively consistent, with the exception of FM. This table shows several common trends that remain relatively constant throughout this study. These trends, which will be referenced throughout this report, are as follows.

1) In general, the bagging techniques perform better when data are noisy and imbalanced than boosting techniques.
2) There is no significant difference between the performances of EBBag and RBBag.
3) Bagging should be implemented without replacement. For all performance metrics except FM, the two bagging versions implemented without replacement outperformed those implemented with replacement.
4) In general, RUSBoost outperforms SMOTEBoost, but both are outperformed by the bagging techniques in most of our experiments.

Once again, note that the results in Table V are averaged over all of the other experimental factors. For example, on average (i.e., averaged over all learners/imbalance levels/NLs/NDs), EBBagN and RBBagN have a significantly higher average AROC than the remaining techniques (although there is no significant difference between EBBagN and RBBagN). Section IV-B provides more details about the performance of the boosting and bagging techniques as the other parameters are varied. For space considerations, we omit further analysis of the experimental factors CD, NL, and ND and instead focus on our primary objective.

B. Evaluating the Boosting and Bagging Techniques

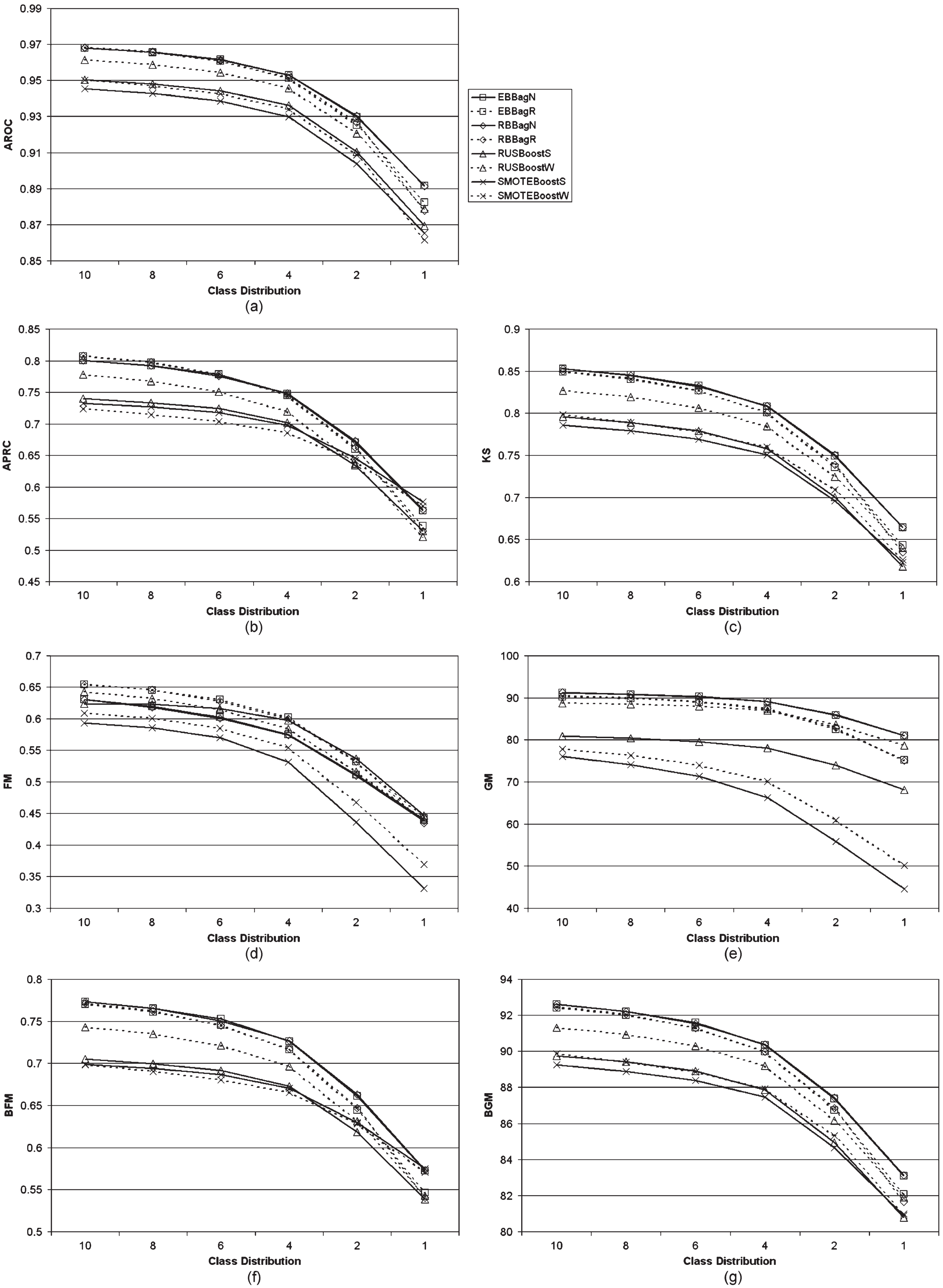
The previous section demonstrated a general difference, averaged over all of the other experimental factors, between the boosting and bagging techniques. In this section, we further refine our initial observations by examining the performance of these techniques as the learner, CD, NL, and ND are varied. The ANOVA models presented in Table III include two- and three-way interactions, all of which are statistically significant. This section will consider the four different two-way interactions that involve the boosting and bagging techniques: Tech×CD, Tech×NL, Tech×ND, and Tech×Lrn. In a practical sense, the significance of the interaction term Tech×CD, for example, implies that the relative mean performances of the boosting/bagging techniques are significantly impacted by changes in the CD. Due to space limitations, we are unable to present additional results and focus instead on our primary objective, i.e., evaluating the performance of the boosting and bagging techniques.

1) CD: Fig. 1 shows the performance of the eight different versions and implementations of boosting and bagging for CDs from $CD = 10\%$ to $CD = 1\%$ (interaction Tech×CD). Each graph within Fig. 1 presents the results using a different performance metric. These results are based on the average performance across the four learners and therefore provide a general idea about each technique's performance when data are noisy with various levels of imbalance. NONE is omitted from this figure to increase graph clarity, but we note that it is outperformed (significantly) by all of the boosting and bagging techniques. Each version of boosting and bagging is identified by a different shape, as indicated in the legend. The two implementations of each version are distinguished by using solid versus dashed lines.

