Data & Knowledge Engineering 68 (2009) 1513–1542

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Knowledge discovery from imbalanced and noisy data

Jason Van Hulse, Taghi Khoshgoftaar *

Empirical Software Engineering Laboratory, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, United States

ARTICLE INFO

Article history:
Received 30 January 2009
Received in revised form 4 August 2009
Accepted 4 August 2009
Available online 23 August 2009

Keywords:
Data sampling
Class noise
Labeling errors
Class imbalance
Skewed class distribution

ABSTRACT

Class imbalance and labeling errors present significant challenges to data mining and knowledge discovery applications. Some previous work has discussed these important topics, however the relationship between these two issues has not received enough attention. Further, much of the previous work in this domain is fragmented and contradictory, leading to serious questions regarding the reliability and validity of the empirical conclusions. In response to these issues, we present a comprehensive suite of experiments carefully designed to provide conclusive, reliable, and significant results on the problem of learning from noisy and imbalanced data. Noise is shown to significantly impact all of the learners considered in this work, and a particularly important factor is the class in which the noise is located (which, as discussed throughout this work, has very important implications to noise handling). The impacts of noise, however, vary dramatically depending on the learning algorithm and simple algorithms such as naïve Bayes and nearest neighbor learners are often more robust than more complex learners such as support vector machines or random forests. Sampling techniques, which are often used to alleviate the adverse impacts of imbalanced data, are shown to improve the performance of learners built from noisy and imbalanced data. In particular, simple sampling techniques such as random undersampling are generally the most effective.

1. Introduction

Building data mining models using unreliable or abnormal datasets presents a significant challenge to classifier construction. Regardless of the strength of a particular classification algorithm, learning from poor quality data will result in sub-optimal performance. In a classification problem, for example, numerous studies have shown that the presence of errors in the training dataset lowers the predictive accuracy of a learner on test data [27,46,50]. There are many different dimensions of data quality, including class noise or labeling errors [7,29], attribute noise [42,50], and missing values [32].

Another commonly-encountered challenge in data mining applications is the occurrence of class imbalance. A dataset used for binary classification tasks is *imbalanced* if the number of positive examples is unequal (and typically less than) the number of negative examples (this work only considers binary classification problems). Positive examples are typically defined as the minority class, while negative examples belong to the majority class. While class imbalance is not necessarily a data quality issue, it can at times be related. For example, when constructing datasets for use in data mining analysis, it is common in some application domains for the observations to be given the negative class by default unless some event occurs, in which case the example is labeled as positive. If the recording mechanism is faulty or if human error is involved in recording the occurrence of the positive class, there may be very few records of the event, and hence few examples in the training dataset are labeled as positive (even though other positive class examples exist, they may go unrecognized and are

* Corresponding author. Tel.: +1 561 297 3994, fax: +1 561 297 2800.
E-mail address: taghi@cse.fau.edu (T. Khoshgoftaar).

0169-023X/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2009.08.005

mistakenly labeled as negative). In other words, data quality issues may in some situations actually cause class imbalance (or at least make the problem more severe).

Consider, for example, the domain of software quality classification, where data mining techniques have been successfully applied to the problem of classifying faulty program modules [24,26]. The observations in a software quality classification dataset are program modules, and the attributes are measurements of the program module characteristics. These measurements consist of code metrics and process metrics. Code metrics, which measure features related to the program module such as the *number of lines of code*, *number of unique operands*, or *number of branches*, are generally recorded by software tools. Process metrics, on the other hand, may be reported by people working on the project and therefore tend to be error prone. In software quality classification, the dependent variable indicates if the program module contained more than γ faults or bugs, where γ is set by a practitioner or domain expert depending on the objective of the analysis. A module is given a class label of *fault-prone* if it contained more than γ faults, while it is labeled *not fault-prone* if it contained no more than γ faults. The class label is a process metric, with the responsibility of recording the faults in the modules often falling on the software engineer responsible for writing the original code, and is highly likely to be inaccurate and undependable. There is often a strong disincentive to accurately report software faults, since a large occurrence of software faults may highlight poor performance of the software development team. In particular, it is more likely that examples that should be labeled *fault-prone* are (accidentally or intentionally) mislabeled *not fault-prone*.

Inaccurate reporting of software fault data has two important impacts on the training dataset used for data mining in software quality classification. First, there are fewer *fault-prone* examples in the dataset, contributing to the class imbalance (software quality classification datasets typically favor the *not fault-prone* class). Second, labeling errors may not be uniformly distributed in both classes - in fact, it is more likely that examples are mistakenly labeled as *not fault-prone* when they should have been *fault-prone*, and it is relatively rare that *fault-prone* examples were mistakenly labeled. A software engineer is unlikely to mistakenly report a module with no faults as *fault-prone*, and further, it is possible that a module contains a bug but the error has not yet been recognized. Consideration of the distribution of labeling errors among the two classes is an important component of this work that to our knowledge is unique.

Although this example is specific to software engineering, this phenomenon is common in many other application domains, such as network intrusion detection or fraud detection. We further contend that the issues of data quality [13] and class imbalance [52] significantly impact the reliability of data mining models in real-world applications, and therefore deserve careful consideration and experimental evaluation. The primary objective of our experiments is to address these two issues, specifically examining class noise in the context of imbalanced data.

1.1. Research objectives

The motivation behind the experiments conducted in this work is as follows. Class imbalance is an issue encountered by data mining practitioners in a wide variety of fields, and the use of sampling techniques to improve classification performance has received significant attention in related work [11,20,46]. At the same time, real-world data often suffers from data quality issues, specifically class noise. While the practitioner may suspect that there are data quality issues, the extent and severity of the issues are typically unknown. Without full awareness of the extent of class noise in the data, the practitioner may attempt to sample the imbalanced data prior to constructing classification models for the purposes of obtaining an altered class distribution (and often without regard to the issue of class noise).

Another perspective for our motivation comes from considering related work regarding class noise. A number of studies have concluded that classifiers will be adversely impacted by class noise [6,50], however, the impact of class noise in imbalanced datasets has received relatively little attention. Although some research has discussed this issue [46,30,2], comprehensive empirical analysis is still lacking.

Given these motivations, a number of interesting research questions arise that are of great interest to both researchers and practitioners:

- Q_1 : What is the impact of class noise on learners constructed using skewed datasets?
- Q_2 : Is the class in which the noise is located significant?
- Q_3 : Are the effects of class noise in imbalanced datasets uniform across different learning algorithms?
- Q_4 : How do sampling procedures, which are often used to alleviate class imbalance, perform in the presence of class noise?
- Q_5 : Do different sampling techniques work better with different levels of noise and with noise represented in different proportions in the two classes?
- Q_6 : What guidance can be given to improve the commonly-used classification filters[1] in the presence of class imbalance?

Therefore, the primary objectives of this work are to explore the impact of class noise when learning from data with a skewed class distribution and to investigate the effectiveness of commonly-used data sampling techniques.

[1] Classification filters, which are used as noise filtering mechanisms, are applied to a dataset before classifier construction. Examples misclassified by the classification filter are deemed noisy and eliminated from the dataset, and the filtered dataset is then used for building the classifier. Classification filters attempt to improve performance by detecting and removing noisy examples from the dataset before classifier construction.

1.2. Experimental dimensions

In addressing these research questions, this work provides significant contributions to the disciplines of data quality and learning from imbalanced data. Unique and significant aspects of these experiments are:

- We utilize a distinctive noise simulation methodology as described in Section 3. Starting with five real-world datasets from the software engineering domain exhibiting skewed class distributions, a learnable subconcept is extracted and class noise is injected relative to this subconcept. In addition, two larger, publicly available datasets from the UCI repository [1] are also injected with class noise.
- Class noise is injected into the extracted datasets using two parameters, the overall level of class noise (Λ) and the percentage of class noise resulting from the corruption of instances from the true the minority class (Ψ). Noise injection occurs by corrupting pre-existing instances in the datasets as opposed to adding new, noisy instances. Essentially, our noise model consists of two parameters (Λ, Ψ) where Λ determines how many instances are corrupted with noise and Ψ determines the relative proportions of noise in each of the two classes. As will be shown, Ψ is the critical parameter of our simulation that has, to our knowledge, never been explored in previous work.
- We utilize 11 different classifiers (see the Appendix Section A for more details) of the types commonly-used in data mining applications, including decision trees (denoted C4.5D and C4.5N), neural networks (denoted MLP and RBF), logistic regression (LR), rule-based learners (RIPPER), nearest neighbor learners (2NN and 5NN), support vector machines (SVM), naïve Bayes (NB), and random forests (RF).
- Seven different sampling techniques-random undersampling (RUS), random oversampling (ROS), SMOTE (SM), borderline-SMOTE (BSM), one-sided selection (OSS), cluster-based oversampling (CBOS), and Wilson's Editing (WE) – are considered in our experimentation. All of these techniques have been used in previous research to alleviate class imbalance. Additional details regarding these sampling techniques is provided in the Appendix Section B.
- The noise injection process is repeated multiple times for each dataset, leading to the construction of over two million classifiers in these experiments.
- Results are reported using two different performance measures: the area under the ROC curve (AUC) and Kolmogorov–Smirnov statistic (KS). Both measure the general ability of the learner to separate positive and negative class examples, and in particular do not suffer from the well-known issues associated with evaluating learners using the overall misclassification rate [35].
- All results include an analysis of their statistical significance using analysis of variance (ANOVA) techniques. ANOVA is useful in settings where numerous experimental factors are being compared, and a researcher would like to understand the impacts of the factors on the performance measure. Without using ANOVA techniques, it is more difficult to rigorously evaluate the impact of different factors on the outcome.
- Our research group strongly advocates the use of robust, well-designed, and statistically valid experimentation to evaluate the relative effectiveness of various techniques and the impacts of different experimental factors on the procedures of interest. There has been a significant amount of work in data mining related to the proposal of different techniques, but the relative lack of systematic comparisons of these techniques has provided data mining researchers and practitioners little guidance on what does and does not work. We believe that this work represents a significant step forward to empirically understanding class noise and imbalance, and their impacts on learning.

This work significantly expands upon previously presented results [44] that relate to learning from noisy and imbalanced data. For example, this previous work did not consider the impacts of data sampling, considered only a single performance metric (AUC), and due to severe space restrictions, only limited results were presented. In addition, this work considers two larger datasets from the UCI repository.

1.3. Paper organization

The remainder of this work is organized as follows. Related work on imbalance and noise handling is provided in Section 2. Section 3 gives a detailed description of our experimental design. The experimental results are presented in Sections 4 and 5. A thorough discussion of the empirical conclusions, specifically addressing our research questions, and threats to empirical validity, are provided in Section 6. Conclusions and future work are provided in Section 7. The Appendix includes a brief description of the learning algorithms and sampling techniques used in this work.

2. Related work

Previously published work relevant to our study is presented in this section. There has been much previous work related to both class imbalance and class noise, however limited attention has been given to the problem of learning from imbalanced and noisy data. Section 2.1 considers noise in classification problems while Section 2.2 covers the impact of class imbalance on learning and methods for handling imbalance, a topic which has seen substantial attention. Finally Section 2.3 discusses the unique contributions of this work.

2.1. Noise handling

An investigation of both class and attribute noise was conducted by Zhu and Wu [50] which concluded that the presence of both class and attribute noise can be harmful to a classifier, with the former having the more severe impact. The effect of noisy data in the context of cost sensitive learning was also analyzed by Zhu and Wu [51]. The authors concluded that a classifier built on noisy data will increase the average cost of misclassification when the classifier is applied to previously unseen instances (i.e., test data). A recent study [15] compared the performance of numerous classifiers in the context of noisy data and found that robustness to noise varies greatly depending on the learning algorithm. Li et al. [31] proposed a fuzzy relevance vector machine to deal with the problems of imbalance and noise.

Some studies have considered noise handling as part of the data preprocessing steps undertaken before the construction of a classifier. One of the most common procedures, the *classification filter* [6,16], uses an n-fold cross validation procedure, where a learner built on $n-1$ of the partitions is used to classify the data in the holdout partition. Instances in the holdout partition that are misclassified by a base classifier are determined to contain class noise. The procedure is repeated n times so that each partition is used as holdout data once.

It is possible that the learner selected for use in the classification filter does not have an appropriate bias and therefore misclassifies instances that are correctly labeled but are simply difficult for the learner to classify. To counteract this possibility, multiple classifiers can be combined into an *ensemble filter*. Those instances that are misclassified by at least a minimum number of learners are considered mislabeled. A *consensus* filter, for example, requires that all of the classifiers mislabel an instance for it to be considered noise, while a *majority* filter requires misclassification by a majority of the learners. Our research group recently provided a detailed examination of ensemble filtering [29] using 25 different classification techniques.

The rule-based classification model (RBCM) [28] was proposed as a noise filter in our recent work [27]. RBCM uses Boolean rules derived from the most significant attributes as determined by the Kolmogorov–Smirnov (KS) test. Critical values are calculated for each significant attribute using the KS statistic. Boolean rules are formed using the critical values for each significant attribute, and rules are labeled as belonging to one class or another based on the number of '1' and '0' bits in the binary encoding of the rules. Instances that satisfy a rule of the opposite class are labeled as noise. Numerous other works have investigated data noise in data mining and classification problems [41,36,39], however none specifically consider noise in the context of class imbalance.

2.2. Methods for handling class imbalance

Sampling techniques have also been studied in previous work, but not from the perspective of class noise. Since they are simple to implement and have existed for the longest time, random undersampling and random oversampling seem to have received the most attention. Drummond and Holte [11] found using C4.5 that majority undersampling is more effective at dealing with the imbalance problem and that minority oversampling often produces little or no change in performance. Maloof [33] performed experiments using both naïve Bayes and C5.0 (the commercial successor to C4.5) and found that undersampling and oversampling produce classifiers that are roughly equivalent to each other. Japkowicz [19] used neural networks and artificially generated one-dimensional data and found that oversampling and undersampling perform similarly on smaller datasets, while undersampling performs better on larger datasets. A later study by Japkowicz and Stephan [21] found that for C5.0, oversampling performs better than undersampling. Weiss and Provost [47] do not compare sampling techniques, but use random undersampling and C4.5 to evaluate the effect of class distribution on classifier performance. They concluded that there is no universally optimal distribution for ideal performance, and that it depends on the domain.

Chawla et al. [10] used C4.5, naïve Bayes, and RIPPER to test their proposed SMOTE technique against random undersampling, one-sided selection, and varying the classifier's decision threshold. They found SMOTE to generally outperform all of the other methods. Han et al. [17] evaluated borderline-SMOTE against regular SMOTE and random oversampling using C4.5 and found borderline-SMOTE to perform the best, followed by SMOTE and then random oversampling. Jo and Japkowicz [22] used both C4.5 and neural networks to compare cluster-based oversampling, random oversampling, random undersampling, pruning small disjuncts from the training data, and an oversampling technique that generates new artificial examples by adding random Gaussian noise to the data. Their experiments found cluster-based oversampling to perform the best out of all the methods. Prati et al. [34] used C4.5 to perform experiments comparing random oversampling, SMOTE, the combination of oversampling with SMOTE and undersampling using Tomek links, and the combination oversampling with SMOTE and undersampling with Wilson's editing. They found the two combinations to work well, though often random oversampling was able to achieve comparable results. They also assert that oversampling generally performs better than undersampling. Batista et al. [3] compares ten different sampling techniques using 13 UCI datasets, concluding that oversampling outperforms undersampling, and two hybrid procedures SMOTE + Tomek and SMOTE + ENN, are particularly useful for datasets with a small number of positive examples. Given all of the previous research and conflicting results, there has been little consensus reached on which sampling techniques work best and under what conditions. Our recent work [43] compared seven sampling techniques using 35 benchmark datasets, and demonstrated that random undersampling performs very well.

A number of studies have dealt with class imbalance from the perspective of noise reduction. Kubat and Matwin [30] proposed a technique called *one-sided selection*, which attempts to intelligently undersample the majority class by removing

majority class examples that are considered either redundant or noisy. *Wilson's editing* [2] uses the kNN technique with $k = 3$ to classify each example in the training set using all the remaining examples, and removes those majority class examples that are misclassified. Other procedures for handling class imbalance, such as SMOTE, cluster-based oversampling, and random sampling do not address the issue of noise.

Cost sensitive learning is closely related to the problem of class imbalance. Weiss et al. [45] compare the performance of oversampling, undersampling and cost sensitive learning when dealing with data that has both an imbalanced class distribution and unequal error costs. A wrapper-based sampling approach to minimizing misclassification cost is evaluated by Chawla et al. [9]. Boosting algorithms are also commonly-used to alleviate the impacts of class imbalance [12,38,23]. Cao et al. [8] proposed techniques for discovering rare but significant impact-targeted activity patterns in imbalanced activity data. Future research could consider detailed experimentation on the impact of noise on cost sensitive learning, boosting, and unsupervised learning tasks such as association rule mining, as this work focuses on data sampling for supervised learning applications.

2.3. Summary

In one of the few works that address noise in an imbalanced data environment, Weiss [46] argues that "noise has a greater impact on rare cases than on common cases" and that relatively high levels of class noise were required to cause rare cases and small disjuncts to be error prone. Weiss, however, utilized simulated data with binary attribute values, a single learner, C4.5, and did not consider the very important factor of which class contains noise (consideration of this factor is unique to our work). Further, the objective of Weiss' work [46] was to analyze various factors (e.g., attribute noise, missing values, training set size, and class noise), whereas this work concentrates solely on class noise and its relation to imbalanced data. Another differentiating factor is that we consider many different learners and examine the utilization of sampling techniques in the context of noisy and imbalanced data. Our previous, preliminary work [44] did consider the problem of learning from noisy and imbalanced data, however, sampling techniques were not considered and fewer datasets were used. In addition, this work dramatically expands upon the results we have presented previously [44].