Examining the various graphs in Fig. 1, a few general conclusions can be drawn, some of which match those conclusions drawn in Section IV-A and some of which expand upon those conclusions. For example, the general conclusion that bagging is superior to boosting when training data are both noisy and imbalanced is supported. Fig. 1 shows that this observation is generally true regardless of the level of imbalance. Using almost every performance metric, the four lines representing the bagging techniques show better performance than those of the boosting techniques for almost all levels of CD. Another observation from Section IV-A can be expanded upon. In general, it is true that bagging without replacement is superior to bagging with replacement, but this difference is most evident when the class imbalance is most severe. Given mild or moderate class imbalance, there is little difference between bagging with and without replacement for most metrics (with the exception of FM). Relative to FM and GM, SMOTEBoostS and SMOTEBoostW both perform much worse than the other techniques when the data are most imbalanced, while at more moderate levels of imbalance, their performance is more similar to that of the other techniques.

Table VI presents the results of the HSD test, indicating the significance of the results shown in Fig. 1. The HSD test was used to compare the performance of the various bagging and boosting techniques at each level of imbalance. As such, the HSD grouping is listed horizontally. For example, the first row of the table compares the AROC of the eight boosting and bagging techniques (and NONE) when $CD = 10$. Since they all share the same letter A, there is no significant difference between the four bagging techniques. RUSBoostW was assigned to group B and is the next best technique. Both RUSBoostS and SMOTEBoostW are in group C, which means that there was no significant difference between them, but both are significantly worse than RUSBoostW. Finally, SMOTEBoostS is in group D, significantly outperforming only NONE.

Table VI demonstrates the statistical significance of the conclusions drawn based on Fig. 1. Specifically, the bagging



(a)

(b)

(c)

(d)

(e)

(f)

(g)

Fig. 1. Performance of the boosting and bagging techniques across various CDs. For each metric, the performance is averaged over all data sets, learners, NDs, and NLs.

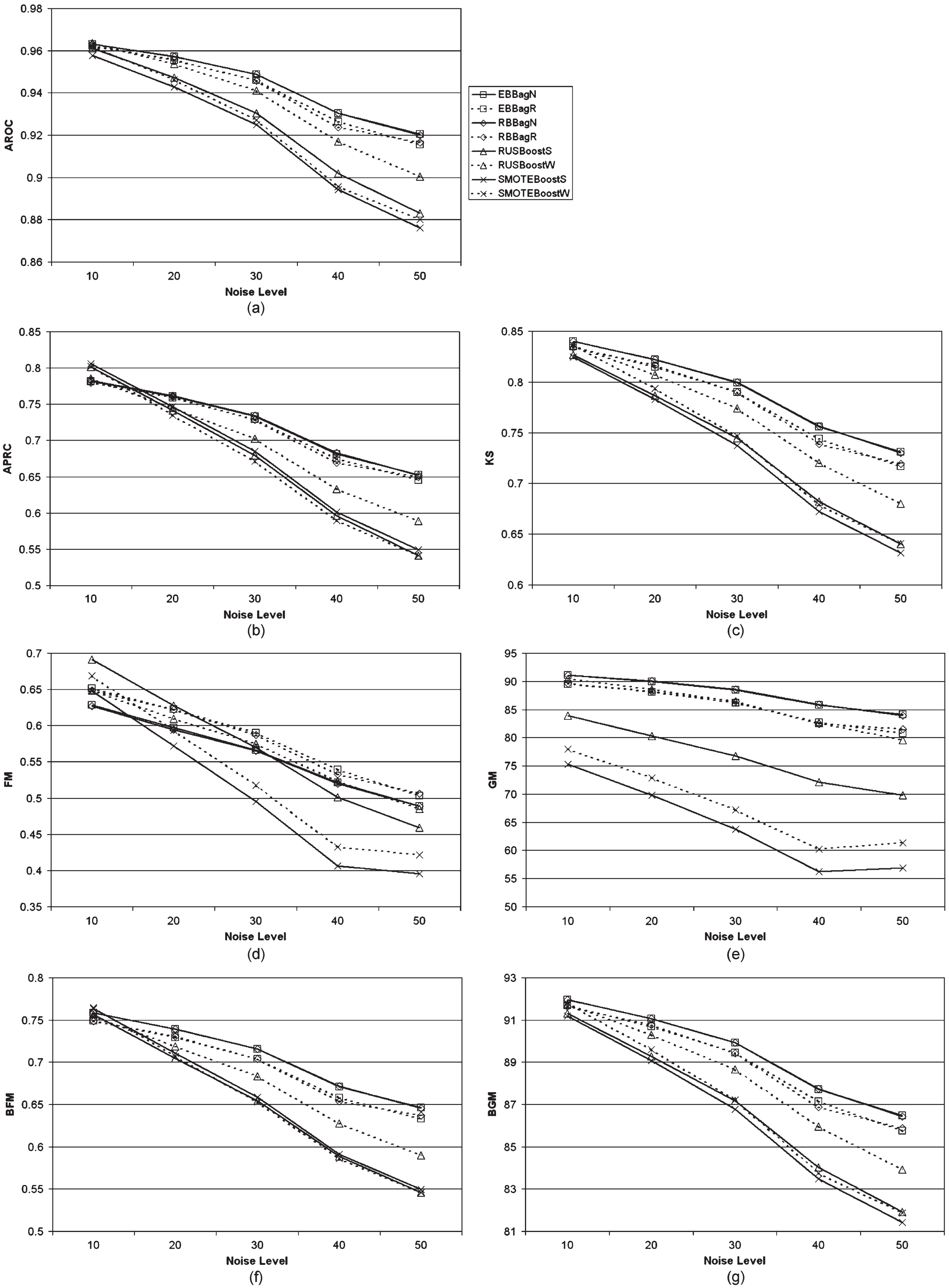
TABLE VI
HSD TEST RESULTS COMPARING THE BAGGING AND BOOSTING TECHNIQUES AT VARIOUS LEVELS OF CLASS IMBALANCE FOR SEVEN PERFORMANCE METRICS