Therefore, while class noise has received attention in data mining literature, to our knowledge no previous work has analyzed labeling errors in imbalanced data as systematically as in this work. Similarly, though numerous strategies for coping with skewed class distributions have been proposed and analyzed, the impact of class noise in an imbalanced dataset has not been adequately addressed. Noise handling in the presence of class imbalance has therefore received relatively little attention in related work. Most experiments with the classification and ensemble filters have paid no attention to class distributions and the need for special attention when one class is relatively rare. The important connections between this and previous work by researchers in the domain of imbalance is discussed in Section 5. A contention of numerous researchers [46,20,21,3] is that class imbalance, or relative rarity, is not by itself a cause for concern. Instead, other factors such as small disjuncts, overlapping classes, absolute rarity (i.e., a too few positive cases), or concept complexity, in combination with imbalance, cause the poor performance obtained by classifiers. This work shows that a related concept, class noise, can also harm classifiers trained from imbalanced data. Further, class noise can be one of the causes of all of the previously mentioned factors. In this respect, our work has a significant connection with previous work in the domain of imbalance.

3. Experimental design

3.1. Datasets

The software measurement data used for our study are from five NASA software projects, CM1, MW1, PC1, KC1, and KC3 [28]. The data was made available through the Metrics Data Program at NASA, and included software measurement data and associated error (fault) data collected at the function, subroutine, or method level. Hence, for the software systems, a function, subroutine, or method is considered as a software module or an instance in the dataset.

The CM1 project, written in C, is a science instrument system used for mission measurements. The MW1 project, written in C, is the software from a zero gravity experiment related to combustion. The PC1 project, written in C, is flight software from an earth orbiting satellite. The KC1 and KC3 projects are different components of the same mission project, however they constitute different personnel and have no software overlap other than minimal third-party software libraries. The KC1 system, written in C++, is a software component of a large ground system. The KC3 system, written in JAVA, is software developed for collection, processing, and delivery of satellite metadata.

The fault data collected for the software systems represent faults detected during software development. Each module was characterized by 13 software measurements [14] including basic measures such as the number of lines of code, number of branches, and the number of operators and complexity measures such as the cyclomatic complexity, essential complexity, and design complexity. The quality of the modules is described by their class labels. A program module was considered *fault-prone* if it had one or more software faults, and *not fault-prone* otherwise. The positive (or minority) class in software measurement datasets is the *fault-prone* class, while the *not fault-prone* class is the negative or majority class.

Two datasets from the UCI repository, Optdigits and Nursery, were also utilized in our experiments. The Optdigits dataset, which has 64 independent variables, can be used to train models to recognize handwritten digits from zero to nine. The digit

Table 1
Dataset characteristics.

	Original			Transformed		
Data	# Positive	# Negative	% Positive	# Positive	# Negative	% Positive
CM1	48	457	9.50	39	277	12.34
KC1	325	1782	15.42	271	1093	19.87
KC3	43	415	9.39	38	264	12.58
MW1	31	372	7.69	20	291	6.43
PC1	76	1031	6.87	53	650	7.54
Optdigits	554	5065	9.86			
Nursery	328	12636	2.53			

'8' was selected as the positive class, and the remaining digits constitute the negative class. The Nursery dataset, which has eight attributes, contains information for ranking nursery schools. The dataset originally contained five classes, one of which ('very recommended') was selected in this work to be the positive class, while the remaining class values were grouped together to form the negative class. Both Optdigits and Nursery are larger than the five software measurement datasets, and Nursery is the most imbalanced of the seven datasets considered in this work. These datasets therefore provide a wide range of sizes, imbalance levels, and application domains. Throughout this work, the five software engineering datasets will be denoted as the SE datasets, while Optdigits and Nursery will collectively be referred to as the UCI datasets. Note that all of the datasets considered in this work have a binary class, and consideration of multi-class domains is left to future work.

3.2. Data preprocessing

One way to conduct the required simulation is to inject class noise into unprocessed, real-world datasets. This can be problematic, however, because data from real-world application domains may already contain class noise.[2] Therefore, our approach with the SE datasets was to first create a subset $D^c \subset D$ for each of the five datasets D such that the target concept encapsulated in D^c is learnable. In other words, learners constructed using D^c should achieve perfect or near-perfect classification accuracy. Once D^c is derived, noise injection can be performed relative to this new dataset. In other words, class noise is injected relative to the target concept in D^c.

To create the datasets D^c, the RBCM-based filter [28], proposed in previous work by our research group and described briefly in Section 2, was used. Numerous classifiers were constructed using the filtered datasets D^c with few or no misclassified examples. Therefore, a set of transformed datasets have been created with a learnable target concept, with the examples in the dataset conforming to that concept. Table 1 shows the characteristics of the original datasets in columns 2, 3, and 4, and the characteristics of the transformed datasets in columns 5, 6, and 7. The SE datasets used in our study have the percentage of minority examples ranging from 6.43% to 19.87%, and are therefore imbalanced with respect to the class.

For the Nursery and Optdigits datasets, unlike the five SE datasets, cleansing was not undertaken prior to noise injection. Our previous experience has shown that these two datasets are relatively free from class noise, and hence preprocessing was unnecessary. The different treatment prior to noise injection for the UCI datasets compared to the SE datasets greatly expands the scope and applicability of our results. Further, since the characteristics of these two sets of data are so different, empirical results are presented separately in later sections of this work.

3.3. Noise injection

After the datasets have been prepared, class noise must be injected based on two considerations. First, how much noise should be injected into the dataset? Second, in what proportions should noise be injected into each class? The first consideration is easily justified – based on the data generation process, datasets from different domains may have different levels of noise. The second idea is more subtle and may require some justification. We contend that noise may be inherent in a dataset in varying degrees among the two classes and that it is too simplistic to assume, for example, that in all situations, half of the noisy examples are from the positive class and half are from the negative class. The introduction of this work provided an example specific to the software engineering domain where noise is not uniformly inherent in both classes, and we believe this situation is common in other domains as well. Therefore, this work considers the impact of varying the class distribution of noise, which has been overlooked in previous studies.

Our experiments consider two noise simulation parameters that address these ideas. The first governs the level of class noise in the dataset and is denoted $\Lambda(\Lambda = 10, 20, 30, 40, \text{ and } 50\%)$. The second parameter is the percentage of class noise resulting from the corruption of instances from the true minority class, denoted $\Psi(\Psi = 0, 25, 50, 75, \text{ and } 100\%)$. Let D^n denote D^c after the injection of class noise. Define

[2] Another approach is to create simulated datasets using a generative model, as has been done in other studies. The use of simulated data has its advantages and disadvantages, though we chose not to do this in our experiments. Our main concern is that simulated data may not provide a realistic representation of performance, and we believe that utilizing our strategy for dataset creation avoids this difficulty.

$P = \{x \in D^c | x \text{ is from the positive class}\}$
$N = \{x \in D^c | x \text{ is from the negative class}\}$
$E = \{x \in D^n | x \text{ contains class noise}\}$
$S^P = |P|, S^E = |E|.$

In our simulation, the total number of noisy examples corrupted in each dataset is given by $2 \times \Lambda \times S^P$. For example, the KC1 dataset has 271 P and 1093 N examples. At a 30% noise level ($\Lambda = 30\%$), KC1 will be corrupted in $2 \times 30\% \times 271 = 163$ instances. Which 163 instances should be selected for corruption? This is controlled by the second parameter Ψ: randomly select $2 \times \Lambda \times S^P \times \Psi$ examples from P and corrupt the class to N, and randomly select $2 \times \Lambda \times S^P \times (1 - \Psi)$ examples from N and corrupt to class P. Continuing the example with KC1 and 30% noise, suppose Ψ is set to 75%. Then $163 \times 75\% = 122$ P examples are corrupted to class N, and $163 \times (1 - 75\%) = 41N$ examples are corrupted to class P.

Given this construction, $S^E = 2 \times \Lambda \times S^P$. With $\Psi = 50\%$, the injection of class noise is identical to the proportional random corruption (PRC) procedure [51]. With $\Psi = 50\%$, the percentage of positive class examples before and after noise injection is equal. By varying the parameter Ψ, we are able to analyze the impact of varying the degree of noise coming from the positive class on the learners. Note that to the best of our knowledge, previous work has not considered varying the parameter Ψ, which as will be shown, is critical to understanding the impact of class noise.

Let $P \to N$ be the set of examples whose class is corrupted from positive to negative, while $N \to P$ are the examples where the class is corrupted from negative to positive:

$P \to N = \{x \in P | \text{the class of } x \text{ is corrupted to } N\}$
$N \to P = \{x \in N | \text{the class of } x \text{ is corrupted to } P\}.$

Then $E = [P \to N] \cup [N \to P]$, i.e., the set of noisy examples consists of those whose class is corrupted from positive to negative, and those whose class is corrupted from negative to positive. Note that the parameter Ψ defines the relative proportion of $P \to N$ and $N \to P$ examples in E. With $\Psi = 50\%, |P \to N| = |N \to P| = \Lambda_0 \times S^P$. If $\Psi = 0\%$, then $P \to N = \emptyset$ and $E = N \to P$ with $|N \to P| = 2 \times \Lambda_0 \times S^P$.

The parameter Λ controls the level of noise in the dataset, although $\Lambda = 10\%$, for example, does not mean that 10% of the total examples are corrupted. Since the data is highly imbalanced, this level of corruption would dramatically impact the minority class. Instead Λ is the level of noise relative to the size of the minority class.

3.4. Dataset noise characteristics

Table 2 presents the noise corruption statistics for the KC1 dataset, which prior to noise injection has 271 P and 1093 N examples. With $\Lambda = 10\%, 54 (= 2 \times 10\% \times 271)$ examples are injected with noise. Since KC1 has 1364 total examples, 54 noisy instances is about 4.0% of the total dataset (columns 3 and 4 in Table 2). With $\Psi = 0\%, P \to N = \emptyset$ (column 5) while $N \to P$ has 54 examples (column 8). Note that 100% of the positive examples were retained in columns 6 and 7 (since $P \to N = \emptyset$), while approximately 5% of the negative class was corrupted (columns 9 and 10). In the final dataset after the corruption process was completed (the last five columns of Table 2), there are 325 positive examples, of which 16.7% or 54 are noisy (the true class of these instances is negative, even though they are labeled positive).

The last row of the table, $\Lambda = 50\%$ and $\Psi = 100\%$, results in a dataset with no positive examples, and hence was removed from the experiments. Note that for a fixed value of Λ, increasing Ψ results in more noise coming from the positive class, i.e., $P \to N$ increasingly dominates $N \to P$. Fewer examples from the original positive class remain after corruption (the values in the columns labeled 'Clean P Left' are decreasing for a fixed value of Λ), and the positive class concept is obscured by examples that are mislabeled as negative but should be positive. With $\Lambda = 40\%$ and $\Psi = 100\%$, for example, of the 271 examples from the original positive class, 217 are corrupted to the negative class. Note that after corruption of KC1, the percentage of positive examples ranges from 4.0% to 39.7%.

3.5. Performance measures

Two measures of classifier performance are presented in this work, the area under the ROC curve (AUC) and the Kolmogorov–Smirnov (KS) [18] statistic. KS measures the maximum difference between the cumulative distribution functions of the predicted probabilities of examples in each class. In other words, KS measures how far apart the distribution functions are, and hence how well the learner is able to separate positive and negative class examples. Denote the classes as N (negative) and P (positive). Each instance x is assigned by the learner a probability of class P membership $p(P|x) \in [0, 1]$ (assuming the learner outputs on a probability scale, which is true of all learners in this study). The true class P and N distribution functions $F_P(t)$ and $F_N(t)$ are unknown, but are estimated by the proportion of class P or N examples with $p(P|x) \leqslant t, 0 \leqslant t \leqslant 1$:

$$\widehat{F}_P(t) = \frac{\#\text{class } P \text{ instances with } p(P|x) \leqslant t}{\#\text{class } P \text{ instances}},$$
$$\widehat{F}_N(t) = \frac{\#\text{class } N \text{ instances with } p(P|x) \leqslant t}{\#\text{class } N \text{ instances}}.$$

Table 2
Noise Simulation Details for the KC1 Dataset.

				Corruption process by class						Statistics for corrupted dataset				
		Injected noise			Clean P Left			Clean N Left		#	% P	#	% N	
Λ	Ψ (%)	#	%	$P \rightarrow N$	#	%	$N \rightarrow P$	#	%	P	Noisy	N	Noisy	% P
10	0	54	4.0	0	271	100.0	54	1039	95.0	325	16.7	1039	0.0	23.8
	25			14	257	95.0	41	1052	96.3	298	13.6	1066	1.3	21.9
	50			27	244	90.0	27	1066	97.5	271	10.0	1093	2.5	19.9
	75			41	230	85.0	14	1079	98.8	244	5.6	1120	3.6	17.9
	100			54	217	80.0	0	1093	100.0	217	0.0	1147	4.7	15.9
20	0	108	7.9	0	271	100.0	108	985	90.1	379	28.6	985	0.0	27.8
	25			27	244	90.0	81	1012	92.6	325	25.0	1039	2.6	23.8
	50			54	217	80.0	54	1039	95.0	271	20.0	1093	5.0	19.9
	75			81	190	70.0	27	1066	97.5	217	12.5	1147	7.1	15.9
	100			108	163	60.0	0	1093	100.0	163	0.0	1201	9.0	11.9
30	0	163	11.9	0	271	100.0	163	930	85.1	434	37.5	930	0.0	31.8
	25			41	230	85.0	122	971	88.8	352	34.6	1012	4.0	25.8
	50			81	190	70.0	81	1012	92.6	271	30.0	1093	7.4	19.9
	75			122	149	55.0	41	1052	96.3	190	21.4	1174	10.4	13.9
	100			163	108	40.0	0	1093	100.0	108	0.0	1256	12.9	7.9
40	0	217	15.9	0	271	100.0	217	876	80.2	488	44.4	876	0.0	35.8
	25			54	217	80.0	163	930	85.1	379	42.9	985	5.5	27.8
	50			108	163	60.0	108	985	90.1	271	40.0	1093	9.9	19.9
	75			163	108	40.0	54	1039	95.0	163	33.3	1201	13.5	11.9
	100			217	54	20.0	0	1093	100.0	54	0.0	1310	16.6	4.0
50	0	271	19.9	0	271	100.0	271	822	75.2	542	50.0	822	0.0	39.7
	25			68	203	75.0	203	890	81.4	407	50.0	958	7.1	29.8
	50			136	136	50.0	136	958	87.6	271	50.0	1093	12.4	19.9
	75			203	68	25.0	68	1025	93.8	136	50.0	1229	16.5	9.9
	100			271	0	0.0	0	1093	0.0	0	–	1364	19.9	0.0

$\widehat{F}_P(t)$ and $\widehat{F}_N(t)$ are the empirical distribution functions of the class P and N examples, respectively. Then the KS statistic is defined as:

$$\mathrm{KS} = \max_{t \in [0,1]} |\widehat{F}_P(t) - \widehat{F}_N(t)|. \tag{1}$$

KS can also be calculated as the maximum difference between the curves generated by the true positive and false positive rates as t changes from 0 to 1. $F_P(t)$ can be further understood as follows:

$$\widehat{F}_P(t) = \frac{\#P \text{ instances with } p(P|x) \leqslant t}{\#P \text{ instances}} = \frac{\#P \text{ misclassified as } N \text{ given } t}{\#P} \tag{2}$$

$$= \frac{\#FN(t)}{\#P} = \mathrm{FNR}(t) = 1 - \mathrm{TPR}(t) \tag{3}$$

where $\#FN(t)$ is the number of false negative instances at decision threshold $t \in [0,1]$ and $\mathrm{FNR}(t)$ is the false negative rate at decision threshold t. Similarly, it can be shown that $\widehat{F}_N(t) = 1 - \mathrm{FPR}(t)$, and it therefore follows that

$$\mathrm{KS} = \max_{t \in [0,1]} |\mathrm{TPR}(t) - \mathrm{FPR}(t)|.$$

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 KS is one (perfect separation), with a minimum of zero (no separation). The KS statistic is a commonly-used performance measure in the domain of credit scoring [40].

The receiver operating characteristic (ROC) curve graphs the true positive rate on the y-axis versus the false positive rate on the x-axis. The ROC curve illustrates the performance of a classifier across the entire range of decision thresholds, and accordingly does not assume any particular misclassification costs or class prior probabilities. For a single numeric measure, the AUC is often used, which gives a general idea of the predictive potential of the classifier. A higher AUC is better, as it indicates that the ROC curve obtains higher true positive and lower false negative rates. We chose to use these two metrics (AUC and KS) because they measure the general ability of the learner to separate the two classes and are independent of the choice of decision threshold.

3.6. Design

This section provides a detailed description of our experimental design. Seven datasets were used in our experiments, and all learners were constructed and evaluated using 10-fold cross validation (CV). When building a learner, nine folds were

Table 3
Summary of notation.

Learners	
5NN	*k*NN learner with 5 nearest neighbors
2NN	*k*NN learner with 2 nearest neighbors
SVM	Support vector machine with parameter 'c' changed from 1 to 5 and 'buildLogisticModels' set to true'
C4.5D	Decision tree using J48 algorithm in WEKA with the default parameters
C4.5N	Decision tree using J48 algorithm with pruning disabled and Laplace smoothing enabled
RF	Random forest classifier
RIPPER	Rule-based learner implemented in JRip algorithm in WEKA
LR	Logistic regression model
MLP	Multilayer perceptron neural network learning algorithm with 'hiddenLayers' parameter changed to '3' and validationSetSize changed to '10'
RBF	Radial basis function network implemented as 'RBF Network' in WEKA with 'numClusters' set to 10
NB	Naïve Bayes classifier
Sampling techniques	
RUS	Random undersampling
ROS	Random oversampling
SM	SMOTE
BSM	Borderline-SMOTE
OSS	One-sided selection
CBOS	Cluster-based oversampling
WE	Wilson's editing
Experimental design	
Ψ	Noise simulation parameter representing the percentage of noise resulting from corruption of instances in the minority class
Λ	Noise simulation parameter representing the total number of instances injected with class noise
SE	Denotes the five software engineering datasets
UCI	Denotes the two datasets from the UCI repository
AUC	Area under the ROC curve, used to measure model performance
KS	Kolmogorov–Smirnov statistic, used to measure model performance
P	Positive class
N	Negative class

used as a training dataset with the remaining fold acting as a test dataset (using CV). Class noise was injected into the training dataset only, using the five different values for Λ and five different values for Ψ, representing the overall level of noise and percentage of class noise resulting from the corruption of instances from the true minority class, respectively. For each of the five SE datasets 10-fold CV was repeated a total of ten times, each time with an independently chosen set of examples for noise corruption according to the different values of Λ and Ψ. For Nursery and Optdigits, 10-fold CV was repeated three times due to the significantly larger size of these datasets and the extensive execution time that ten repetitions would have required. One caveat to this design is that $\Lambda = 50\%$ and $\Psi = 100\%$ could not be accommodated, because this scenario resulted in a dataset with no positive examples. Therefore, this case is excluded.

In total, for the SE datasets, 5 datasets $\times$ 10 runs of 10-fold CV results in 500 uncorrupted training datasets. With Λ = 10, 20, 30, and 40% all five different values of Ψ (0, 25, 50, 75, and 100%) were applied, while for $\Lambda = 50\%$, only Ψ = 0, 25, 50, and 75% were used. Therefore, for each of the 500 uncorrupted datasets, 24 different corruption schemes were used, resulting in $500 \times 24 = 12,000$ corrupted datasets. For each of these training datasets, 17 combinations of sampling technique plus parameter (which includes no sampling) were applied, and 11 learners were constructed on each of these datasets (Sections A and B provide more information on the sampling techniques and learners). Therefore in total, $17 \times 11 \times 12,000 = 2,244,000$ classifiers[3] were built in these experiments for the SE datasets.

For the UCI datasets, 2 datasets $\times$ 3 runs of 10-fold CV results in 60 uncorrupted datasets. 24 corruption schemes were used, resulting in $60 \times 24 = 1440$ corrupted training datasets. Nine combinations[4] of sampling technique plus parameter were used in conjuction with 11 learners, resulting in the construction of $1440 \times 9 \times 11 = 142,560$ classifiers for the two UCI datasets. Experiments were conducted using several high performance computing clusters and a supercomputer with very large memory. Even with these significant hardware resources available, many months of execution time were required to complete the experiments presented in this work, and hence judicious choices regarding parameter selections were required. AUC and KS measures are presented to assess classifier performance because they measure the general ability of the learners to sep-

[3] The actual number of classifiers was slightly less than this because some sampling percentages could not be performed for some of the datasets and noise levels, e.g., when the number of positive examples after corruption was more than 35%, RUS with 35% could not be performed.

[4] Sampling techniques ROS, CBOS, OSS, and BSM were not utilized in conjunction with the UCI datasets. As shown in our empirical results, OSS and CBOS performed very poorly for the SE datasets. SM, ROS, and BSM are all oversampling techniques; SM generally performs better than ROS, and is more commonly-used than BSM. Hence the signficant runtime that would have been required to generate output for CBOS, OSS, ROS, and BSM could be saved as the results would not provide significant value.

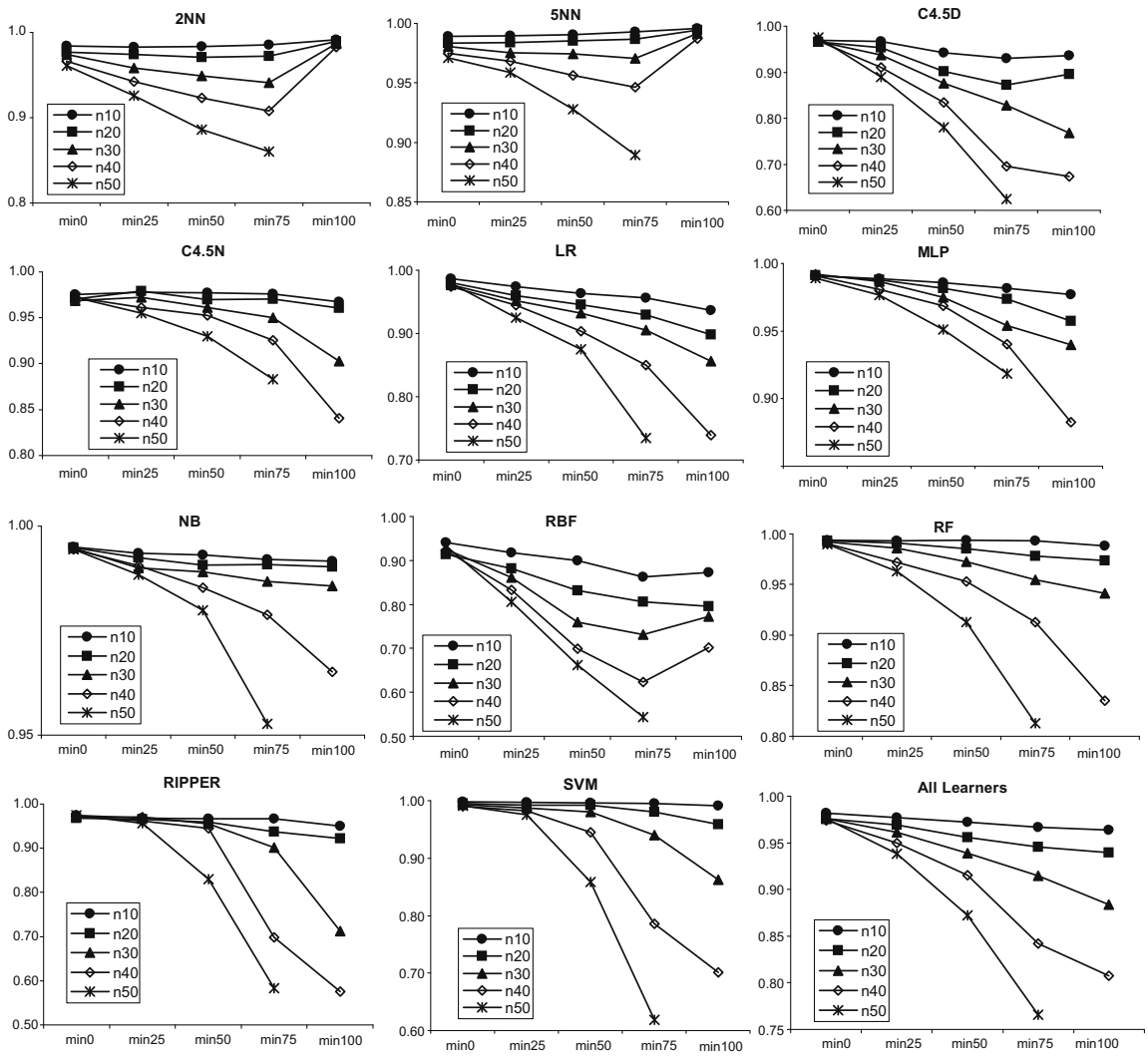

Fig. 1. AUC by minority class noise % (Ψ), 5 SE datasets.

arate the two classes and are independent of the choice of the decision threshold. Table 3 provides a summary of the notation used throughout this work.

Note that all reported results for sampling techniques are optimized over the parameter values for each dataset and learner. For example, for dataset KC1 with $\Psi = 50\%$ and $\Lambda = 20\%$ with classifiers built using SVM, RUS with 35% generated ten AUC values (because there are ten repetitions of 10-fold CV), RUS with 50% generated ten AUC values, and likewise for RUS 65%. If RUS with 35% generated the highest average AUC for KC1, $\Psi = 50\%$ and $\Lambda = 20\%$, then RUS with 35% is reported in the results for RUS for that particular dataset (and in particular, RUS 50% and RUS 65% are not reported). It is possible that given the same dataset with a different learner, for example C4.5D, RUS with 50% might generate the highest average AUC. Generally speaking, 35% often produced the best results in our experiments, which matches with the conclusions of a previous study [25]. Note, however, that analyzing different undersampling or oversampling percentages is not one of the objectives of this study.

The results presented in the remainder of this work generally represent averages of the AUC and KS performance metrics over the five SE datasets or two UCI datasets. Averaging across datasets, as opposed to presenting results for each dataset separately, was performed to enhance readability, reduce redundancy, and for space considerations. The conclusions discussed in this work generally hold for all of the datasets individually. Further, ANOVA analysis, which is used to determine the statistical significance of the findings, does take into account cross-dataset variance. We do, however, present the results separately for the five SE datasets and the two UCI datasets, which gives the reader confidence that the results are generally valid across datasets and application domains.

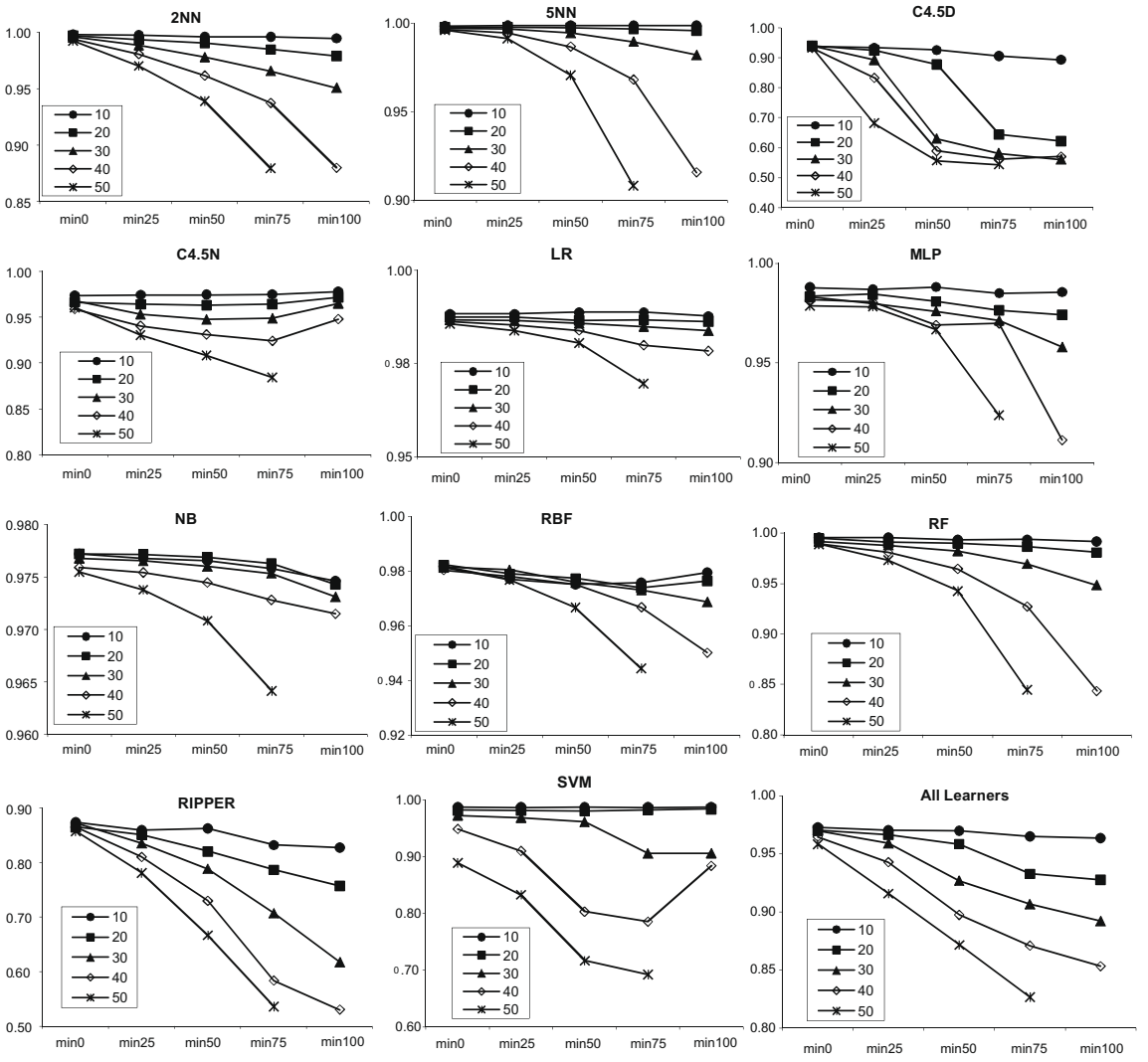

Fig. 2. AUC by minority class noise % (Ψ), 2 UCI datasets.

4. Experimental evaluation I: learning from imbalanced and noisy data

This section explores the performance of the 11 classifiers when learning from imbalanced and noisy data, and varying the levels of the two factors Λ and Ψ. All results in Section 4 do *not* consider sampling. The objective of this section is to analyze the impact of noise and class imbalance on learning, and the effects of sampling are considered in Section 5. We only present the results for the AUC performance measure, as the empirical conclusions obtained by considering the KS are similar.

Fig. 1 displays the AUC for the different values of Λ and Ψ for each learner built without using sampling, averaged over five software engineering (SE) datasets. Fig. 2 displays the same information for the two UCI datasets. The *x*-axis in both figures is Ψ, the percentage of noise coming from the positive class (labeled 'min0', ..., 'min100'), with the *y*-axis representing the AUC. Each level of noise Λ is represented by a different line (labeled 'n10', ..., 'n50'). Note that each learner is presented relative to a different scale to make the trends more visible, so care must be taken when comparing trends for different learners.

Many of the learners display a similar 'fan' pattern. With $\Psi = 0\%$, regardless of the level of noise Λ, the performance of the learners is often very similar (i.e., the five lines are clustered closely together). For most of the learners, the line for $\Lambda = 10\%$ is relatively horizontal, meaning that the AUC does not dramatically decrease. As Λ increases, the slope of the line becomes steeper as Ψ increases for almost all learners. For $\Lambda = 40\%$ or 50% and $\Psi \geqslant 50\%$, the deterioration in AUC is significant for most learners. As anticipated, $\Lambda = 50\%$ results in the worst performance for all learners. There is a strong inverse correlation between Λ and AUC across Ψ – as Λ increases, the performance of the learner decreases, though this impact is more signif-

icant with higher values of Ψ. In addition, increasing the percentage of minority class noise generally decreased classifier performance.

For all of the learners, 50% class noise with $\Psi = 0\%$ (i.e., 0% of the noise resulting from the corruption of instances from the true minority class) does not significantly harm classifier performance, whereas $\Psi = 75\%$ with 50% class noise dramatically impacts the classifiers. Therefore, we conclude that it is both the level of noise and the class in which the noise resides that dictates the impact of labeling errors on classification using imbalanced data. This observation is discussed further in Section 6.

Table 4 presents for each learner (without sampling), the average AUC over five measurement datasets (left-hand side of the table) and two UCI datasets (right-hand-side of the table) for different levels of noise Λ (for each noise level, the AUC is averaged over all values of Ψ). For all learners, the AUC decreases (often dramatically) as the level of noise increases from 10% to 50%. Much of the reduction occurs as Λ increases from 30% to 50%. There is substantial variation in the impact of class noise on the learning algorithms. C4.5 performs better with Laplace smoothing and without pruning compared to the standard C4.5 decision tree algorithm (i.e., C4.5D), even at higher noise levels, contrary to expectations. Ideally, pruning should help avoid overfitting in noisy data, however our results clearly demonstrate that pruning and/or non-smoothed probability estimates may in fact hurt the predictive capability of the decision tree learner. 5NN generally outperforms 2NN, especially at higher noise levels, which might be expected since 5NN considers more nearest neighbors and hence is less likely to be impacted by noise. NB is the most stable learner of the 11 considered in this study. Two of the newer and more sophisticated techniques, RF and SVM, are impacted more by class noise than some of the simpler algorithms such as NB and 5NN.

Table 5 presents the average AUC over the five SE datasets and two UCI datasets for different levels of Ψ (i.e., $P \rightarrow N$-type noise) for each learner (for each level of Ψ, the AUC is averaged over all levels of Λ). The performance of many learners deteriorates significantly as Ψ increases. C4.5D and RIPPER are the two learners which exhibit the most significant and consistent deterioration in performance, while other learners such as NB, and to a lesser extent 2NN, 5NN, and MLP, are relatively unaffected.

Fig. 3 shows the AUC, averaged over all five SE datasets, for each learner (without sampling) for the different levels of Ψ (leftmost figure) and Λ (rightmost figure). For a given learner, if the points are clustered closely together, there is less of an impact on that learner when varying either Ψ or Λ. For example, the AUC for NB is very similar regardless of Ψ or Λ. C4.5N and MLP also perform well as Λ and Ψ vary, while at the opposite extreme, RBF, RIPPER, and C4.5D are all severely impacted. SVM displays an interesting trend, being a very good learner when Λ or Ψ are small, but as either of these parameters increases, the performance of SVM deteriorates.

Table 4
Change in AUC by level of noise Λ.

AUC	5 SE datasets						2 UCI datasets					
	10%	20%	30%	40%	50%	% Change	10%	20%	30%	40%	50%	% Change
2NN	0.986	0.977	0.962	0.945	0.908	−7.85	0.996	0.989	0.976	0.951	0.945	−5.12
5NN	0.992	0.987	0.978	0.967	0.937	−5.53	0.999	0.997	0.992	0.973	0.967	−3.18
C4.5D	0.949	0.918	0.875	0.817	0.817	−13.89	0.920	0.802	0.721	0.698	0.680	−26.12
C4.5N	0.975	0.970	0.951	0.930	0.935	−4.07	0.975	0.966	0.957	0.941	0.921	−5.57
LR	0.964	0.943	0.924	0.882	0.879	−8.82	0.988	0.987	0.986	0.983	0.980	−0.86
MLP	0.985	0.978	0.970	0.953	0.959	−2.64	0.987	0.980	0.974	0.963	0.962	−2.51
NB	0.993	0.992	0.989	0.983	0.979	−1.43	0.976	0.976	0.976	0.974	0.971	−0.53
RBF	0.898	0.846	0.810	0.756	0.736	−18.09	0.978	0.978	0.976	0.970	0.967	−1.07
RF	0.993	0.984	0.969	0.933	0.920	−7.35	0.994	0.989	0.976	0.941	0.938	−5.68
RIPPER	0.965	0.951	0.902	0.830	0.836	−13.37	0.851	0.816	0.765	0.704	0.710	−16.53
SVM	0.995	0.984	0.953	0.881	0.861	−13.51	0.987	0.982	0.942	0.866	0.782	−20.73

Table 5
Change in AUC by level of minority class noise Ψ.