	CD	NONE	EBBagN	EBBagR	RBBagN	RBBagR	RUSBoostS	RUSBoostW	SMOTEBoostS	SMOTEBoostW
AROC	10	E	A	A	A	A	C	B	D	C
	8	E	A	A	A	A	C	B	D	C
	6	E	A	A	A	A	C	B	D	CD
	4	E	A	A	A	A	C	B	D	CD
	2	E	A	AB	A	A	C	B	D	CD
	1	E	A	B	A	B	C	B	CD	D
KS	10	E	A	A	A	A	C	B	D	C
	8	E	A	A	A	A	C	B	D	C
	6	E	A	A	A	A	C	B	D	CD
	4	D	A	A	A	A	C	B	C	C
	2	G	AB	C	A	BC	EF	D	F	E
	1	E	A	B	A	BC	D	B	CD	CD
APRC	10	E	A	A	A	A	C	B	CD	D
	8	E	A	A	A	A	C	B	C	D
	6	E	A	A	A	A	C	B	C	D
	4	E	A	A	A	A	C	B	CD	D
	2	E	AB	BC	A	AB	D	D	CD	D
	1	E	B	C	AB	CD	CD	D	A	AB
FM	10	F	BC	A	BC	A	C	AB	E	D
	8	F	BC	A	C	A	BC	B	E	D
	6	H	DE	A	E	AB	BC	CD	G	F
	4	F	C	A	C	A	AB	BC	E	D
	2	E	B	A	B	A	A	B	D	C
	1	D	A	A	A	A	A	A	C	B
GM	10	G	A	B	A	AB	D	C	F	E
	8	G	A	B	A	B	D	C	F	E
	6	G	A	B	A	B	D	C	F	E
	4	F	A	B	A	B	C	B	E	D
	2	G	A	C	A	BC	D	B	F	E
	1	G	A	C	A	C	D	B	F	E
BFM	10	D	A	A	A	A	C	B	C	C
	8	D	A	A	A	A	C	B	C	C
	6	E	A	A	A	A	C	B	CD	D
	4	D	A	A	A	A	C	B	C	C
	2	E	A	B	A	B	D	C	C	CD
	1	C	A	B	A	B	B	B	A	A
BGM	10	E	A	A	A	A	CD	B	D	C
	8	D	A	A	A	A	C	B	C	C
	6	D	A	A	A	A	C	B	C	C
	4	D	A	A	A	A	C	B	C	C
	2	D	A	AB	A	AB	C	B	C	C
	1	E	A	B	A	BC	D	B	CD	CD

techniques not only outperform the boosting techniques in most cases but also significantly outperform them. Furthermore, in many cases, the difference between bagging with and without replacement is not significant, but for some metrics, particularly when the CD is close to $CD = 1$, the difference becomes significant. The opposite is true using FM to evaluate the models. In this case, bagging with replacement significantly outperforms bagging without replacement, except when the data are most severely imbalanced.

2) NL: Fig. 2 shows the impact of the level of class noise on the performance of the various boosting and bagging techniques. Once again, NONE is omitted from these graphs since all eight boosting and bagging techniques outperform NONE and to improve graph clarity. When the data are relatively clean, the performance of the bagging and boosting techniques is very similar. However, as the amount of noise in the training data is increased, the difference in performance between the boosting and bagging techniques increases dramatically. For example, using AROC [Fig. 2(a)], there is very little difference between the best technique and the worst technique (less than 0.006) for $NL = 10$, and RUSBoostW performs the best. However, as the NL is increased, the bagging techniques are clearly superior to the boosting techniques. Once again, bagging without replacement is superior to bagging with replacement, with no difference between the two versions of bagging. Also, as shown in the previous sections, RUSBoost is the better of the two boosting versions.

Similar results have been seen for all seven performance metrics, with few minor exceptions. Using APRC [Fig. 2(b)] to measure performance, RUSBoost and SMOTEBoost both



(a)

(b)

(c)

(d)

(e)

(f)

(g)

Fig. 2. Performance of the boosting and bagging techniques across various NLs. For each metric, the performance is averaged over all data sets, learners, CDs, and NDs.

TABLE VII
HSD Test Results Comparing the Bagging and Boosting Techniques at Various Levels of Noise for Seven Performance Metrics

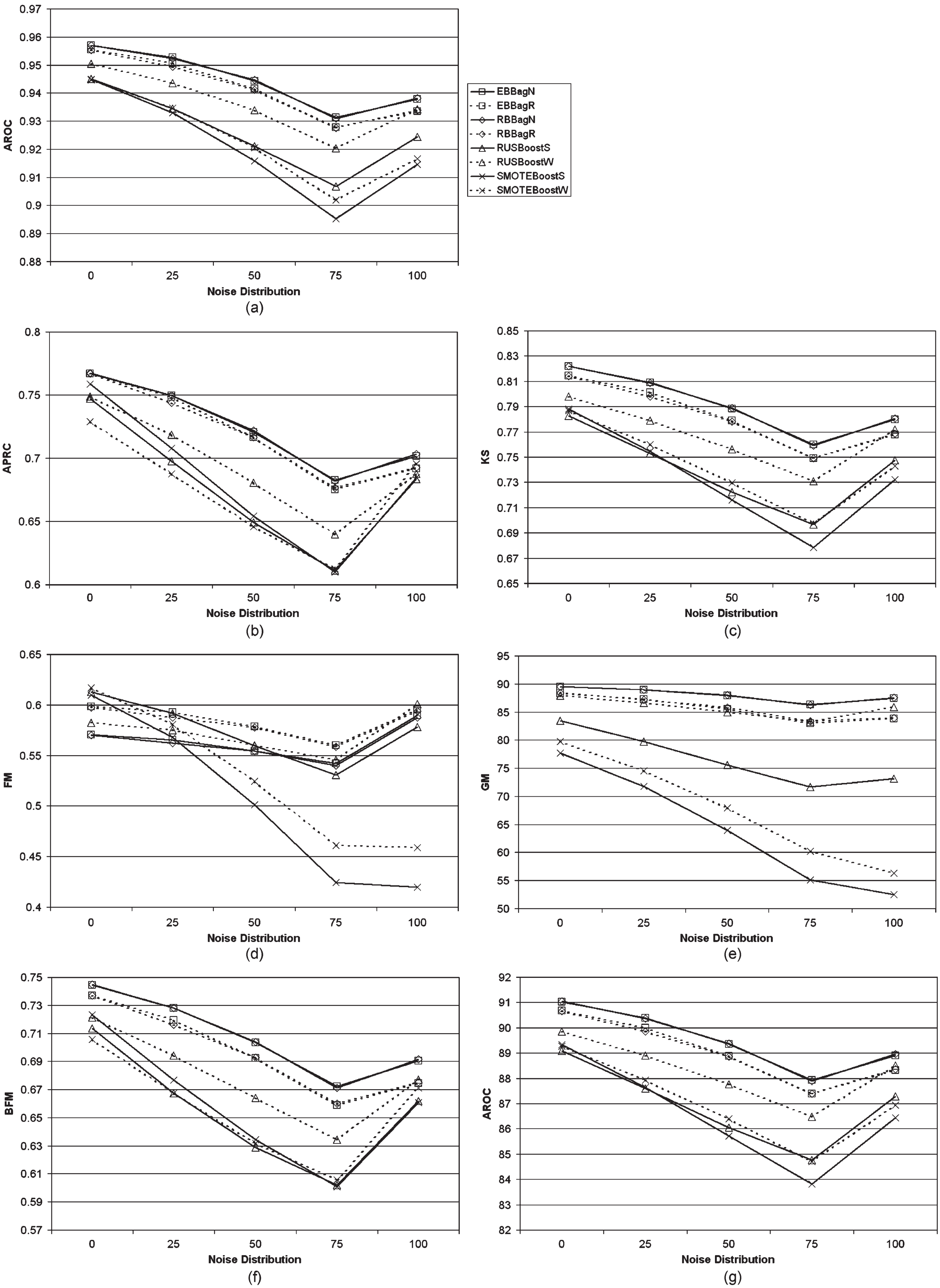
	NL	NONE	EBBagN	EBBagR	RBBagN	RBBagR	RUSBoostS	RUSBoostW	SMOTEBoostS	SMOTEBoostW
AROC	10	C	A	A	A	A	A	A	B	A
	20	D	A	A	A	A	B	A	C	BC
	30	E	A	A	A	A	C	B	D	CD
	40	F	A	AB	A	B	D	C	E	E
	50	E	A	A	A	A	C	B	D	CD
APRC	10	C	B	B	B	B	A	B	A	A
	20	D	A	A	A	A	BC	BC	B	C
	30	E	A	A	A	A	CD	B	C	D
	40	E	A	AB	A	B	D	C	D	D
	50	D	A	A	A	A	C	B	C	C
KS	10	D	A	A	A	AB	BC	A	C	A
	20	E	A	AB	A	A	CD	B	D	C
	30	E	A	B	A	B	D	C	D	D
	40	E	A	B	A	B	D	C	D	D
	50	E	A	B	A	AB	D	C	D	D
FM	10	E	D	C	D	C	A	C	C	B
	20	E	C	A	C	A	A	B	D	C
	30	E	B	A	B	A	B	B	D	C
	40	G	BC	A	C	AB	D	BC	F	E
	50	F	B	A	B	A	C	B	E	D
GM	10	F	A	B	A	B	C	A	E	D
	20	F	A	B	A	B	C	B	E	D
	30	F	A	B	A	B	C	B	E	D
	40	F	A	B	A	B	C	B	E	D
	50	G	A	B	A	B	D	C	F	E
BFM	10	E	AB	CD	AB	D	ABC	BCD	A	A
	20	E	A	B	A	B	D	C	CD	D
	30	E	A	B	A	B	D	C	D	D
	40	E	A	B	A	B	D	C	D	D
	50	E	A	B	AB	AB	D	C	D	D
BGM	10	D	A	AB	A	ABC	BC	AB	C	AB
	20	D	A	AB	A	AB	C	B	C	C
	30	D	A	A	A	A	C	B	C	C
	40	E	A	AB	A	B	D	C	D	D
	50	D	A	A	A	A	C	B	C	C