AUC	5 SE datasets						2 UCI datasets					
	0%	25%	50%	75%	100%	% Change	0%	25%	50%	75%	100%	% Change
2NN	0.973	0.957	0.943	0.933	0.988	+1.55	0.995	0.986	0.973	0.953	0.951	−4.47
5NN	0.980	0.975	0.967	0.957	0.992	+1.26	0.997	0.996	0.990	0.972	0.973	−2.42
C4.5D	0.969	0.931	0.867	0.790	0.819	−15.49	0.937	0.853	0.716	0.648	0.662	−29.34
C4.5N	0.972	0.969	0.958	0.941	0.918	−5.54	0.966	0.953	0.945	0.939	0.966	0.00
LR	0.980	0.951	0.924	0.875	0.857	−12.48	0.987	0.986	0.985	0.982	0.984	−0.28
MLP	0.991	0.984	0.972	0.954	0.939	−5.21	0.983	0.982	0.976	0.965	0.957	−2.62
NB	0.995	0.991	0.988	0.980	0.983	−1.16	0.977	0.976	0.975	0.973	0.973	−0.32
RBF	0.927	0.859	0.770	0.713	0.786	−15.28	0.981	0.978	0.974	0.967	0.969	−1.30
RF	0.992	0.981	0.964	0.931	0.935	−5.77	0.992	0.986	0.975	0.944	0.941	−5.15
RIPPER	0.972	0.964	0.931	0.817	0.790	−18.73	0.867	0.827	0.774	0.689	0.683	−21.17
SVM	0.993	0.987	0.954	0.864	0.879	−11.57	0.956	0.936	0.889	0.870	0.940	−1.63

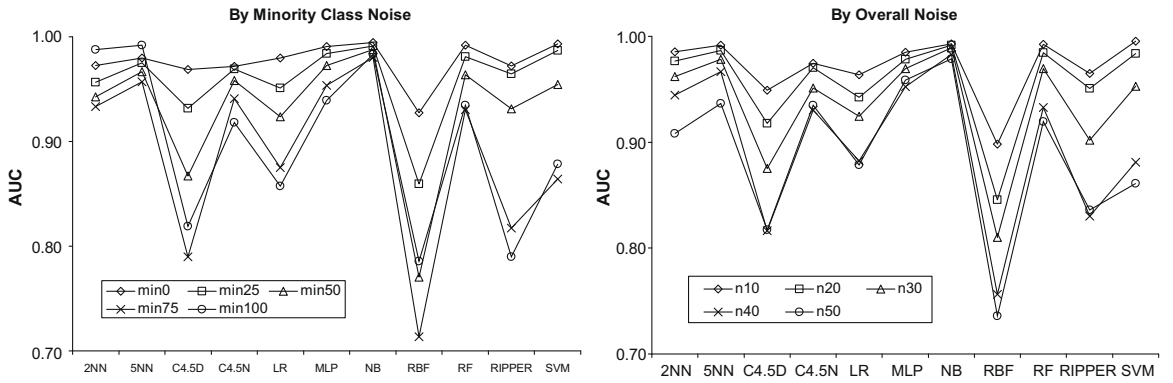

Fig. 3. Trends in AUC for Ψ and Λ, 5 SE datasets.

4.1. Analysis of variance: learning from mislabeled and imbalanced data

In order to analyze the statistical significance of the factors used in our experiments, *analysis of variance* (ANOVA) techniques are utilized. ANOVA models are useful because the variability in performance can be divided among the different experimental factors, allowing the factors which significantly contribute to changes in the performance to be determined. A three factor, full factorial ANOVA model [4] can be written as

$$\mathrm{AUC}_{ijkl} = \mu + \Theta_i + \Lambda_j + \Psi_k + (\Theta\Lambda)_{ij} + (\Theta\Psi)_{ik} + (\Lambda\Psi)_{jk} + (\Theta\Lambda\Psi)_{ijk} + \epsilon_{ijkl}$$

where:

- μ is the overall mean performance (AUC).
- Θ, Λ, Ψ are the three main factors or effects in the experiment, the learner, percentage of overall noise, and percentage of minority class noise, respectively (sampling is not considered in this ANOVA).
- $\Theta_i, \Lambda_j, \Psi_k$ are the treatment effects of the ith, jth, and kth levels of the experimental factors Θ, Λ, Ψ. i is an index on the learners, $i = 1, \ldots, 11$, j is an index on the percentage of overall noise, $j = 1, \ldots, 5$, and k is an index on the percentage of minority class noise, $k = 1, \ldots, 5$.
- $(\Theta\Lambda)_{ij}, (\Theta\Psi)_{ik}, (\Lambda\Psi)_{jk}$ are two-way interaction terms between the main effects.
- $(\Theta\Lambda\Psi)_{ijk}$ is a three-way interaction term between the main effects.
- AUC_{ijkl} is the AUC of the lth observation for the ith level of Θ, jth level of Λ, and the kth level of Ψ. Note that since there are five datasets, each with ten different cross validation runs, $l = 1, \ldots, 50$. If the experiments are considered in an $11 \times 5 \times 5$ design matrix, with each of the main factors representing a different dimension in the matrix, then each cell of the matrix has 50 observations.
- ϵ_{ijkl} is the random error.

The ANOVA model can be used to test the hypothesis that the AUC for the main factors $\Theta_i, \Lambda_j, \Psi_k$ are equal against the alternative hypothesis that at least one mean is different. If the alternative hypothesis (i.e., that at least one mean is different) is accepted, numerous procedures can be used to determine which of the means are significantly different from the others. This involves the comparison of two means, with the null hypothesis that the means are equal. Multiple comparison tests in this work utilized Tukey's Studentized Range (Honestly Significant Difference or HSD) test. All tests of statistical significance utilize a significance level α of 5%. A second ANOVA model was constructed using the KS as the dependent variable. The results were similar to that of the AUC, and hence the details are omitted.

Table 6
ANOVA table.

Effect	DF	5 SE datasets				2 UCI datasets			
		SS	MS	F	p-Value	SS	MS	F	p-Value
Learner (Θ)	10	95.13	9.51	1210	0.0	26.93	2.69	484	0.0
Overall noise (Λ)	4	45.26	11.31	1439	0.0	4.15	1.04	186	0.0
Minority noise (Ψ)	4	52.41	13.10	1667	0.0	3.40	0.85	153	0.0
$\Theta \times \Lambda$	40	8.25	0.20	26	0.0	2.57	0.06	12	0.0
$\Theta \times \Psi$	40	29.05	0.73	92	0.0	3.14	0.08	14	0.0
$\Lambda \times \Psi$	15	24.41	1.63	207	0.0	1.32	0.09	16	0.0
$\Theta \times \Lambda \times \Psi$	150	13.39	0.09	11	0.0	1.59	0.01	2	0.0

Table 7
Effect Λ: SE datasets.

Λ (%)	N	AUC	HSD
10	2750	0.981	A
20	2750	0.970	B
30	2750	0.954	C
40	2750	0.926	D
50	2200	0.917	E

Table 8
Effect Ψ: SE datasets.

Ψ (%)	N	AUC	HSD
0	2750	0.983	A
25	2750	0.970	B
50	2750	0.950	C
100	2200	0.930	D
75	2750	0.918	E

Table 9
Effect Θ: SE datasets.

Θ	N	AUC	HSD
NB	1200	0.990	A
5NN	1200	0.978	B
MLP	1200	0.977	BC
RF	1200	0.974	C
SVM	1200	0.968	D
2NN	1200	0.965	DE
C4.5N	1200	0.963	E
LR	1200	0.939	F
RIPPER	1200	0.927	G
C4.5D	1200	0.903	H
RBF	1200	0.842	I

Table 6 presents the ANOVA table for our experiment. The first column lists the seven experimental factors, three main effects, three two-way interactions, and one three-way interaction, followed by the degrees of freedom (DF), sum of squares (SS), mean square (MS), F-statistic, and the p-value of the F-statistic. Two ANOVA models were constructed, one for the five SE datasets and another for the two UCI datasets. All of the effects are statistically significant, with a p-value much less than 5%. The three main effects are analyzed in Tables 7–9 for the SE datasets and Tables 10–12 for the UCI datasets, all of which contribute significantly to variability in the AUC based on the p-values in Table 6. Note that for each of these tables, the AUC is averaged over all values of the other variables. For example, in Table 7, the average AUC for all 11 learners, 5 datasets, 10 runs of cross validation, and 5 levels of Ψ for $\Lambda = 10\%$ is 0.981 (note N = 2750, which is 11 learners $\times$ 5 datasets $\times$ 10 runs of cross validation $\times$ 5 values of Ψ).

Tables 7 and 10 show the overall level of noise Λ. The column entitled 'N' represents the number of observations, the third column is the mean AUC, and the last column is the HSD grouping. If two or more levels of a factor have the same block letter, then they are not significantly different, with an $\alpha = 5\%$ confidence level. Tables 8 and 11 show the percentage of noise from the minority class Ψ. In general, as Ψ increases, the AUC decreases. The only deviation from this trend is that $\Psi = 75\%$ has a lower AUC than $\Psi = 100\%$. This is explained, however, by the fact that $\Lambda = 50\%$ could not be implemented with $\Psi = 100\%$. By considering Figs. 1 and 2 and in particular the subfigure entitled 'All Learners', it is clear that for $\Lambda \leqslant 40\%$, $\Psi = 100\%$ performs worse than $\Psi = 75\%$. The third main effect, the learner Θ, is given in Tables 9 and 12.

The ANOVA model included three two-way interactions and one three-way interaction between the main effects. An interaction between factors is significant if changes in one of the factors has a significant influence on the other. A graph of the interaction between the learners and level of noise (the interaction term $\Theta \times \Lambda$), is given in Fig. 3 (for space considerations, only the five SE datasets are included here). If there was no significant relationship between the learners and level of noise, then the lines in Fig. 3 would be approximately parallel to one another. In other words, changing the learner would result in the same change in AUC for different levels of class noise. As observed earlier, this is not the case, since for example the NB learner results in AUC values closely clustered for the five levels of noise, but other learners like RBF, C4.5D, and RIPPER exhibit more variability in the AUC as Λ changes. The interaction between the learners and noise from the minority class is also presented in Fig. 3. This interaction was also significant in the ANOVA model, and it can be seen that there is a strong interaction between Θ and Ψ. The power of applying ANOVA analysis is that it can be objectively determined if the relation-

Table 10
Effect Λ: UCI datasets.

Λ (%)	N	AUC	HSD
10	330	0.981	A
20	330	0.970	B
30	330	0.956	C
40	330	0.935	D
50	264	0.924	E

Table 11
Effect Ψ: UCI datasets.

Ψ (%)	N	AUC	HSD
0	330	0.979	A
25	330	0.968	B
50	330	0.952	C
100	264	0.940	D
75	330	0.933	D

Table 12
Effect Θ: UCI datasets.

Θ	N	AUC	HSD
5NN	144	0.992	A
LR	144	0.987	AB
2NN	144	0.984	BC
MLP	144	0.981	BC
NB	144	0.979	C
RF	144	0.978	C
RBF	144	0.977	C
C4.5N	144	0.963	D
SVM	144	0.940	E
C4.5D	144	0.803	F
RIPPER	144	0.782	F

ship between factors is statistically significant. The interaction term $\Lambda \times \Psi$ is given in the bottom-right of Figs. 1 and 2 (in the panel labeled 'All Learners'). Significance of the three-way interaction term $\Theta \times \Lambda \times \Psi$ implies that the two-way interaction $\Lambda \times \Psi$ depends on the learner Θ. Two-dimensional graphs of the interaction term $\Theta \times \Lambda \times \Psi$ were provided in Figs. 1 and 2, and as discussed previously the relationship between Λ and Ψ does depend on the learner being used. In other words, the ANOVA model has validated the statistical significance of the remarks made in the previous section that some learners are more or less sensitive to different levels of noise and to the interaction between the level and distribution of noise. Indeed, the statistical tests of this section confirm that these behaviors are not anomalies.

5. Experimental evaluation II: using sampling to alleviate imbalance in noisy data

This section considers the utilization of sampling in the context of noisy and imbalanced data. For space considerations, we often include only the results for the AUC, however, the results using the KS statistic are typically similar. Since sampling techniques are often used to alleviate class imbalance, we would like to understand the effect of class noise on sampling. Tables 13 and 14 present the AUC and KS for each sampling technique and learner, averaged over all five SE datasets, while Tables 15 and 16 display the same information for the two UCI datasets (the AUC and KS values are averaged over all values of Λ and Ψ in these four tables). The value of the performance measure which results in the best performance for a given learner is underlined, with the worst performance in bold. WE and RUS generally perform very well (RUS in particular substantially improves both C45D and RIPPER), while CBOS and OSS exhibit relatively weak performance for all learners. These results suggest that when constructing learners from noisy and imbalanced data, the application of sampling techniques, and in particular RUS and WE, will positively impact the classifier's performance. Additional evaluation of this conclusion follows throughout this section as we explore the performance of sampling technique/learner combinations for different levels of Λ and Ψ.

Tables 17 and 18 display the AUC for each learner and sampling technique by the overall level of class noise (Λ), averaged over all values of Ψ. RUS and WE are particularly strong with increasing levels of noise. In other words, RUS and WE, in particular, perform very well when the data is both noisy and imbalanced. When the data is imbalanced but relatively clean (i.e.,

Table 13
Overall results by learner and sampling technique, AUC, 5 SE datasets.

Sampling technique	2NN	5NN	C4.5D	C4.5N	LR	MLP	NB	RBF	RF	RIPPER	SVM
BSM	0.950	0.972	0.901	0.965	0.914	0.974	0.980	0.884	0.974	0.933	0.981
CBOS	0.881	0.907	0.813	**0.870**	**0.837**	**0.858**	**0.862**	**0.796**	0.930	**0.823**	**0.891**
NONE	0.957	0.973	0.878	0.953	0.920	0.969	0.988	0.812	0.961	0.899	0.938
OSS	**0.849**	**0.890**	**0.838**	0.911	0.910	0.951	0.970	0.807	**0.887**	0.867	0.940
ROS	0.945	0.966	0.878	0.945	0.918	0.973	0.988	0.819	0.956	0.918	0.982
RUS	0.945	0.973	0.946	0.957	0.921	0.977	0.982	0.886	0.976	0.963	0.988
SM	0.945	0.968	0.910	0.960	0.910	0.970	0.983	0.858	0.972	0.934	0.981
WE	0.961	0.975	0.910	0.961	0.937	0.972	0.990	0.864	0.970	0.920	0.961

Table 14
Overall results by learner and sampling technique, KS, 5 SE datasets.

Sampling technique	2NN	5NN	C4.5D	C4.5N	LR	MLP	NB	RBF	RF	RIPPER	SVM
BSM	0.866	0.888	0.849	0.902	0.794	0.896	0.909	0.786	0.901	0.867	0.913
CBOS	**0.732**	**0.764**	**0.679**	**0.769**	**0.684**	**0.728**	**0.688**	**0.671**	0.816	**0.667**	**0.753**
NONE	0.859	0.880	0.785	0.884	0.807	0.876	0.922	0.731	0.880	0.785	0.860
OSS	0.747	0.767	0.747	0.822	0.793	0.841	0.884	0.703	**0.763**	0.750	0.857
ROS	0.858	0.871	0.785	0.869	0.811	0.904	0.922	0.736	0.869	0.837	0.921
RUS	0.830	0.881	0.893	0.896	0.805	0.886	0.908	0.805	0.890	0.913	0.928
SM	0.855	0.875	0.854	0.894	0.796	0.894	0.916	0.763	0.894	0.867	0.917
WE	0.869	0.881	0.841	0.904	0.846	0.882	0.936	0.799	0.902	0.831	0.903

Table 15
Overall results by learner and sampling technique, AUC, 2 UCI datasets.

Sampling technique	2NN	5NN	C4.5D	C4.5N	LR	MLP	NB	RBF	RF	RIPPER	SVM
NONE	0.972	0.986	**0.768**	0.953	0.985	0.973	0.975	0.974	**0.969**	**0.772**	**0.917**
RUS	0.971	0.990	0.899	**0.939**	0.977	0.974	0.974	0.969	0.981	0.933	0. 980
SM	**0.958**	**0.973**	0.857	0.946	**0.977**	**0.973**	**0.968**	**0.968**	0.971	0.913	0.983
WE	0.980	0.988	0.803	0.957	0. 985	0.976	0.975	0.976	0.974	0.799	0. 943

Table 16
Overall results by learner and sampling technique, KS, 2 UCI datasets.

Sampling technique	2NN	5NN	C4.5D	C4.5N	LR	MLP	NB	RBF	RF	RIPPER	SVM
NONE	0.885	0.928	**0.566**	0.827	0.901	**0.891**	0.863	0.878	**0.875**	**0.519**	**0.733**
RUS	0.888	0.933	0.811	**0.782**	0.876	0.892	0.859	**0.863**	0.882	0.841	0.890
SM	**0.870**	**0.899**	0.735	0.795	**0.871**	0.901	**0.843**	0.868	0.881	0.803	0.899
WE	0.909	0.938	0.613	0. 835	0.901	0.903	0.863	0. 884	0.889	0.577	0.789

with $\Lambda = 10\%$), the improvement gained by the application of sampling is often small. Further, the impact of sampling depends on the learner – even at the 50% noise level, sampling does not significantly improve NB or LR, while at lower levels of noise, learners such as C4.5D and RIPPER can be improved dramatically by sampling. This suggests a significant three-way interaction between the learner, sampling technique, and level of noise.

Tables 19 and 20 present the AUC for each learner and sampling by the level of noise corrupted from the minority class (Ψ), averaged over all values of Λ. With $\Psi = 0\%$, sampling rarely provided significant benefit as far as increased AUC. As Ψ increases, the effectiveness of RUS compared to the other sampling techniques increases dramatically. With $\Psi = 100\%$, RUS is the best sampling technique for seven learners in both Tables 19 and 20. WE performs very well as a sampling technique with a lower level of minority class noise ($\Psi = 25\%$ and $\Psi = 50\%$), but with more noise coming from the minority class, RUS is generally the best sampling technique.