outperform all four of the bagging techniques on the cleanest data, but they are more sensitive to increases in noise, performing worse than the bagging techniques for all $NL \geq 20$. Using FM [Fig. 2(d)], the usual exception to the general trends (bagging with replacement outperforming bagging without replacement) is observed. Also, using FM, RUSBoost implemented with resampling performs better than usual, performing as well as or better than bagging when $NL \leq 20$, and as good as bagging without replacement, for $NL = 30$. SMOTEBoost also performs well at the lowest NL but is very sensitive to increases in noise, performing worse than the other techniques for $NL \geq 20$.

Table VII presents the results of Tukey's HSD test, showing the significance of the results shown in Fig. 2. As in Table VI, the HSD test results are meant to be read horizontally. For example, the first row provides a comparison of the boosting and bagging techniques using AROC at the lowest NL, i.e., $NL = 10$. At this NL, using this metric, seven of the boosting and bagging techniques (the exception is SMOTEBoostS) performed similarly (they are all assigned to group A), and all outperformed NONE. This table supports the conclusions drawn based on Fig. 2. For example, using AROC, while the eight bagging and boosting techniques perform similarly when $NL = 10$, as the NL is increased, the bagging techniques emerge as the better option, significantly outperforming the boosting techniques.

In summary, at low levels of class noise, there is little difference in performance between the boosting and bagging techniques, and any would significantly improve the performance of the baseline learner. However, at higher NLs, technique selection becomes critical, and regardless of the performance metric (with the possible exception of FM), we recommend using either EBBagN or RBBagN.

3) ND: The impact of ND on each of the bagging and boosting techniques is shown in Fig. 3. Before analyzing the information presented in these graphs, one small apparent anomaly must be explained. In each graph in Fig. 3, there is a clear trend as the ND is increased from 0 to 75. As more and more of the noise in the data is created by changing the label of positive (minority class) examples, the performance of the resulting models declines. This suggests that (as illustrated by Van Hulse *et al.* [2]) noise of type $\mathrm{P} \rightarrow \mathrm{N}$ is more detrimental to



(a)

(b)

(c)

(d)

(e)

(f)

(g)

Fig. 3. Performance of the boosting and bagging techniques across various NDs. For each metric, the performance is averaged over all data sets, learners, CDs, and NLs.

TABLE VIII
HSD Test Results Comparing the Bagging and Boosting Techniques at Various Levels of ND for Seven Performance Metrics

	ND	NONE	EBBagN	EBBagR	RBBagN	RBBagR	RUSBoostS	RUSBoostW	SMOTEBoostS	SMOTEBoostW
AROC	0	D	A	A	A	A	C	B	C	C
	25	D	A	A	A	A	C	B	C	C
	50	E	A	A	A	A	C	B	D	C
	75	E	A	A	A	A	C	B	D	C
	100	D	A	A	A	A	B	A	C	C
APRC	0	E	A	A	A	A	C	BC	AB	D
	25	E	A	A	A	A	CD	B	BC	D
	50	D	A	A	A	A	C	B	C	C
	75	D	A	A	A	A	C	B	C	C
	100	D	AB	ABC	A	ABC	C	BC	C	ABC
KS	0	D	A	A	A	A	C	B	C	C
	25	E	A	AB	A	B	D	C	D	D
	50	F	A	B	A	B	DE	C	E	D
	75	E	A	A	A	A	C	B	D	C
	100	E	AB	B	A	AB	C	AB	D	CD
FM	0	E	DE	BC	E	C	A	D	AB	A
	25	E	CD	A	D	A	A	BC	CD	AB
	50	E	B	A	B	A	B	B	D	C
	75	G	CD	A	CD	AB	D	BC	F	E
	100	E	AB	A	AB	A	B	A	D	C
GM	0	F	A	B	A	B	C	B	E	D
	25	F	A	B	A	B	C	B	E	D
	50	F	A	B	A	B	C	B	E	D
	75	F	A	B	A	B	C	B	E	D
	100	G	A	C	A	C	D	B	F	E
BFM	0	E	A	A	A	A	CD	BC	B	D
	25	F	A	AB	A	B	E	C	D	E
	50	E	A	B	A	B	D	C	D	D
	75	F	A	C	AB	BC	E	D	E	E
	100	D	A	B	A	B	C	B	C	BC
BGM	0	D	A	A	A	A	C	B	BC	C
	25	D	A	A	A	A	C	B	C	C
	50	E	A	A	A	A	CD	B	D	C
	75	E	A	A	A	A	C	B	D	C
	100	D	A	A	A	A	B	A	C	BC