One of the important conclusions that can be derived from Tables 17–20 is that imbalance by itself may not harm the classifier, but some other problem in combination with class imbalance is the actual cause. Most of the learners perform very well when the noise is less impactful (i.e., Λ and Ψ are small), and applying sampling often does not improve performance. Interestingly, this pattern is most evident in Tables 19 and 20 when Ψ is small, i.e., when most of the noise results from the corruption of instances from the true majority class. For almost all learners, the improvements in AUC obtained by sampling occur primarily when the data contains a high percentage of impactful noise, i.e., higher levels of Λ and/or higher levels of Ψ.

Table 17
AUC by learner, sampling technique, and overall noise, 5 SE datasets.

Λ (%)	Sampling technique	2NN	5NN	C4.5D	C4.5N	LR	MLP	NB	RBF	RF	RIPPER	SVM
10	BSM	0.982	0.991	0.960	0.983	0.961	0.991	0.991	0.954	0.996	0.970	0.996
	CBOS	0.947	0.961	**0.912**	**0.964**	**0.925**	**0.962**	**0.930**	**0.877**	0.988	**0.916**	**0.958**
	NONE	0.986	0.992	0.949	0.975	0.964	0.985	0.993	0.898	0.993	0.965	0.995
	OSS	**0.920**	**0.955**	0.939	0.974	0.963	0.983	0.982	0.897	**0.960**	0.947	0.995
	ROS	0.980	0.988	0.964	0.981	0.966	0.993	0.993	0.906	0.994	0.969	0.996
	RUS	0.981	0.992	0.972	0.976	0.949	0.985	0.990	0.923	0.996	0.973	0.994
	SM	0.981	0.991	0.967	0.984	0.962	0.992	0.991	0.941	0.996	0.971	0.996
	WE	0.986	0.991	0.964	0.975	0.982	0.987	0.994	0.940	0.994	0.967	0.998
20	BSM	0.968	0.984	0.934	0.978	0.934	0.985	0.986	0.921	0.990	0.959	0.989
	CBOS	0.912	0.935	**0.860**	**0.931**	**0.874**	**0.912**	**0.894**	**0.831**	0.970	**0.867**	**0.924**
	NONE	0.977	0.987	0.918	0.970	0.943	0.978	0.992	0.846	0.984	0.951	0.984
	OSS	**0.887**	**0.927**	0.891	0.949	0.940	0.972	0.977	0.852	**0.931**	0.926	0.983
	ROS	0.967	0.980	0.924	0.974	0.943	0.988	0.992	0.854	0.985	0.950	0.991
	RUS	0.969	0.986	0.963	0.971	0.938	0.984	0.986	0.906	0.990	0.971	0.993
	SM	0.968	0.984	0.948	0.979	0.936	0.987	0.991	0.898	0.991	0.960	0.991
	WE	0.979	0.987	0.946	0.973	0.959	0.984	0.994	0.905	0.990	0.961	0.994
30	BSM	0.952	0.975	0.902	0.969	0.917	0.978	0.981	0.882	0.980	0.937	0.984
	CBOS	0.878	0.908	**0.802**	**0.879**	**0.834**	**0.859**	**0.865**	**0.800**	0.940	**0.813**	**0.893**
	NONE	0.962	0.978	0.875	0.951	0.924	0.970	0.989	0.810	0.969	0.902	0.953
	OSS	**0.847**	**0.894**	0.832	0.910	0.918	0.950	0.970	0.812	**0.893**	0.869	0.955
	ROS	0.949	0.969	0.876	0.957	0.924	0.979	0.989	0.817	0.967	0.919	0.987
	RUS	0.952	0.977	0.952	0.964	0.925	0.979	0.983	0.890	0.983	0.964	0.990
	SM	0.948	0.973	0.918	0.968	0.915	0.977	0.988	0.861	0.980	0.939	0.985
	WE	0.967	0.979	0.914	0.964	0.943	0.972	0.992	0.871	0.980	0.929	0.979
40	BSM	0.937	0.965	0.859	0.948	0.877	0.960	0.970	0.837	0.953	0.898	0.969
	CBOS	0.848	0.882	**0.746**	**0.822**	**0.783**	**0.795**	**0.829**	0.753	0.886	**0.760**	**0.849**
	NONE	0.945	0.967	0.817	0.930	0.882	0.953	0.983	0.756	0.933	0.830	0.881
	OSS	**0.808**	**0.852**	0.766	0.864	0.866	0.918	0.954	**0.749**	**0.839**	0.794	0.890
	ROS	0.930	0.958	0.815	0.920	0.879	0.959	0.983	0.762	0.927	0.867	0.970
	RUS	0.929	0.966	0.932	0.946	0.908	0.968	0.976	0.865	0.968	0.953	0.983
	SM	0.930	0.959	0.865	0.942	0.869	0.955	0.978	0.806	0.954	0.897	0.969
	WE	0.949	0.968	0.864	0.944	0.899	0.955	0.985	0.807	0.946	0.865	0.920
50	BSM	0.902	0.941	0.837	0.940	0.871	0.949	0.971	0.813	0.945	0.895	0.966
	CBOS	0.804	0.833	**0.728**	**0.727**	**0.754**	**0.738**	**0.777**	**0.702**	0.850	**0.745**	**0.818**
	NONE	0.908	0.937	0.817	0.935	0.879	0.959	0.979	0.736	0.920	0.836	0.861
	OSS	**0.767**	**0.806**	0.747	0.845	0.850	0.929	0.967	0.705	**0.794**	0.781	0.860
	ROS	0.890	0.925	0.797	0.879	0.870	0.938	0.979	0.742	0.897	0.875	0.964
	RUS	0.882	0.934	0.903	0.920	0.873	0.965	0.973	0.836	0.936	0.951	0.977
	SM	0.888	0.925	0.837	0.918	0.855	0.929	0.964	0.764	0.934	0.894	0.961
	WE	0.912	0.941	0.851	0.945	0.895	0.958	0.983	0.782	0.934	0.869	0.901

Other researchers have argued that class imbalance is not the primary problem, for example Japkowicz [20] argues that it is actually small disjuncts which degrade classifier performance. In a related study, Japkowicz and Stephen [21] show that concept complexity can also hinder classifier performance in imbalanced data. Batista et al. [3], by experimenting with 13 imbalanced UCI datasets using the C4.5 learner, conclude that the problem is often related to learning with too few positive examples with confounding effects such as overlapping classes. Interestingly, class noise is related to all of these issues, since labeling errors can create small disjuncts (which are sometimes noisy when $\Psi = 0\%$, or the minority class concept can become disjointed when a large portion of the minority class is corrupted with $\Psi = 100\%$ – see Section 6), make the target concept more complex to learn, and cause the classes to overlap with one another. We contend that class noise, which commonly-occurs in many different application domains, can be the root cause of all of these issues, and hence detailed study of the impact of class noise when learning from imbalanced data is critical.

As either Ψ or Λ increases, the relative performances of the sampling techniques change dramatically. With $\Psi = 0\%$, the baseline learner often performs better without any sampling technique. The relatively weaker performance of WE with $\Psi = 0\%$ is intuitive, because all of the class noise is in the positive class, and WE only removes examples from the negative class. Therefore WE with $\Psi = 0\%$ will not reduce the level of noise in the dataset. With $\Psi = 25\%$, WE is clearly the best sampling technique as measured by the AUC, while the baseline learner without sampling is rarely optimal. As Ψ increases further, the ability of WE begins to diminish, while RUS becomes the best technique most often with $\Psi = 100\%$. WE is most effective when there is a relatively small, non-zero percentage of noise corrupted from the minority class, while with more noise corrupted from the minority class (high values of Ψ), a more dramatic reduction in the majority class may be required.

Table 18
AUC by learner, sampling technique, and overall noise, 2 UCI datasets.

Λ (%)	Sampling technique	2NN	5NN	C4.5D	C4.5N	LR	MLP	NB	RBF	RF	RIPPER	SVM
10	NONE	0.996	0.999	**0.920**	0.975	0.988	0.987	0.976	0.978	0.994	**0.851**	0.987
	RUS	0.990	0.997	0.934	**0.965**	0.985	**0.986**	0.976	**0.974**	0.993	0.953	**0.980**
	SM	**0.988**	**0.993**	0.927	0.976	**0.983**	0.988	**0.970**	0.976	**0.993**	0.955	0.982
	WE	0.998	0.999	0.938	0. 976	0.989	0.989	0.976	0. 979	0.995	0.869	0.987
20	NONE	0.989	0.997	**0.802**	0.966	0.987	0.980	0.976	0.978	0.989	**0.816**	0.982
	RUS	0.984	0.994	0.918	**0.954**	0.983	**0.979**	0.976	**0.975**	0.989	0.946	**0.981**
	SM	**0.976**	**0.986**	0.889	0.964	**0.979**	0.982	**0.970**	0.975	**0.986**	0.935	0.984
	WE	0.995	0.998	0.840	0. 969	0.987	0.985	0.976	0. 980	0.991	0.839	0.984
30	NONE	0.976	0.992	**0.721**	0.957	0.986	**0.974**	0.976	0.976	**0.976**	**0.765**	**0.942**
	RUS	0.975	0.991	0.900	**0.942**	0.979	0.975	0.975	**0.970**	0.984	0.936	0.981
	SM	**0.961**	**0.977**	0.851	0.950	**0.979**	0.976	**0.969**	0.971	0.976	0.911	0.985
	WE	0.985	0.994	0.763	0. 960	0.986	0.975	0.976	0.978	0.979	0. 793	0.974
40	NONE	0.951	0.973	**0.698**	0.941	0.983	**0.963**	0.974	0.970	**0.941**	**0.704**	**0.866**
	RUS	0.961	0.988	0.881	**0.925**	**0.970**	0.968	0.973	0.964	0.976	0. 921	0.980
	SM	**0.940**	**0.964**	0.812	0.932	0.976	0.963	**0.966**	**0.958**	0.956	0.886	0.983
	WE	0.961	0.976	0.731	0.946	0.983	0. 964	0.974	0.973	0.951	0.742	0.913
50	NONE	0.945	0.967	**0.680**	0.921	0.980	0.962	0.971	0.967	**0.938**	**0.710**	**0.782**
	RUS	0.937	0.980	0.854	0.900	**0.968**	0.958	0.967	**0.959**	0.957	0.905	0.977
	SM	**0.914**	**0.941**	0.794	**0.898**	0.968	**0.954**	**0.961**	0.960	0.940	0.870	0.980
	WE	0.956	0.971	0.727	0.927	0. 980	0.964	0.971	0.969	0.948	0.744	0. 834

As was discussed previously, learners are most harmed by high levels of $P \rightarrow N$ noise, i.e., high values of Ψ. This results in a large number of examples that are labeled negative, but should have been labeled positive. As more of the noise is in the negative class, it becomes increasingly important to handle. WE, which is targeting mislabeled examples from the negative class, may be insufficient to reduce noise as Ψ increases, because the user has no direct control over the amount of noise that is eliminated. Therefore, a technique such as RUS may become more effective when the negative class has more noise. In this scenario, RUS has the effect of both balancing the class distribution and reducing noise from the negative class. For example, suppose the negative class has 1000 examples, 50 of which are noisy. Suppose RUS is performed to reduce the 1000 negative examples to 100, a 90% reduction of the negative class. Since the selection of examples are random, then it is expected that 45 of the 50 noisy examples (90%) may also be eliminated. WE may not be able to detect 90% of the noisy examples, and hence more noise remains in the dataset. When undersampling the majority class at very high levels, RUS may be more likely to remove a significant portion of the $P \rightarrow N$-type noise.

Fig. 4 shows the improvement in AUC obtained by the best sampling technique (on the y-axis) for each of the 11 learners for the different levels of Ψ (on the x-axis). The line with the boxes is the AUC obtained by applying the best sampling technique, while the line with '$*$' is the AUC obtained without sampling. Fig. 5 presents the improvement in AUC over the baseline learner for the different levels of Λ. These figures present the results for the five SE datasets only; similar results were observed for the two UCI datasets, and are hence omitted. The improvements obtained by using sampling as either Ψ or Λ varied is dramatically different depending on the learner used. For some learners such as RIPPER, SVM, and C4.5D, sampling significantly improves the performance of the learner at higher levels of Λ or Ψ. For other learners, for example NB, 2NN, and 5NN, there is little performance improvement obtained by sampling even at higher levels of noise. One explanation for this phenomenon is that some learners, after the application of sampling, may be better able to handle the effects of noise – increased concept complexity, small disjuncts, absolute rarity, and class overlap – than others.

5.1. Analysis of variance: sampling with skewed and noisy data

In this section, we construct two additional ANOVA models with an extra factor, the sampling technique Γ. Therefore, the ANOVA model in this section considers four main effects, learner Θ, the percentage of noise Λ, the percentage of noise from the minority class Ψ, and the sampling technique Γ. We also include the six two-way interactions between the four main effects (three-way and four-way interactions were not used because the factors included capture the primary relationships we were interested in analyzing). Table 21 is the ANOVA table with the significance levels of the ten factors, all of which are significant with a p-value less than 5%.

Tables 22–25 include an analysis of the four main effects. The values presented in these tables are averaged over all of the other experimental factors. In Table 23, RUS is the best overall sampling technique (averaged over all learners, levels of noise, and noise distributions), and is significantly better than WE. Both OSS and CBOS perform significantly worse than NONE.

Six interaction terms for the five SE datasets are graphed in Figs. 6–11. In Fig. 6, the average AUC by learner for each level of Λ is plotted. Fig. 7 shows the relationship between Λ and Ψ, while Fig. 8 shows the interaction term $\Theta \times \Psi$. There is a strong interaction between Λ and Ψ, which is confirmed by the F-statistic (which is relatively large for this interaction,

Table 19
AUC by learner, sampling technique, and minority class noise, 5 SE datasets.

Ψ (%)	Sampling technique	2NN	5NN	C4.5D	C4.5N	LR	MLP	NB	RBF	RF	RIPPER	SVM
0	BSM	0.946	0.973	0.963	0.981	0.967	0.988	0.993	0.968	0.994	0.973	0.995
	CBOS	0.933	0.951	0.954	0.972	**0.961**	0.981	0.993	**0.908**	0.985	0.962	**0.987**
	NONE	0.973	0.980	0.969	0.972	0.980	0.991	0.995	0.927	0.992	0.972	0.993
	OSS	**0.812**	**0.872**	**0.937**	**0.971**	0.968	**0.986**	0.995	0.917	**0.916**	**0.920**	0.990
	ROS	0.945	0.973	0.963	0.979	0.974	0.991	0.995	0.934	0.989	0.974	0.996
	RUS	0.955	0.979	0.972	0.975	0.964	0.988	0.994	0.925	0.990	0.977	0.996
	SM	0.950	0.976	0.964	0.980	0.969	0.991	**0.992**	0.954	0.994	0.975	0.996
	WE	0.958	0.977	0.968	0.973	0.979	0.989	0.994	0.932	0.989	0.971	0.995
25	BSM	0.938	0.965	0.934	0.968	0.933	0.979	0.987	0.909	0.982	0.962	0.988
	CBOS	0.886	0.900	0.860	**0.884**	**0.869**	**0.893**	**0.871**	**0.815**	0.955	**0.871**	**0.924**
	NONE	0.957	0.975	0.931	0.969	0.951	0.984	0.991	0.859	0.981	0.964	0.987
	OSS	**0.826**	**0.874**	**0.853**	0.907	0.945	0.979	0.989	0.868	**0.880**	0.908	0.987
	ROS	0.940	0.963	0.938	0.963	0.949	0.986	0.991	0.863	0.975	0.963	0.991
	RUS	0.942	0.974	0.954	0.966	0.936	0.982	0.989	0.894	0.979	0.970	0.992
	SM	0.938	0.968	0.945	0.972	0.938	0.984	0.990	0.885	0.983	0.963	0.990
	WE	0.962	0.976	0.958	0.972	0.972	0.988	0.994	0.915	0.986	0.965	0.993
50	BSM	0.940	0.967	0.907	0.966	0.918	0.974	0.983	0.850	0.973	0.949	0.984
	CBOS	0.850	**0.870**	**0.783**	**0.805**	**0.798**	**0.805**	**0.813**	**0.743**	0.918	**0.794**	**0.860**
	NONE	0.943	0.967	0.867	0.958	0.924	0.972	0.988	0.770	0.964	0.931	0.954
	OSS	**0.842**	0.879	0.803	0.883	0.917	0.962	0.984	0.766	**0.860**	0.880	0.966
	ROS	0.933	0.955	0.875	0.938	0.922	0.975	0.988	0.777	0.955	0.933	0.987
	RUS	0.928	0.965	0.940	0.955	0.900	0.976	0.981	0.857	0.968	0.964	0.988
	SM	0.930	0.960	0.914	0.960	0.908	0.971	0.986	0.811	0.970	0.944	0.985
	WE	0.957	0.971	0.913	0.964	0.949	0.978	0.992	0.855	0.978	0.952	0.983
75	BSM	0.946	0.968	0.840	0.961	0.882	0.959	0.973	0.820	0.960	0.898	0.971
	CBOS	**0.836**	**0.870**	**0.710**	**0.762**	**0.746**	**0.745**	**0.783**	**0.706**	0.874	**0.730**	**0.819**
	NONE	0.933	0.957	0.790	0.941	0.875	0.954	0.980	0.713	0.931	0.817	0.864
	OSS	0.870	0.896	0.772	0.883	0.866	0.936	0.972	0.696	**0.852**	0.816	0.873
	ROS	0.930	0.950	0.789	0.902	0.872	0.941	0.980	0.719	0.917	0.867	0.967
	RUS	0.924	0.957	0.921	0.938	0.859	0.966	0.972	0.832	0.956	0.953	0.980
	SM	0.928	0.950	0.853	0.940	0.860	0.934	0.968	0.764	0.952	0.896	0.964
	WE	0.944	0.961	0.843	0.952	0.893	0.958	0.983	0.775	0.946	0.865	0.908
100	BSM	0.989	0.993	0.852	0.944	0.858	0.966	0.961	0.872	0.958	0.873	0.965
	CBOS	**0.903**	0.950	**0.744**	0.945	**0.806**	**0.870**	**0.849**	0.813	**0.914**	**0.745**	**0.861**
	NONE	0.988	0.992	0.819	0.918	0.857	0.939	0.983	0.786	0.935	0.790	0.879
	OSS	0.908	**0.939**	0.823	**0.911**	0.837	0.879	0.896	**0.785**	0.940	0.798	0.869
	ROS	0.987	0.991	0.813	0.943	0.866	0.970	0.983	0.799	0.942	0.835	0.967
	RUS	0.985	0.993	0.942	0.949	0.949	0.970	0.974	0.930	0.991	0.947	0.982
	SM	0.989	0.993	0.864	0.944	0.863	0.969	0.978	0.879	0.960	0.880	0.968
	WE	0.987	0.992	0.861	0.940	0.882	0.940	0.985	0.840	0.948	0.831	0.915

1047) from Table 21 (recall that non-parallel lines would indicate an interacting relationship between factors). Therefore, the impacts of class noise Λ are closely related to the level of Ψ. For example, with $\Psi = 75\%$, 30% class noise will dramatically reduce the performance, while with $\Psi = 0\%$, 30% class noise has little impact on most classifiers. The statistical significance of this relationship as quantified by the ANOVA model confirms the importance of this interaction.