classification performance than noise of type N → P. However, when the ND is increased from 75 to 100, there is an apparent improvement in performance. This is because, as explained in Section III, the combination of $NL = 50$ and $ND = 100$ could not be performed (because there would be no minority class examples left in the training data). Since this level of noise $NL = 50$ is not performed for $ND = 100$, the performance at $ND = 100$ appears to be better than that when $ND = 75$. To avoid confusion, analysis in this section will be confined to NDs $0 \leq ND \leq 75$.

As the ND is increased from 0 to 75, the difference between the performances of the bagging and boosting techniques becomes much greater. However, unlike in Fig. 2, the performance gap between boosting and bagging is significant even at the lowest level of ND. Table VIII presents the results of the HSD test for these experiments, and even with $ND = 0$, bagging often significantly outperforms boosting. The usual trends observed throughout this study also apply across the different values of ND, so these will not be restated. The primary conclusion derived from this section, however, is that the dispersion of performance increases dramatically as the ND is increased from 0 to 75, and hence, as a larger percentage of noise is of type P → N, technique selection becomes more critical as bagging will dramatically outperform boosting.

4) Techniques and Learners: In this section, we identify which techniques work best with the individual learners. Table IX shows the average performances for the various combinations of boosting/bagging technique and learner and the HSD test results indicating which of the techniques result in significantly different performances for the given learner. In general, the primary trends already identified apply across learners. That is, for all four learners, bagging tends to perform better than boosting, bagging without replacement performed better than bagging with replacement, and RUSBoost performs better than SMOTEBoost. There are some exceptions, however. For example, using C4.5N and RIPPER, RUSBoost (either performed with resampling or reweighting) often performs as well as the bagging techniques. In addition, for most metrics, NB is not significantly improved by boosting or bagging, while the other three learners are. Therefore, we observe that boosting or bagging the NB learner may not significantly improve upon the base NB learner. For the C4.5N and RIPPER

TABLE IX
HSD TEST RESULTS COMPARING THE BAGGING AND BOOSTING TECHNIQUES FOR EACH LEARNER AVERAGED OVER ALL DATA SETS, CDs, NLs, AND NDs

		NONE		EBBagN		EBBagR		RBBagN		RBBagR		RUSBoostS		RUSBoostW		SMOTEBoostS		SMOTEBoostW	
	Learner	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD	Mean	HSD
AROC	C4.5D	.695	D	.948	A	.947	A	.948	A	.946	A	.923	C	.936	B	.925	C	.927	C
	C4.5N	.897	D	.949	A	.948	A	.948	A	.948	AB	.941	C	.948	AB	.945	B	.942	C
	NB	.947	ABC	.949	A	.946	BC	.949	AB	.946	C	.932	E	.937	D	.919	G	.923	F
	RIPPER	.712	F	.934	A	.928	B	.935	A	.928	B	.911	C	.926	B	.895	E	.905	D
APRC	C4.5D	.411	D	.718	A	.725	A	.718	A	.723	A	.676	C	.703	B	.705	B	.695	B
	C4.5N	.583	E	.743	AB	.740	ABC	.743	AB	.740	ABC	.731	C	.749	A	.733	BC	.712	D
	NB	.752	B	.763	A	.750	B	.763	A	.749	B	.664	C	.662	C	.656	C	.630	D
	RIPPER	.413	D	.677	A	.669	A	.679	A	.669	A	.639	C	.667	AB	.638	C	.655	B
KS	C4.5D	.396	F	.803	A	.796	A	.802	A	.795	A	.728	E	.763	B	.738	D	.749	C
	C4.5N	.701	E	.799	A	.797	AB	.799	A	.796	AB	.772	D	.792	B	.780	C	.775	CD
	NB	.800	A	.802	A	.793	B	.802	A	.792	B	.761	D	.777	C	.734	F	.740	E
	RIPPER	.401	F	.764	A	.746	B	.766	A	.746	B	.698	D	.737	B	.685	E	.710	C
FM	C4.5D	.389	F	.582	B	.616	A	.580	B	.613	A	.585	B	.558	C	.508	E	.545	D
	C4.5N	.489	G	.570	CD	.620	AB	.568	D	.617	B	.628	A	.581	C	.526	F	.549	E
	NB	.525	CD	.515	DE	.528	C	.514	E	.527	C	.556	AB	.566	A	.550	B	.518	CDE
	RIPPER	.466	E	.588	A	.574	B	.587	A	.573	B	.528	C	.582	AB	.449	F	.513	D
GM	C4.5D	46.07	F	89.44	A	87.34	B	89.38	A	87.36	B	76.73	C	86.40	B	63.03	E	67.43	D
	C4.5N	66.05	E	89.24	A	87.59	B	89.22	A	87.66	B	83.15	C	88.59	A	64.47	F	67.54	D
	NB	68.71	E	87.94	A	86.09	B	87.94	A	86.17	B	79.21	C	86.89	B	73.34	D	73.88	D
	RIPPER	59.22	E	85.66	A	81.65	B	85.81	A	82.07	B	68.36	C	81.14	B	57.99	F	64.02	D
BFM	C4.5D	.488	E	.713	A	.707	A	.712	A	.706	A	.648	D	.681	B	.673	BC	.670	C
	C4.5N	.607	D	.721	A	.713	A	.721	A	.713	A	.695	B	.717	A	.697	B	.684	C
	NB	.726	A	.729	A	.714	B	.729	A	.714	B	.659	C	.664	C	.649	D	.630	E
	RIPPER	.497	E	.672	A	.656	B	.673	A	.656	B	.616	D	.652	B	.618	D	.640	C
BGM	C4.5D	52.37	E	90.10	A	89.74	A	90.06	A	89.68	A	86.37	D	88.09	B	86.81	CD	87.36	BC
	C4.5N	84.83	D	89.91	A	89.78	AB	89.91	A	89.75	AB	88.58	C	89.54	B	88.93	C	88.68	C
	NB	89.93	A	90.04	A	89.58	B	90.04	A	89.55	B	87.98	D	88.76	C	86.53	F	86.89	E
	RIPPER	67.94	E	88.14	A	87.24	B	88.19	A	87.22	B	84.85	C	86.79	B	84.09	D	85.30	C

TABLE X
MISCLASSIFICATION RATE BY BOOSTING ITERATION, RUSBOOSTW WITH THE C4.5D LEARNER, OPTDIGITS8, $CD = 4\%$, $NL = 20\%$, AND $ND = 50\%$

	Iteration									
Type	1	2	3	4	5	6	7	8	9	10
N→N	16.23%	16.72%	23.63%	23.19%	19.32%	27.98%	41.30%	33.45%	29.58%	28.77%
P→N	73.33%	70.83%	54.17%	70.00%	68.33%	70.83%	51.67%	65.00%	44.17%	54.17%
Total Negative Class	16.70%	17.17%	23.89%	23.58%	19.73%	28.33%	41.38%	33.72%	29.70%	28.99%
P→P	2.71%	7.50%	13.54%	10.42%	15.83%	16.25%	17.08%	16.25%	40.63%	19.58%
N→P	26.67%	32.50%	21.67%	25.83%	36.67%	20.83%	19.17%	25.83%	40.83%	28.33%
Total Positive Class	7.50%	12.50%	15.17%	13.50%	20.00%	17.17%	17.50%	18.17%	40.67%	21.33%
Overall	16.33%	16.99%	23.54%	23.17%	19.74%	27.89%	40.43%	33.09%	30.14%	28.68%

TABLE XI
AVERAGE WEIGHTS BY BOOSTING ITERATION, RUSBOOSTW WITH THE C4.5D LEARNER, OPTDIGITS8, $CD = 4\%$, $NL = 20\%$, and $ND = 50\%$

	Iteration									
Type	1	2	3	4	5	6	7	8	9	10
N→N	1.00	1.00	0.99	0.98	0.97	0.95	0.94	0.93	0.93	0.92
P→N	1.00	2.26	4.11	5.26	7.07	9.48	11.14	11.54	12.71	12.98
Total Negative Class	1.00	1.01	1.01	1.02	1.02	1.02	1.02	1.02	1.02	1.02
P→P	1.00	0.72	0.59	0.51	0.43	0.38	0.35	0.34	0.31	0.32
N→P	1.00	1.23	1.22	1.07	0.99	0.98	0.91	0.88	0.86	0.86
Total Positive Class	1.00	0.82	0.72	0.62	0.54	0.50	0.47	0.45	0.42	0.43

classifiers, RUSBoostW performs as well as bagging with replacement and often performs as well as bagging without replacement. C4.5D is the only learner where bagging without replacement consistently outperforms RUSBoostW.

The results in Table IX can also be used to compare the learner–technique combinations to one another, for example, C4.5D+EBBagN with NB+NONE. The first four rows of the table compare all 36 AROC combinations of learner+technique, and overall, NB+EBBagN has the highest AROC of 0.949. We omit a detailed analysis of these results, except to state that using NB as the baseline classifier with either EBBagN or RBBagN often produces strong performance for AROC, APRC, KS, BFM, and BGM, while decision tree learners C4.5N and C4.5D with bagging work well when the objective is to optimize GM or FM.

5) Discussion of Results: The general conclusion obtained from our experiments is the clear preference of bagging over boosting when imbalanced data are noisy. The intuitive explanation for this result is that boosting will focus too much on noisy examples, hurting the overall performance. To illustrate this phenomenon, Tables X and XI provide additional details on the performance of the RUSBoostW algorithm for one particular experimental data set (OptDigits8 with $CD = 4\%$, $NL = 20\%$, and $ND = 50\%$) over the ten boosting iterations using C4.5D as the base learner. Table X shows the misclassification rates for each iteration for both noisy and non-noisy positive

TABLE XII
MISCLASSIFICATION RATE BY BOOSTING ITERATION, RUSBOOSTW WITH THE C4.5D LEARNER, OPTDIGITS8, $CD = 4\%$, $NL = 40\%$, AND $ND = 100\%$

	Iteration									
Type	1	2	3	4	5	6	7	8	9	10
N→N	19.94%	29.96%	24.24%	30.15%	30.29%	42.10%	28.56%	45.94%	28.94%	51.03%
P→N	71.25%	61.04%	68.75%	68.29%	67.97%	68.45%	53.27%	64.24%	68.33%	68.33%
Total Negative Class	21.60%	30.96%	25.68%	31.38%	31.50%	42.95%	29.36%	46.53%	30.22%	51.59%
P→P	4.17%	6.67%	12.96%	4.63%	7.29%	13.10%	20.24%	12.50%	21.67%	16.67%
Overall	21.53%	30.86%	25.63%	31.28%	31.40%	42.83%	29.32%	46.39%	30.18%	51.45%

TABLE XIII
AVERAGE WEIGHTS BY BOOSTING ITERATION, RUSBOOSTW WITH THE C4.5D LEARNER, OPTDIGITS8, $CD = 4\%$, $NL = 40\%$, AND $ND = 100\%$

	Iteration									
Type	1	2	3	4	5	6	7	8	9	10
N→N	1.00	0.97	0.96	0.93	0.91	0.90	0.89	0.89	0.90	0.88
P→N	1.00	1.85	2.42	3.28	3.69	4.15	4.52	4.49	4.11	4.62
Total Negative Class	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.01
P→P	1.00	0.75	0.67	0.56	0.50	0.45	0.43	0.41	0.39	0.37

and negative class instances, while Table XI shows the average weight for each iteration. Non-noisy positive and negative class instances are denoted by P → P and N → N, respectively.

In the first iteration, the base learner C4.5D favors the positive class, which has a misclassification rate of 7.50% compared to the misclassification rate of 16.70% for the negative class. This is the reason for adding RUS to boosting—to enhance the performance of the learner on the positive class relative to the negative class. Within each class, the noisy examples are misclassified at a much higher rate than nonnoisy examples. N → P-type noise was misclassified at a 26.67% rate in the first iteration, while P → N-type noise was misclassified at a rate of 73.33%. Note that the vast majority of the examples are N → N-type examples, and this misclassification rate dominates the others in the calculation of the overall misclassification rate in the last row of Table X. The instance weights in iteration 1 (prior to model construction) are initialized to one. When updating instance weights, AdaBoost.M2 utilizes the posterior probabilities calculated by the learner. In iteration 2, the weights are updated using the results from iteration 1. The average weights for both P → N-type noise and N → P-type noise increase, with the former one increasing by a substantially larger amount. Therefore, C4.5D will place increased emphasis on these noisy instances in the second iteration.