The interaction term $\Theta \times \Lambda$ (Fig. 6) is relatively weak, as the lines are somewhat close to parallel. This is confirmed by the F-statistics (70 and 15) from Table 21, which though significant is not large relative to the other factors. Contrast this with the F-statistics for the interaction $\Theta \times \Psi$ (356 and 21). Since these factors have the same number of degrees of freedom (40), they can be directly compared, and clearly $\Theta \times \Psi$ is a more important experimental factor. This implies that Ψ, the percentage of minority class noise, has a more significant role in differentiating between learner performance that the overall level of noise Λ (these results are also evident from Table 6, when we did not consider the effects of sampling). We have seen support for this result in prior analysis, because some learners (2NN and 5NN in particular) performed well with $\Psi = 100\%$, while most others performed very poorly. These trends contribute to the significance of the interaction $\Theta \times \Psi$.

Similar to the comments made comparing the interactions $\Theta \times \Psi$ and $\Theta \times \Lambda$, Fig. 9 shows $\Gamma \times \Lambda$ (F = 78 and 9) as a less significant interaction compared to $\Gamma \times \Psi$ (F = 240 and 27) in Fig. 11 (which is once again confirmed by the F-statistics). In other words, Ψ has a greater effect on differences in the performance of sampling techniques than Λ. The interaction $\Theta \times \Gamma$, which is also a significant factor in the ANOVA analysis, is given in Fig. 10.

Table 20
AUC by learner, sampling technique, and minority class noise, 2 UCI datasets.

Ψ (%)	Sampling technique	2NN	5NN	C4.5D	C4.5N	LR	MLP	NB	RBF	RF	RIPPER	SVM
0	NONE	0.995	0.997	0. 937	0.966	0.987	0.983	0.977	0.981	0. 992	**0.867**	**0.956**
	RUS	0.979	0.996	**0.914**	**0.953**	0.983	**0.978**	0.976	0.978	0.988	0.949	0.984
	SM	**0.958**	**0.979**	0.921	0.961	**0.980**	0.981	**0.970**	**0.977**	**0.981**	0.927	0.985
	WE	0.994	0.997	0.935	0.965	0.987	0.984	0.976	0. 983	0.993	0.915	0.985
25	NONE	0.986	0.996	**0.853**	0.953	0.986	0.982	0.976	0.978	0.986	**0.827**	**0.936**
	RUS	0.972	0.994	0.900	**0.939**	0.981	**0.977**	0.975	0.974	0.983	0.937	0.982
	SM	**0.955**	**0.974**	0.886	0.944	**0.978**	0.980	**0.969**	**0.973**	**0.977**	0.915	0.984
	WE	0.994	0.997	0.907	0. 960	0.986	0.982	0.976	0.980	0.990	0. 847	0.953
50	NONE	0.973	0.990	**0.716**	0.945	0.985	0.976	0.975	0.974	0.975	**0.774**	**0.889**
	RUS	0.963	0.989	0.889	**0.929**	0.978	**0.971**	0.974	0.968	0.976	0.934	0.979
	SM	**0.952**	**0.968**	0.846	0.932	**0.975**	0.973	**0.967**	**0.968**	**0.970**	0.911	0.982
	WE	0.985	0.993	0.766	0. 953	0.985	0.980	0.975	0. 977	0.980	0.793	0.928
75	NONE	0.953	0.972	**0.648**	0.939	0.982	0.965	0.973	0.967	**0.944**	**0.689**	**0.870**
	RUS	0.960	0.984	0.883	**0.924**	0.971	0.965	0.970	**0.959**	0.970	0.917	0.975
	SM	**0.950**	**0.963**	0.803	0.929	**0.970**	**0.960**	**0.964**	0.961	0.958	0.903	0.979
	WE	0.963	0.976	0.690	0.943	0. 983	0.968	0.973	0.968	0.954	0.718	0. 891
100	NONE	**0.951**	**0.973**	**0.662**	0.966	0.984	**0.957**	0.973	0.969	**0.941**	**0.683**	**0.940**
	RUS	0.982	0.989	0.913	**0.951**	**0.972**	0.980	0.974	0.964	0. 986	0.928	0.980
	SM	0.977	0.986	0.822	0.969	0.984	0.971	**0.968**	**0.961**	0.971	0.909	0.986
	WE	0.959	0.975	0.695	0.967	0.984	0.963	0.973	0.971	0.947	0. 705	0.962

6. Discussion of results

The empirical results clearly demonstrate that $P \rightarrow N$ noise has a significant impact on almost all of the classifiers when learning from imbalanced data. While some researchers have preliminarily considered this idea (for example, Wilson's editing [2] and Zhu and Wu [51]), noise handling literature has not sufficiently addressed this important issue. An example illustrating the impact of class noise in skewed data and the differences between $\Psi = 0\%$ and $\Psi = 100\%$ is given in Fig. 12, which contains a scatterplot of a simulated, two-dimensional dataset. The dataset contains 20 positive (squares) and 50 negative (triangles) examples. In the original dataset, on the left-hand-side of the figure, the data is easily separated by a linear decision boundary. The middle figure shows the case where $\Lambda = 30\%$ and $\Psi = 0\%$ (in other words, all noisy examples are of the type $N \rightarrow P$). The noisy examples are filled-in squares. The original cluster of positive examples remains, while the cluster of negative examples has become more noisy (with mislabeled positive examples). The negative class consists of 38 correctly labeled examples, while the positive class consists of 32 examples, 12 of which are mislabeled.

The right-hand-side of Fig. 12 shows the same dataset with $\Lambda = 30\%$ and $\Psi = 100\%$. The negative class consists of 62 examples, 12 of which are mislabeled, while the positive class contains only eight examples. In this situation, the cluster of positive examples is significantly obscured by the presence of incorrectly labeled negative examples. It is therefore more difficult to learn the 'true' minority class because it is significantly fragmented and obscured. With $\Psi = 0\%$, the true minority class is uncorrupted, and there are just some noisy examples in the negative class. With $\Psi = 100\%$, the positive class now appears to consist of at least two small disjuncts, while with $\Psi = 0\%$, the noisy positive examples are relatively isolated and hence may not appear as disjuncts. In addition, from the perspective of concept complexity, the $\Psi = 100\%$ case appears to be much more complex than $\Psi = 0\%$, because in the later scenario the frequency of noisy positive examples is small relative to the correctly labeled examples in that section of the feature-space. Finally, $\Psi = 100\%$ results in absolute rarity in the positive class – only eight positive class examples remain after noise corruption.

Two lines were added to the figure with $\Psi = 100\%$ and are labeled 'A' and 'B', indicating two possible decision boundaries that might be derived from a learner using this noisy data. One possible decision boundary learned is labeled 'A', since a number of negative examples near the true class boundary are mislabeled, causing a large portion of the positive class region near the true boundary to be missed. Boundary 'B' also captures a large percentage of mislabeled negative examples which truly belong to the positive region of the decision space.

All of the learners utilized in this study were adversely impacted by noise, however some learners (NB, 2NN, 5NN, MLP, and RF) were more stable at higher noise levels than others. In a relative sense, RIPPER, C4.5D, SVM, and RBF demonstrated a more dramatic decrease in performance compared to the other learners at higher noise levels. NB in particular performed very well in our experiments, as confirmed by the ANOVA analysis in Section 5.1.

Tables 17–20 show the effectiveness of the sampling techniques for varying levels of Λ and Ψ. The relative effectiveness of RUS generally increases as either Λ or Ψ increases. In other words, for many of the learners, utilizing RUS (or any other sampling technique, for that matter) at lower noise levels does not improve the performance of learners on imbalanced data.

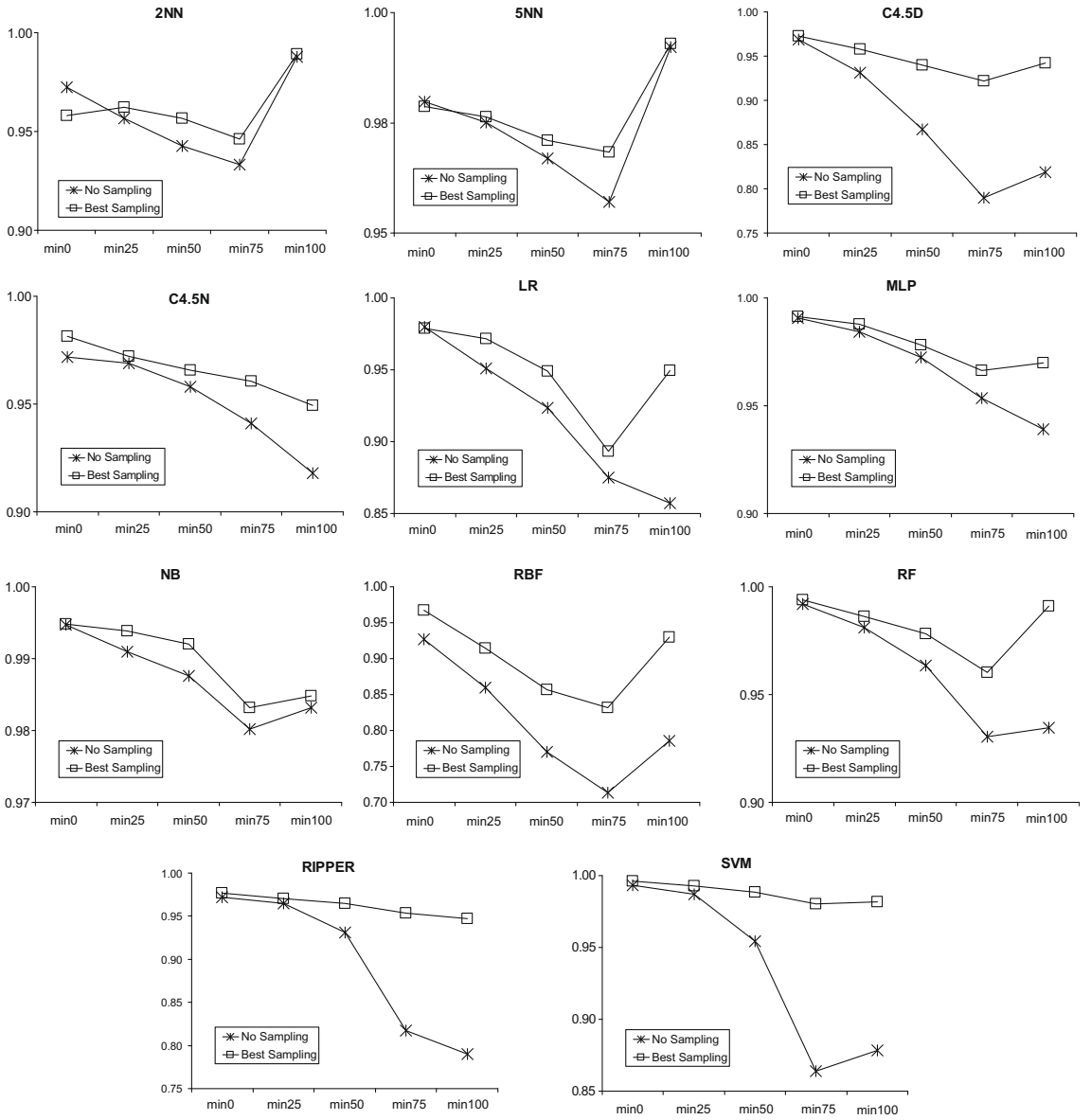

Fig. 4. AUC Comparison of baseline learner and best sampling technique by Ψ, 5 SE datasets.

This observation implies, as mentioned previously, that imbalance by itself may not be the issue – instead, imbalance coupled with another factor, in this case class noise, is the true problem. Note that this observation is not true for all learners. Indeed, even at 10% class noise, the performance of both C4.5D and RBF were improved after sampling was applied, while for other learners such as 2NN, 5NN, and NB, sampling even at higher noise levels did not improve performance.

In addition to the strong performance of random undersampling, WE also works very well. The impact of the best sampling technique, when compared to the baseline learner without sampling, varies depending on the learner, level of noise, and distribution of class noise. Consider Figs. 4 and 5, which compare the AUC obtained by the learner without sampling to that of the best sampling technique for Ψ and Λ, respectively. For SVM, C4.5D, RBF, and RIPPER (and to a lesser extent RF, LR, and MLP), there is a divergence between the AUC for the baseline learner and the best sampling technique. In other words, for these learners, sampling provides a substantial benefit at higher levels of minority class noise or overall noise. One possible explanation of this phenomenon is that sampling is alleviating the side-effects (e.g., disjuncts or overlapping classes) of class noise. However, not all learners benefit from the application of sampling, for example 5NN, 2NN, and NB.

Therefore, we conclude that in general, sampling improves the performance of many different classifiers when confronted with imbalanced data with class noise, and WE and RUS are generally the two best techniques. Further, RUS performed very well with higher levels of minority class noise Ψ. As shown previously, this type of class noise is the most harmful, and RUS is

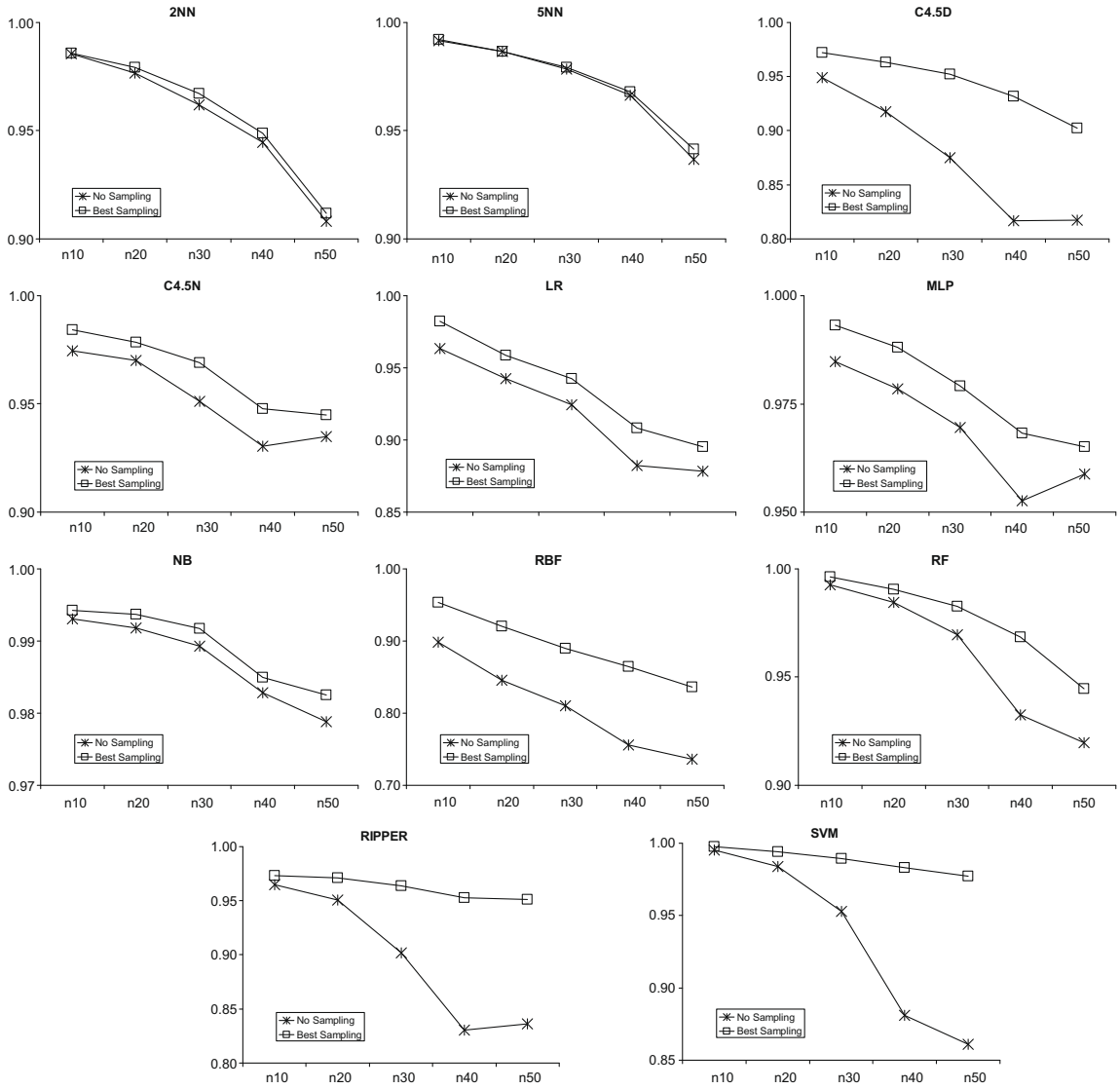

Fig. 5. AUC Comparison of baseline learner and best sampling technique by Λ, 5 SE datasets.

Table 21
ANOVA table, AUC.

Effect	5 SE datasets					2 UCI datasets				
	DF	SS	MS	F	*p*-Value	DF	SS	MS	F	*p*-Value
Learner (Θ)	10	516.19	51.62	5602	0.0	10	52.21	5.22	997	0.0
Sampling technique (Γ)	7	302.54	43.22	4691	0.0	3	0.95	0.32	61	0.0
Overall noise (Λ)	4	390.41	97.60	10593	0.0	4	10.66	2.67	509	0.0
Minority class noise (Ψ)	4	287.61	71.90	7804	0.0	4	5.68	1.42	271	0.0
$\Theta \times \Gamma$	70	71.95	1.03	112	0.0	30	9.88	0.33	63	0.0
$\Theta \times \Lambda$	40	25.83	0.65	70	0.0	40	3.22	0.08	15	0.0
$\Theta \times \Psi$	40	131.31	3.28	356	0.0	40	4.33	0.11	21	0.0
$\Gamma \times \Lambda$	28	20.21	0.72	78	0.0	12	0.55	0.05	9	0.0
$\Gamma \times \Psi$	28	61.88	2.21	240	0.0	12	1.71	0.14	27	0.0
$\Lambda \times \Psi$	15	144.70	9.65	1047	0.0	15	3.57	0.24	46	0.0

generally the most effective at eliminating these examples from the dataset. This is particularly interesting because RUS is not designed to specifically address class noise. An unintended side effect of undersampling is to remove a significant portion of the mislabeled negative class examples, so in addition to balancing the class distribution, noisy examples are also eliminated. WE, which was designed to address instance mislabeling, among other things, also performs well, which would be

Table 22
Main effect: Ψ.

Ψ (%)	5 SE datasets			2 UCI datasets		
	N	AUC	HSD	*N*	AUC	HSD
0	22000	0.978	A	1320	0.977	A
25	22000	0.960	B	1320	0.970	B
50	22000	0.941	C	1320	0.960	C
100	17600	0.937	D	1056	0.958	C
75	22000	0.917	E	1320	0.948	D

Table 23
Main effect: Λ.

Λ (%)	5 SE datasets			2 UCI datasets		
	N	AUC	HSD	*N*	AUC	HSD
10	22000	0.979	A	1320	0.981	A
20	22000	0.966	B	1320	0.974	B
30	22000	0.949	C	1320	0.963	C
40	22000	0.922	D	1320	0.949	D
50	17600	0.905	E	1056	0.938	E

Table 24
Main effect: Γ.

Γ	5 SE datasets			2 UCI datasets		
	N	AUC	HSD	*N*	AUC	HSD
RUS	13200	0.967	A	1584	0.970	A
WE	13200	0.965	B	1584	0.964	B
BSM	13200	0.963	C			
SM	13200	0.961	D	1584	0.963	B
ROS	13200	0.956	E			
NONE	13200	0.954	E	1584	0.957	C
OSS	13200	0.923	F			
CBOS	13200	0.885	G			

Table 25
Main effect: Θ.

Θ (%)	5 SE datasets			2 UCI datasets		
	N	AUC	HSD	*N*	AUC	HSD
NB	9600	0.980	A	576	0.978	CD
SVM	9600	0.978	B	576	0.968	E
MLP	9600	0.971	C	576	0.980	BC
RF	9600	0.970	C	576	0.980	BC
5NN	9600	0.965	D	576	0.990	A
C4.5N	9600	0.956	E	576	0.957	F
2NN	9600	0.944	F	576	0.979	CD
RIPPER	9600	0.930	G	576	0.871	G
LR	9600	0.929	G	576	0.983	B
C4.5D	9600	0.907	H	576	0.858	H
RBF	9600	0.865	I	576	0.975	D

expected. The poor performance of OSS in our experiments is surprising, however it still may be useful for different types of noise not considered in this study. Further experimentation regarding the utility of OSS is advised.

6.1. Addressing our research questions

Section 1.1 included six research objectives to be addressed in this work. Here we summarize our findings in relation to each of these questions:

Q_1 : What is the impact of class noise on learners constructed using skewed datasets?

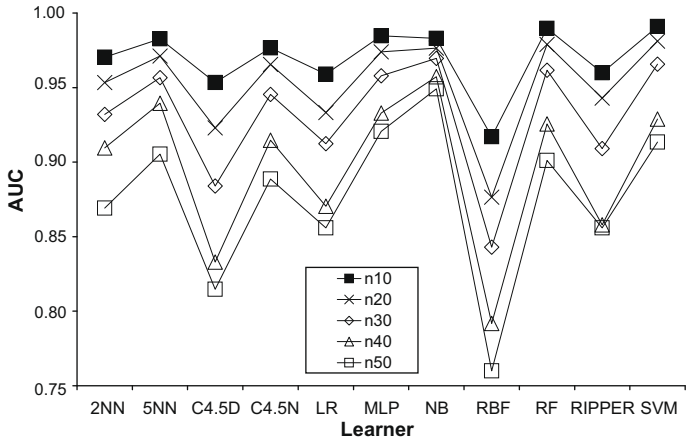

Fig. 6. $\Theta \times \Lambda$.

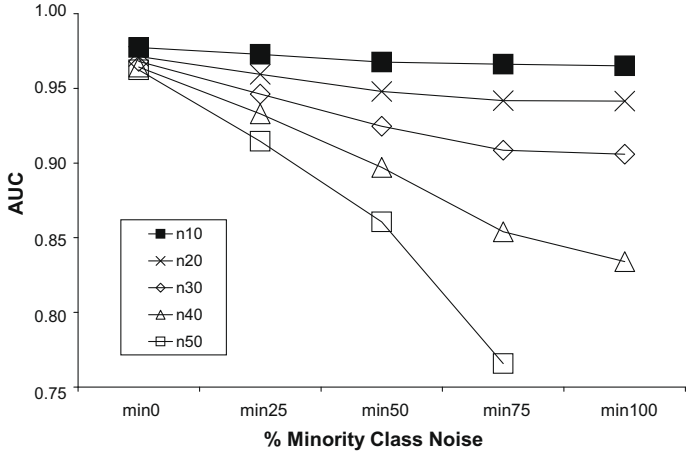

Fig. 7. $\Lambda \times \Psi$.

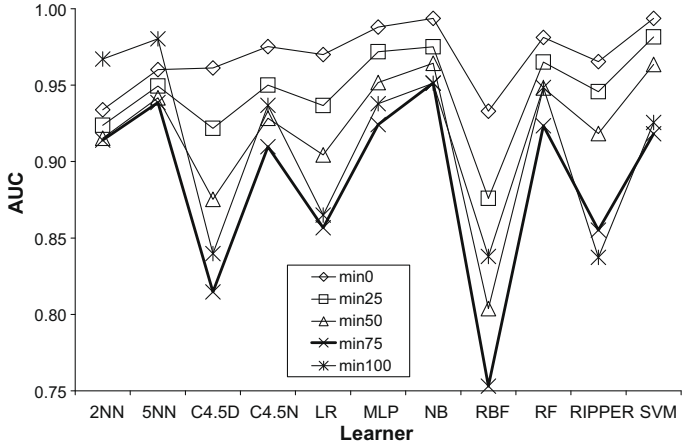

Fig. 8. $\Theta \times \Psi$.

A_1 : Class noise severely impacts the performance of all of the learners utilized in this work, although some learners were relatively robust in the presence of noise.

Q_2 : Is the class in which the noise is located significant?

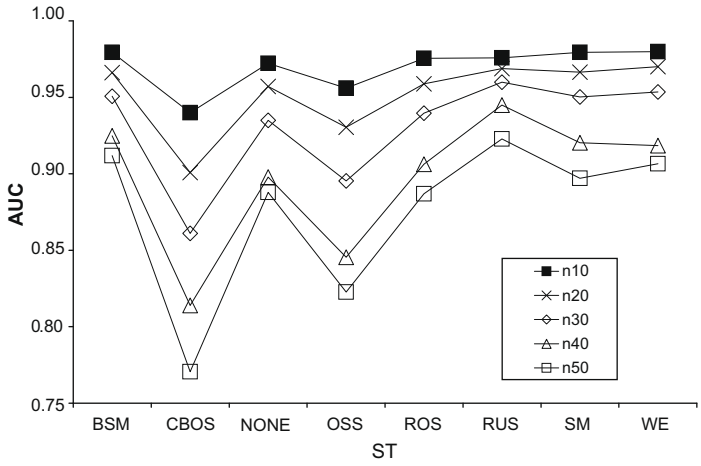

Fig. 9. $\Gamma \times \Lambda$.

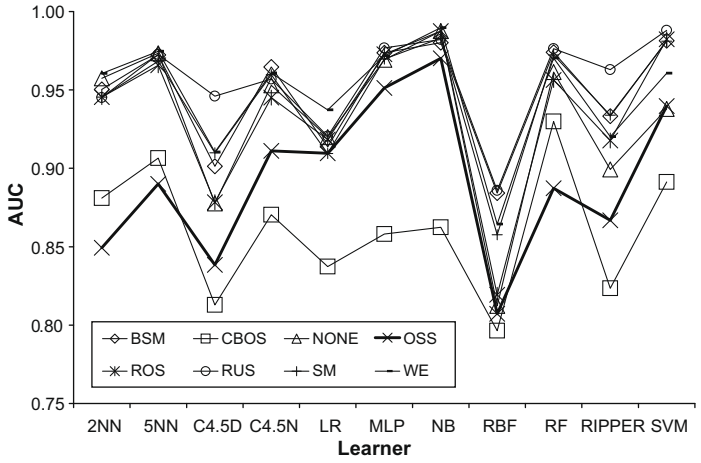

Fig. 10. $\Theta \times \Gamma$.

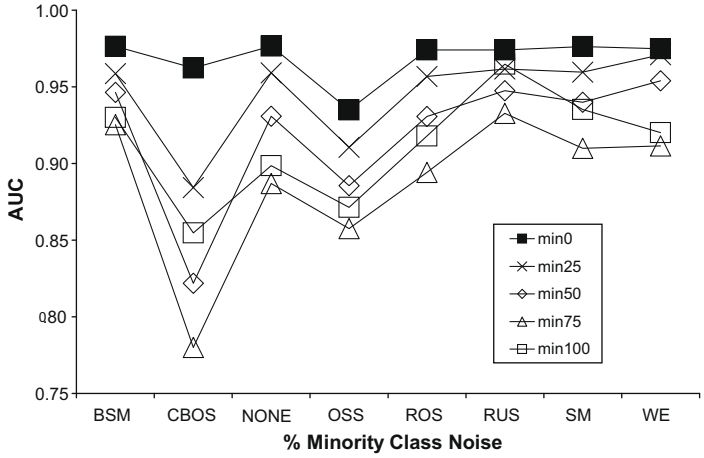

Fig. 11. $\Gamma \times \Psi$.

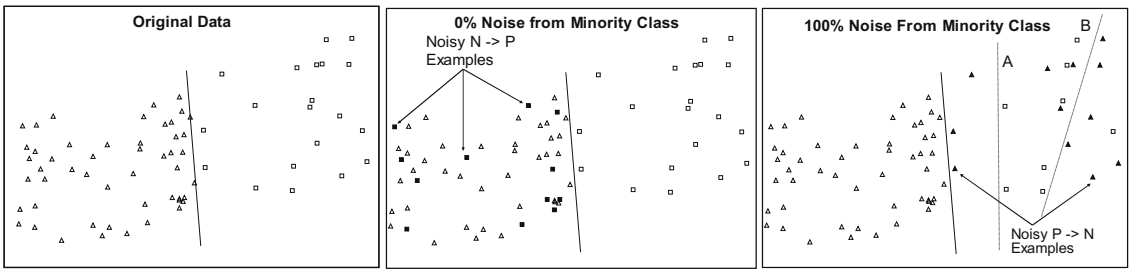

Fig. 12. Sample data with simulated noise.

A_2 : The class in which the noise is located is absolutely critical. As we have shown, high levels of noise with all of that noise coming from the majority class (i.e., all $N \rightarrow P$-type noise) is much less impactful than the same number of noisy examples from the minority class (all noise is of type $P \rightarrow N$). Stated another way, the AUC of most of the learners does not deteriorate as the noise level increases if $\Psi = 0\%$.

Q_3 : Are the effects of class noise in imbalanced datasets uniform across different learning algorithms?

A_3 : Different learners react differently to the presence of noise, and there is wide variation in the relative performances of the algorithms. Some are relatively robust to higher levels of class noise (NB, MLP and to a lesser extent, 2NN and 5NN), while others show moderate or significant deterioration (RIPPER, SVM, C4.5D, RF). The decision tree learner C4.5N, which uses Laplace smoothing and no pruning, was more robust to noise than C4.5D. This result seemed counterintuitive, since theoretically pruning might be useful for eliminating subtrees built on noisy data. As expected, 5NN was generally more robust, and performed better than, 2NN. Generally speaking, two of the newer and more complex learning algorithms, SVM and RF, were less robust than simpler algorithms such as NB, 2NN, and 5NN, and were outperformed in an absolute sense at more impactful noise levels. In summary, class noise did not affect all learners in a uniform manner, and we recommend further consideration of this observation in future work.

Q_4 : How do sampling procedures, which are often used to alleviate class imbalance, perform in the presence of class noise?

A_4 : There was substantial variability in the effectiveness of sampling techniques for different learners. Sampling does help boost the performance of learners such as C4.5D, RIPPER, RF, and SVM, especially for higher levels of Λ and Ψ. For other learners such as NB, 2NN, 5NN, and MLP, sampling does not substantially improve learner performance. Interestingly, these four learners are also generally more robust to class noise. This observation confirms our previously stated hypothesis that class imbalance by itself is not necessarily a cause for concern, but instead imbalance and class noise together are a problem. Since these four learners are robust to class noise, any improvement obtained by sampling would be directed towards the imbalance problem, and as there is no improvement in performance, imbalance by itself does not present a significant problem for these learners.[5]

Note that WE and OSS are the only two sampling techniques specifically designed to handle class noise. The relatively strong performance of WE in our studies was expected, but we found the poor performance of OSS puzzling. The difficulty with the oversampling techniques SM, BSM, ROS, and CBOS is that they may be prone to replicating noisy examples in the minority class, thereby exacerbating the noisy data problem. By including a cluster-based oversampling process to smooth out the within-class imbalance, CBOS in particular may transform small, sparsely populated noisy disjuncts into large clusters, forcing the learner to focus on these noisy regions and hurting performance. RUS, which despite its simplicity performed quite well, also does not specifically target noisy examples. However as a consequence of the reduction of the majority class, noisy examples may indeed also be randomly selected for removal, so the absolute number of noisy examples decreases after the application of RUS.

Q_5 : Do different sampling techniques work better with different levels of noise and with noise represented in different proportions in the two classes?

A_5 : The relatively simple undersampling techniques (RUS and WE) generally performed the best, though results varied by learner, level of noise, and distribution of noise. Omitting the learners which did not generally benefit from the use of sampling (NB, MLP, 2NN, 5NN), RUS was the best sampling technique at higher levels of Ψ. At lower noise levels, the best sampling technique was often only marginally better than the baseline learner. RUS worked best with RIPPER and C4.5D in almost all cases. CBOS and OSS performed extremely poorly. The very weak performance of OSS was surprising, since it was designed to deal with noise (among other things). OSS rarely performed well, and in fact was often the

[5] We believe this statement is true relative to the AUC and KS performance metrics. Other threshold-based metrics such as the geometric mean or F-measure may result in a different conclusion. Regardless of the predictive power of the learner, the default decision threshold used to calculate these measures may be inappropriate for imbalanced data and in such a case, sampling might be beneficial for even these learners. Our previous work has explored this result [43].

worst or second-worst sampling technique. Of course, OSS may be useful for dealing with other types of noise not considered in this work. In addition, WE did not perform well with $\Psi = 0\%$, because WE targets noise from the majority class, while $\Psi = 0\%$ means that all of the noise is in the minority class ($N \rightarrow P$-type noise).

Q_6 : What guidance can be given to improve the commonly-used classification filters in the presence of class imbalance?

A_6 : Classification filters are inappropriate for noise handling with imbalanced data. In particular, it may not be sensible to filter any of the minority class examples from a skewed dataset. The reasoning behind this conclusion is three-fold. First, as we have shown, $N \rightarrow P$ noise does not significantly impact most learning algorithms, primarily because the majority class is relatively large. Second, if correctly labeled positive examples are mistakenly filtered from a dataset, fewer positive examples remain, making the learning process even more difficult. Third, while a normal (i.e., non-cost sensitive) learner uses a class assignment strategy that minimizes the overall misclassification rate, this classification strategy is often inappropriate for identifying minority class cases – in other words, too many minority class examples will be misclassified. Therefore, great care should be taken when filtering positive examples from a highly skewed dataset and classification filters should not be used.

6.2. Threats to validity

Experimental research commonly includes a discussion of two different types of threats to validity [49]. Threats to *internal validity* relate to unaccounted influences that may impact the empirical results. Throughout all of our experiments, great care was taken to ensure the authenticity and validity of our results. The benchmark WEKA data mining tool [48] was used for the construction of all classifiers, and we have included all of the parameter settings to ensure repeatability. The parameters for the learners were chosen to ensure good performance in many different circumstances and to be reasonable for the datasets. The SAS GLM procedure [37] was used to construct the ANOVA models presented in this work, and the assumptions for constructing ANOVA models were validated. In particular, the target variable was transformed $\widehat{\text{AUC}} = \arcsin\sqrt{\text{AUC}}$ as suggested in Berenson et al. [4] to normalize the data and stabilize the variances before constructing the ANOVA model. All output was validated for accuracy by members of our research group, giving us confidence in the legitimacy of the results. We have included seven different sampling techniques in this work, which includes all of those most commonly-used in related work. Since our research group developed software to implement the sampling techniques, all programs were checked for validity and accuracy.

Threats to *external validity* consider the generalization of the results outside the experimental setting, and what limits, if any, should be applied. Seven different datasets were used in our experiments, and we believe that the results presented in this work are valid for other datasets and application domains. In particular, these datasets represent a wide variety of sizes and imbalance levels and come from three different application domains. Future work is required to confirm our conclusions, but the inclusion of five real-world software measurement and two UCI datasets provides strong support for our empirical results. One of the strengths of our work is the use of datasets derived from real-world data, which as mentioned is in contrast to many of the previous studies on data quality, where only simulated data is often used. Constructing over 2 million learners provides a high degree of reliability in our results, and we have considerable confidence in the validity of our experimentation. Future work can also consider different methodologies for injecting class noise. One possibility is to use simulated datasets injected with noise, although such a procedure has its drawbacks.