After increasing for the first few iterations, the average weight for the N → P examples begins to decrease. Note that the misclassification rate for the N → P examples shows substantial volatility. Despite this, the average weight decreases since the weight calculation considers the posterior probabilities—even if instances are misclassified, if the posterior probabilities improve, then the weights will decrease. In contrast, for P → N-type noise, the average weights continue to increase, despite the misclassification rate generally decreasing. The posterior probabilities computed by C4.5D diverge further and further from the true class, even if there are fewer misclassified instances. For the negative class, the increasing emphasis on noisy examples leads to a substantial overall deterioration in performance for this class as a whole. The misclassification rate for the N → N examples increases from 16.23% in iteration 1 to as high as 41.3% in iteration 7. Similarly, the misclassification rate for the P → P examples increases from 2.71% in iteration 1 to as high as 40.63% in iteration 9. Relative to each class, the misclassification rates for noisy and nonnoisy instances generally converge as more iterations are run, and the overall misclassification rate tends to increase.

Tables XII and XIII provide the misclassification rates and average weights for RUSBoostW with the C4.5D learner using the OptDigits8 data set with $CD = 4\%$, $NL = 40\%$, and $ND = 100\%$. In this situation, there is no N → P-type noise. Similar patterns are observed here as were discussed previously, although the misclassification rates of P → N-type noise tend to stay much higher. In addition, the overall learner performance is worse; however, this is expected since this data set has both more noise and more P → N-type noise. These data support the hypothesis that noisy data adversely impact boosting algorithms and that the impact becomes more severe as the noise becomes more impactful.

V. CONCLUSION

This paper has presented the results of a comprehensive suite of experiments comparing the performance of eight techniques for dealing with the combined problems of class imbalance and class noise. Our experiments, for which nearly four million models were trained and evaluated, vary key factors of data quality including CD, NL, and ND and evaluate their impact on four common classifiers when trained using boosting and bagging techniques specifically calibrated to improve the performance of a base learner on imbalanced data. All models are evaluated using seven different performance metrics, providing a complete perspective on classification performance, and all results are tested for significance via ANOVA modeling.

The experimental results demonstrate several clear trends as enumerated in the following.

1) In general, the bagging techniques perform better when data are noisy and imbalanced than the boosting techniques. The difference between bagging and boosting is

less significant when the data are relatively clean but imbalanced. For data with significant amounts of impactful noise (i.e., high NLs or more P → N-type noise), however, bagging is strongly recommended over boosting, as class noise is clearly detrimental to the boosting techniques.

2) There is no significant difference between the performances of EBBag and RBBag. This holds true across all performance metrics and experimental parameters.
3) Bagging should be implemented without replacement. For all performance metrics except FM, the two bagging versions implemented without replacement outperformed those implemented with replacement. While the performance difference was not always statistically significant, it was consistent.
4) In general, RUSBoost outperforms SMOTEBoost, but both are outperformed by the bagging techniques in most of our experiments. However, when the NL is low, these techniques can outperform bagging.
5) While the relative performance of the bagging and boosting techniques is similar across most performance metrics and experimental parameters, there are exceptions, which typically occur at the extreme values of these parameters. For example, the FM and GM of SMOTEBoost (Fig. 3) deteriorate at a dramatically faster rate than those of other techniques as the ND increases.

Therefore, in summary, we recommend the use of bagging without replacement for learning from noisy and imbalanced data. Boosting procedures may be a very popular method for handling imbalanced data; however, improved noise handling is required to make both RUSBoost and SMOTEBoost robust enough to handle class noise. Future work will continue to investigate the effectiveness of boosting and bagging techniques in the context of noisy and imbalanced data. An investigation of their performance on multiclass data sets can be performed. Finally, we advocate additional research into the topic of improving the noise-handling ability of boosting techniques used to deal with imbalanced data.

References

[1] X. Zhu and X. Wu, “Class noise vs. attribute noise: A quantitative study of their impacts,” *Artif. Intell. Rev.*, vol. 22, no. 3/4, pp. 177–210, Nov. 2004.

[2] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, “Skewed class distributions and mislabeled examples,” in *Proc. IEEE ICDMW*, Omaha, NE, Oct. 2007, pp. 477–482.

[3] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, “Experimental perspectives on learning from imbalanced data,” in *Proc. 24th ICML*, Corvallis, OR, Jun. 2007, pp. 935–942.

[4] G. M. Weiss, “Mining with rarity: A unifying framework,” *SIGKDD Explor.*, vol. 6, no. 1, pp. 7–19, 2004.

[5] 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, Jun. 2002.

[6] Y. Freund and R. Schapire, “Experiments with a new boosting algorithm,” in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[7] 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*, Coconut Grove, FL, May 2008, pp. 306–311.

[8] A. Estabrooks, T. Jo, and N. Japkowicz, “A multiple resampling method for learning from imbalanced data sets,” *Comput. Intell., Int. J.*, vol. 20, no. 1, pp. 18–36, Feb. 2004.

[9] I. Davidson and W. Fan, “When efficient model averaging out-performs boosting and bagging,” in *Proc. 10th Eur. Conf. Principles Practice Knowl. Discov. Databases*, 2006, pp. 478–486.

[10] 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.

[11] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, “RUSBoost: A hybrid approach to alleviating class imbalance,” *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[12] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, “Boosting Classifiers Built With Skewed Data,” Dept. Comput. Sci. Eng., Florida Atlantic Univ., Boca Raton, FL, Jul. 2009, Tech. Rep..

[13] S. Hido and H. Kashima, “Roughly balanced bagging for imbalanced data,” in *Proc. 8th SIAM Int. Conf. Data Mining*, 2008, pp. 143–152.

[14] N. Japkowicz and S. Stephen, “The class imbalance problem: A systematic study,” *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–450, Oct. 2002.

[15] 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*, 2005, pp. 878–887.

[16] 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.

[17] D. Wilson, “Asymptotic properties of nearest neighbor rules using edited data sets,” *IEEE Trans. Syst., Man, Cybern.*, vol. 2, no. 3, pp. 408–421, Jul. 1972.

[18] 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. Imbalanced Data Sets II*, 2003, pp. 1–8.