7. Conclusions

This work presented a comprehensive experimental investigation into knowledge discovery using noisy and imbalanced data, a topic of great interest to practitioners and researchers in many different application domains. In this work, we have included experimentation with a number of experimental factors (7 datasets, 11 learners, 24 combinations of class noise, and 7 sampling techniques) which have resulted in the construction of over 2 million learners. We have also included statistical analysis of the experimental results, an important dimension of reliable empirical work.

The results of our experiments conclusively demonstrate the adverse impacts of class noise on learning from imbalanced data, and in particular noise resulting from the corruption of instances from the true minority class (i.e., $P \rightarrow N$-type noise, or examples with a correct label of P that are mistakenly labeled as N in the training dataset) is seen to be the most severe type. As discussed, this result, which to our knowledge has not been previously analyzed so extensively, has very important consequences to the domain of noise handling. There was substantial variability in the impact of class noise on learner performance depending on the algorithm considered. Simple algorithms such as naïve Bayes and nearest neighbor learners were often more robust (and performed better, in an absolute sense) than newer, more complex learners such as support vector machines or random forests. In addition, seven different and commonly-used sampling techniques were analyzed, and random undersampling, a relatively simple procedure which randomly reduces the size of the majority class, was often found to be the most effective technique, especially at more impactful noise levels. Wilson's editing, another undersampling technique which specifically targets mislabeled examples, also performed very well while more complex oversampling techniques like cluster-based oversampling, SMOTE, and Borderline-SMOTE were either generally moderate or weak. One-sided selection, another sampling technique proposed to handle class noise (among other things), performed surprisingly poorly. Indeed, there was substantial variability in the performance of the sampling techniques across learners and levels of noise.

As with any empirical study, future work can include further studies with additional (real-world and simulated) datasets to confirm our results. Variations on our experimental design can also be considered, for example using different noise injection procedures or different learning algorithms. Other performance measures can also be considered, as we focus solely on the AUC and KS. Other methodologies for handling imbalanced data such as cost sensitive learning can also be evaluated. Future work can also consider the important problem of attribute noise. An exciting and useful area for research is further exploration into the relative sensitivities of learning algorithms in the presence of noise. Multi-class problems are increasingly important in a wide variety of domains, and a detailed study of class noise in such an environment would be very interesting. Our noise injection methodology would need to be extended to reflect different noise mixing distributions among the classes similar to our experimental parameter Ψ. Finally, we recommend the development of a procedure that is better able to simultaneously deal with noise and class imbalance.

Appendix A. Learners

The eleven learners used in our experiments are described briefly in this section, along with an explanation of the parameters. These classifiers were considered since they are commonly-used in the data mining community. All learners were built using the WEKA data mining tool, version 3.5.2 [48]. Changes to default parameter values in WEKA were made only when experimentation showed a general improvement in classifier performance based on preliminary analysis.

Naïve Bayes (NB) is a simple classifier that utilizes Bayes' rule of conditional probability which assumes independence of the predictor attributes. *Logistic regression* (LR) is a statistical regression model for categorical prediction. No modifications to the default parameter values of NB or LR in WEKA were made.

For a *multilayer perceptrons* (MLP) learner, two parameters were changed from the default values. The 'hiddenLayers' parameter was changed to '3' to define a network with one hidden layer containing three nodes, and the 'validationSetSize' parameter was changed to '10' to cause the classifier to leave 10% of the training data aside to be used as a validation set to determine when to stop the iterative training process. Another type of artificial neural network, *Radial basis function networks* (RBF), was utilized in our experiments. The only parameter change for RBF (called 'RBF Network' in WEKA) was to set the parameter 'numClusters' to 10.

Two different *K nearest neighbors* classifiers (IBk in WEKA) were constructed, using $k = 2$ and $k = 5$, and denoted 2NN and 5NN. The 'distanceWeighting' parameter was set to 'Weight by 1/distance' to use inverse distance weighting in determining how to classify an instance. Repeated incremental pruning to produce error reduction (RIPPER) is a rule-based learner. The WEKA implementation of RIPPER is called JRip, and our experiments used the default parameters for this learner.

The *random forest* (RF) classifier [5] utilizes bagging and the 'random subspace method' to construct randomized decision trees. The outputs of ensembles of these randomized, unpruned decision trees are combined to produce the final prediction. No changes to the default parameters were made in our experiments. The *support vector machine* (SVM) learner in WEKA is called SMO. For our experiments, the complexity constant 'c' in WEKA was changed from 1.0 to 5.0, and the 'buildLogisticModels' parameter, which allows proper probability estimates to be obtained, was set to 'true' [48]. A linear kernel was used for the SVM learner in this work, and use of non-linear kernels is left for future exploration.

Two different versions of the C4.5 classifier were used in this study, denoted C4.5D and C4.5N. C4.5D uses the default WEKA parameter settings, while C4.5N uses no pruning and Laplace smoothing [47]. The WEKA implementation of C4.5 is called J48.

Appendix B. Data sampling techniques

Seven sampling techniques proposed for dealing with imbalanced data are considered in this work: random undersampling (RUS), random oversampling (ROS), one-sided selection (OSS), cluster-based oversampling (CBOS), Wilson's editing (WE), SMOTE (SM), and borderline-SMOTE (BSM).

The two most common preprocessing techniques are random *minority oversampling* (ROS) and random *majority undersampling* (RUS). In ROS, instances of the minority class are randomly duplicated in the dataset. In RUS, instances of the majority class are randomly discarded from the dataset.

In one of the earliest attempts to improve upon the performance of random resampling, Kubat and Matwin [30] proposed a technique called *one-sided selection* (OSS). One-sided selection attempts to intelligently undersample the majority class by removing majority class examples that are considered either redundant or 'noisy.'

The technique called *Wilson's editing* [2] (WE) uses the kNN technique with $k = 3$ to classify each example in the training set using all the remaining examples, and removes those majority class examples that are misclassified. Barandela et al. also propose a modified distance calculation, which causes an example to be biased more towards being identified with positive examples than negative ones.

Chawla et al. [10] proposed an intelligent oversampling method called Synthetic Minority Oversampling Technique (SMOTE). SMOTE (SM) adds new, artificial minority examples by extrapolating between preexisting minority instances rather than simply duplicating original examples. The technique first finds the k nearest neighbors of the minority class for each minority example (the paper recommends $k = 5$). The artificial examples are then generated in the direction of some or all of the nearest neighbors, depending on the amount of oversampling desired.

Han et al. presented a modification of Chawla et al.'s SMOTE technique which they call *borderline-SMOTE* [17] (BSM). BSM selects minority examples which are considered to be on the border of the minority decision region in the feature-space and only performs SMOTE to oversample those instances, rather than oversampling them all or a random subset.

Cluster-based oversampling [22] (CBOS) attempts to even out the between-class imbalance, as well as the within-class imbalance. There may be subsets of the examples of one class that are isolated in the feature-space from other examples of the same class, creating a within-class imbalance. Small subsets of isolated examples like this are called *small disjuncts*. Small disjuncts often cause degraded classifier performance, and this technique aims to eliminate them without removing data.

RUS, ROS, SM, and BSM require a user-defined parameter δ. All four sampling techniques were used with three parameters, $\delta = 35\%, 50\%$, and 65%, where the dataset after application of the sampling technique with parameter δ has δ positive class examples. For example, if a dataset has 100 positive examples and 500 negative examples, RUS with $\delta = 35\%$ results in a tranformed dataset with 100 positive examples and 187 negative examples (since $100/287 \approx \delta = 35\%$). ROS at 50% would result in a dataset with 500 positive and 500 negative examples. When performing Wilson's editing, we utilized both the weighted and unweighted (standard Euclidean) versions, and denote them WE-W and WE-E. In addition, classifiers were built with no sampling, which we denote 'NONE'. To summarize, a total of 16 sampling techniques and no sampling were done for each dataset. Our research group has put substantial effort into implementing software tools for all of the sampling techniques used in this experimentation.

References

[1] A. Asuncion, D. Newman, UCI machine learning repository <http://www.ics.uci.edu/~mlearn/MLRepository.html>, 2007.
[2] R. Barandela, R.M. Valdovinos, J.S. Sanchez, F.J. Ferri, The imbalanced training sample problem: under or over sampling? In Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition (SSPR/SPR'04), Lecture Notes in Computer Science 3138, 2004, pp. 806–814.
[3] G.E.A.P.A. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Exploration Newsletter 6 (1) (2004) 20–29.
[4] M.L. Berenson, D.M. Levine, M. Goldstein, Intermediate Statistical Methods and Applications: A Computer Package Approach, Prentice-Hall Inc., 1983.
[5] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[6] C.E. Brodley, M.A. Friedl, Identifying and eliminating mislabeled training instances, in: Proceedings of 13th National Conference on Artificial Intelligence, AAAI Press, 1996, pp. 799–805.
[7] C.E. Brodley, M.A. Friedl, Identifying mislabeled training data, Journal of Artificial Intelligence Research 11 (1999) 131–167.
[8] L. Cao, Y. Zhao, C. Zhang, Mining impact-targeted activity patterns in imbalanced data, IEEE Transactions on Knowledge and Data Engineering 20 (8) (2008) 1053–1066.
[9] N.V. Chawla, D.A. Cieslak, L.O. Hall, A. Joshi, Automatically countering imbalance and its empirical relationship to cost, Data Mining and Knowledge Discovery 17 (2) (2008) 225–252.
[10] N.V. Chawla, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, SMOTE: synthetic minority oversampling technique, Journal of Artificial Intelligence Research (16) (2002) 321–357.
[11] C. Drummond, R.C. Holte, C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling, in: Workshop on Learning from Imbalanced Data Sets II, International Conference on Machine Learning, 2003.
[12] W. Fan, S.J. Stolfo, J. Zhang, P.K. Chan, AdaCost: misclassification cost-sensitive boosting, in: Proceedings of the 16th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1999, pp. 97–105.
[13] Y. Feng, Z. Wu, Z. Zhou, Enhancing reliability throughout knowledge discovery process, in: Sixth IEEE International Conference on Data Mining – Reliability Issues in Knowledge Discovery Workshop (RIKD06), 2006, pp. 754–758.
[14] N.E. Fenton, S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, second ed., PWS Publishing Company, ITP, Boston, MA, 1997.
[15] A. Folleco, T.M. Khoshgoftaar, J. Van Hulse, L. Bullard, Identifying learners robust to low quality data, in: Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI 2008), Las Vegas, NV, 2008, pp. 190–195.
[16] D. Gamberger, N. Lavrač, C. Groselj, Experiments with noise filtering in a medical domain, in: Proceedings of the 16th International Conference on Machine Learning, Morgan Kaufmann, 1999, pp. 143–153.
[17] H. Han, W.Y. Wang, B.H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: International Conference on Intelligent Computing (ICIC'05). Lecture Notes in Computer Science 3644, Springer-Verlag, 2005, pp. 878–887.
[18] D.J. Hand, Good practice in retail credit scorecard assessment, Journal of the Operational Research Society 56 (2005) 1109–1117.
[19] N. Japkowicz, Learning from imbalanced data sets: a comparison of various strategies, in: AAAI Workshop on Learning from Imbalanced Data Sets (AAAI'00), 2000, pp. 10–15.
[20] N. Japkowicz, Class imbalances: are we focusing on the right issue? In Workshop on Learning from Imbalanced Datasets II, ICML, 2003.
[21] N. Japkowicz, S. Stephan, The class imbalance problem: a systematic study, Intelligent Data Analysis 6 (5) (2002) 429–450.
[22] T. Jo, N. Japkowicz, Class imbalances versus small disjuncts, SIGKDD Explorations 6 (1) (2004) 40–49.
[23] M.V. Joshi, V. Kumar, R.C. Agarwal, Evaluating boosting algorithms to classify rare classes: comparison and improvements, in: Proceedings of IEEE International Conference on Data Mining, November 2001, pp. 257–264.
[24] T.M. Khoshgoftaar, E.B. Allen, Classification of fault-prone software modules: prior probabilities, costs and model evaluation, Empirical Software Engineering 3 (1998) 275–298.
[25] T.M. Khoshgoftaar, C. Seiffert, J. Van Hulse, A. Napolitano, A. Folleco, Learning with limited minority class data, in: Proceedings of the Sixth IEEE International Conference on Machine Learning and Applications (ICMLA'07), IEEE Computer Society, Cincinnati, OH, 2007, pp. 348–353.
[26] T.M. Khoshgoftaar, N. Seliya, Comparative assessment of software quality classification techniques: an empirical case study, Empirical Software Engineering Journal 9 (2) (2004) 229–257.
[27] T.M. Khoshgoftaar, N. Seliya, The necessity of assuring quality in software measurement data, in: Proceedings of 10th International Software Metrics Symposium, IEEE Computer Society, Chicago, IL, September 2004, pp. 119–130.
[28] T.M. Khoshgoftaar, N. Seliya, K. Gao, Detecting noisy instances with the rule-based classification model, Intelligent Data Analysis: An International Journal 9 (4) (2005) 347–364.
[29] T.M. Khoshgoftaar, S. Zhong, V. Joshi, Enhancing software quality estimation using ensemble-classifier based noise filtering, Intelligent Data Analysis: An International Journal 9 (1) (2005) 3–27.
[30] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: one sided selection, in: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 179–186.
[31] D.-F. Li, W.-C. Hu, W. Xiong, J.-B. Yang, Fuzzy relevance vector machine for learning from unbalanced data and noise, Pattern Recognition Letter 29 (9) (2008) 1175–1181.

[32] R.J.A. Little, D.B. Rubin, Statistical Analysis with Missing Data, 2nd ed., John Wiley and Sons, Hoboken, NJ, 2002.
[33] M. Maloof, Learning when data sets are imbalanced and when costs are unequal and unknown, in: Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data Sets, 2003.
[34] R. Prati, G. Batista, M. Monard, Learning with class skews and small disjuncts, in: XVIIth Brazilian Symposium on Artificial Intelligence (SBIA'04). Lecture Notes in Computer Science 3171, Springer-Verlag, 2004, pp. 296–306.
[35] F. Provost, T. Fawcett, R. Kohavi, The case against accuracy estimation for comparing induction algorithms, in: Proceedings of the 15th International Conference on Machine Learning (IMLC-98), 1998.
[36] S. Ramasway, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: Proceedings of ACM SIGMOD Conference on Management of Data, ACM, 2000, pp. 427–438.
[37] SAS Institute, SAS/STAT User's Guide. SAS Institute Inc., 2004.
[38] Y. Sun, M.S. Kamel, A.K.C. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognition 40 (12) (2007) 3358–3378.
[39] C.M. Teng, Correcting noisy data, in: Proceedings of Sixth International Conference Machine Learning (ICML 99), Morgan Kaufmann, 1999, pp. 239–248.
[40] L. Thomas, D. Edelman, J. Crook, Credit scoring and its applications, SIAM Monographs on Mathematical Modeling and Computation (2002).
[41] J. Van Hulse, T.M. Khoshgoftaar, Class noise detection using frequent itemsets, Intelligent Data Analysis: An International Journal 10 (6) (2006) 487–507.
[42] J. Van Hulse, T.M. Khoshgoftaar, H. Huang, The pairwise attribute noise detection algorithm. Knowledge and Information Systems Journal, Special Issue on Mining Low Quality Data 11(2) (2007) 171–190.
[43] J. Van Hulse, T.M. Khoshgoftaar, A. Napolitano, Experimental perspectives on learning from imbalanced data, in: Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007), Corvalis, OR, June 2007, pp. 935–942.
[44] J. Van Hulse, T.M. Khoshgoftaar, A. Napolitano, Skewed class distributions and mislabeled examples, in: Proceedings of the Seventh IEEE International Conference on Data Mining – Workshops (ICDM'07), Omaha, NE, 2007, pp. 477–482.
[45] G. Weiss, K. McCarthy, B. Zabar, Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? in: Proceedings of the 2007 International Conference on Data Mining, CSREA Press, Las Vegas, NV, USA, 2007, pp. 35–41.
[46] G.M. Weiss, Mining with rarity: a unifying framework, SIGKDD Explorations 6 (1) (2004) 7–19.
[47] G.M. Weiss, F. Provost, Learning when training data are costly: the effect of class distribution on tree induction, Journal of Artificial Intelligence Research (19) (2003) 315–354.
[48] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufmann, San Francisco, CA, 2005.
[49] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, Boston, MA, 2000.
[50] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study of their impacts, Artificial Intelligence Review 22 (3–4) (2004) 177–210.
[51] X. Zhu, X. Wu, Cost-guided class noise handling for effective cost-sensitive learning, in: Fourth IEEE International Conference on Data Mining (ICDM 2004), November 2004, pp. 297–304.
[52] L. Zhuang, H. Dai, Reducing performance bias for unbalanced text mining, in: Sixth IEEE International Conference on Data Mining – Reliability Issues in Knowledge Discovery Workshop (RIKD06), 2006, pp. 770–774.

Jason Van Hulse received the Ph.D. degree in Computer Engineering from the Department of Computer Science and Engineering at Florida Atlantic University in 2007, the M.A. degree in Mathematics from Stony Brook University in 2000, and the B.S. degree in Mathematics from the University at Albany in 1997. 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, and is a member of the IEEE, IEEE Computer Society, and ACM. He has worked in the data mining and predictive modeling field at First Data Corp. since 2000, and is currently Vice President, Decision Science. He is also a Research Assistant Professor of Computer Science and Engineering at Florida Atlantic University.

Taghi M. Khoshgoftaar is a professor of the Department of Computer Science and Engineering, Florida Atlantic University and the Director of the Empirical Software Engineering Laboratory and Data Mining and Machine Learning Laboratory. 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 400 refereed papers in these areas. He is a member of the IEEE, IEEE Computer Society, and IEEE Reliability Society. He was the program chair and 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 (2008). He is the general chair of the 21st International Conference on Software Engineering and Knowledge Engineering (2009). 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, and 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.