[19] M. Maloof, “Learning when data sets are imbalanced and when costs are unequal and unknown,” in *Proc. ICML Workshop Learn. Imbalanced Data Sets*, 2003, pp. 73–80.

[20] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, “AdaCost: Misclassification cost-sensitive boosting,” in *Proc. 16th Int. Conf. Mach. Learn.*, San Francisco, CA, 1999, pp. 97–105.

[21] K. M. Ting, “A comparative study of cost-sensitive boosting algorithms,” in *Proc. 17th Int. Conf. Mach. Learn.*, San Francisco, CA, 2000, pp. 983–990.

[22] 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.

[23] M. V. Joshi, V. Kumar, and R. C. Agarwal, “Evaluating boosting algorithms to classify rare cases: Comparison and improvements,” in *Proc. 1st IEEE Int. Conf. Data Mining*, Nov. 2001, pp. 257–264.

[24] W. Jiang, “Some theoretical aspects of boosting in the presence of noisy data,” in *Proc. 18th ICML*, 2001, pp. 234–241.

[25] A. Karmaker and S. Kwek, “A boosting approach to remove class label noise,” in *Proc. 5th Int. Conf. HIS*, Washington, DC, 2005, pp. 206–211.

[26] N. C. Oza, “AveBoost2: Boosting for noisy data,” in *Proc. 5th Int. Workshop Multiple Classifier Syst.*, J. K. Fabio Roli and T. Windeatt, Eds., Jun. 2004, pp. 31–40.

[27] P. Kang and S. Cho, “EUS SVMs: Ensemble of under-sampled SVMs for data imbalance problems,” in *Proc. 13th Int. Conf. Neural Inf. Process.*, vol. 4232, *Lecture Notes in Computer Science*, Hong Kong, 2006, pp. 837–846, Part I.

[28] X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory under-sampling for class-imbalance learning,” in *Proc. 6th Int. Conf. Data Mining*, Hong Kong, 2006, pp. 965–969.

[29] M. Molinara, M. T. Ricamato, and F. Tortorella, “Facing imbalanced classes through aggregation of classifiers,” in *Proc. 14th Int. Conf. Image Anal. Process.*, Modena, Italy, 2007, pp. 43–48.

[30] R. Yan, Y. Liu, R. Jin, and A. Hauptmann, “On predicting rare classes with SVM ensembles in scene classification,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2003, pp. III-21-1–III-21-4.

[31] C. Seiffert, T. M. Khoshgoftaar, and J. Van Hulse, “Improving software-quality predictions with data sampling and boosting,” *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 6, pp. 1283–1294, Nov. 2009.

[32] J. Van Hulse and T. M. Khoshgoftaar, “Knowledge discovery from imbalanced and noisy data,” *Data Knowl. Eng.*, vol. 68, no. 12, pp. 1513–1542, Dec. 2009.

[33] D. Gamberger, N. Lavraě, and C. Grošelj, “Experiments with noise filtering in a medical domain,” in *Proc. 16th Int. Conf. Mach. Learn.*, San Francisco, CA, 1999, pp. 143–151.

[34] C. E. Brodley and M. A. Friedl, “Identifying mislabeled training data,” *J. Artif. Intell. Res.*, vol. 11, pp. 131–167, 1999.

[35] J. Van Hulse and T. M. Khoshgoftaar, "Class noise detection using frequent itemsets," *Intell. Data Anal.*, vol. 10, no. 6, pp. 487–507, Dec. 2006.

[36] C. M. Teng, "Correcting noisy data," in *Proc. 16th Int. Conf. Mach. Learn.*, 1999, pp. 239–248.

[37] J. Van Hulse, T. M. Khoshgoftaar, C. Seiffert, and L. Zhao, "The multiple imputation quantitative noise corrector," *Intell. Data Anal.*, vol. 11, no. 3, pp. 254–263, Aug. 2007.

[38] D. Anyfantis, M. Karagiannopoulos, S. Kotsiantis, and P. Pintelas, "Robustness of learning techniques in handling class noise in imbalanced datasets," in *Artificial Intelligence and Innovations 2007: From Theory to Applications*, vol. 247. Boston, MA: Springer-Verlag, 2007, ser. IFIP International Federation for Information Processing.

[39] A. Asuncion and D. Newman, UCI machine learning repository, 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[40] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques,* 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.

[41] 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, no. 1, pp. 315–354, Jul. 2003.

[42] N. Seliya, T. M. Khoshgoftaar, and J. Van Hulse, "A study on the relationships of classifier performance metrics," in *Proc. 21st IEEE ICTAI*, Newark, NJ, Nov. 2009, pp. 59–66.

[43] T. M. Khoshgoftaar, C. Seiffert, J. Van Hulse, A. Napolitano, and A. Folleco, "Learning with limited minority class data," in *Proc. 6th ICMLA*, Cincinati, OH, 2007, pp. 348–353.

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

[45] *SAS Institute, SAS/STAT User's Guide*, SAS Institute, Inc., Chicago, IL, 2004.

Taghi M. Khoshgoftaar (M'86) received the Ph.D. degree from Virginia Polytechnic Institute and State University, Blacksburg.

He is currently a Professor with the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, and the Director of the Data Mining and Machine Learning Laboratory. His research interests are in software engineering, software reliability and quality engineering, computational intelligence, data mining, machine learning, and statistical modeling. He has published more than 400 refereed papers in these areas.

Dr. 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 and the General Chair of the International Conference on Software Engineering and Knowledge Engineering in 2008 and 2009, respectively. He is the Program Chair of the IEEE International Conference on Machine Learning and Applications (2010). He has served on technical program committees of various international conferences, symposia, and workshops. Also, he has served as North American Editor of the *Software Quality Journal*. He was on the editorial boards of *Multimedia Tools and Applications* and *Empirical Software Engineering* and is on the editorial boards of the *Software Quality Journal*, the *Journal of Software Engineering and Knowledge Engineering*, the *Journal of Fuzzy Systems*, *Knowledge and Information Systems*, and *Social Network Analysis and Mining*.

Jason Van Hulse (M'05) received the B.S. degree in mathematics from the University at Albany, State University of New York, Albany, 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 the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, in 2007.

Since 2000, he has been with First Data Corporation, Atlanta, GA, where he has worked in the data mining and predictive modeling fields and 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. and Ph.D. degrees in computer science from Florida Atlantic University, Boca Raton, in 2006 and 2009, respectively.

He is currently with Florida Atlantic University. His research interests include data mining and machine learning, evolutionary computation, and artificial intelligence.