Information Sciences 259 (2014) 571–595

Contents lists available at ScienceDirect

Information Sciences

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

An empirical study of the classification performance of learners on imbalanced and noisy software quality data

Chris Seiffert, Taghi M. Khoshgoftaar [*], Jason Van Hulse, Andres Folleco

Florida Atlantic University, Boca Raton, FL 33431, USA

ARTICLE INFO

Article history:
Available online 9 January 2011

Keywords:
Imbalance
Class noise
Sampling
Binary classification

ABSTRACT

Data mining techniques are commonly used to construct models for identifying software modules that are most likely to contain faults. In doing so, an organization's limited resources can be intelligently allocated with the goal of detecting and correcting the greatest number of faults. However, there are two characteristics of software quality datasets that can negatively impact the effectiveness of these models: class imbalance and class noise. Software quality datasets are, by their nature, imbalanced. That is, most of a software system's faults can be found in a small percentage of software modules. Therefore, the number of fault-prone, *fp*, examples (program modules) in a software project dataset is much smaller than the number of not fault-prone, *nfp*, examples. Data sampling techniques attempt to alleviate the problem of class imbalance by altering a training dataset's distribution. A program module contains class noise if it is incorrectly labeled. While several studies have been performed to evaluate data sampling methods, the impact of class noise on these techniques has not been adequately addressed. This work presents a systematic set of experiments designed to investigate the impact of both class noise and class imbalance on classification models constructed to identify fault-prone program modules. We analyze the impact of class noise and class imbalance on 11 different learning algorithms (learners) as well as 7 different data sampling techniques. We identify which learners and which data sampling techniques are most robust when confronted with noisy and imbalanced data.

1. Introduction

One of the main goals of any software engineering project is to deliver a product which is of the highest possible quality given limited time and resources. Fault detection and correction is an essential step in delivering a high quality software product, but this process can be very expensive if resources are not intelligently allocated. Data mining techniques can be used to identify software modules that are most likely to contain faults [25,23]. By distinguishing between *fault-prone* (*fp*) and *not fault-prone* (*nfp*) modules, resources can be intelligently allocated with the goal of detecting and correcting as many faults as possible. However, there are two characteristics common among software quality datasets that can hinder our ability to distinguish between *fp* and *nfp* (class variable) modules: class imbalance and class noise.

In this study, we consider only the binary classification problem. That is, we consider only two possible classes of software modules: *fp* or *nfp*. In a binary classification problem, a dataset is considered imbalanced if one class appears more frequently than the other. Class imbalance can make it significantly more difficult to detect examples (program modules) belonging to

[*] Corresponding author. Address: Data Mining and Machine Learning Laboratory, Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA. Tel.: +1 561 297 3994; fax: +1 561 297 2800.
E-mail address: taghi@cse.fau.edu (T.M. Khoshgoftaar).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.12.016

the underrepresented class. This imbalance can be severe in the case of software quality data, since a small percentage of software modules often contain a large percentage of the faults. For example, in the dataset on which this study is based, 16% of the modules contain 90% of the faults. The underrepresented class is called the minority class, while the more frequently occurring class is called the majority class. In the domain of software quality, the minority class (*fp*) is also called the *positive* class since it is the class we wish to detect, while the majority class (*nfp*) is called the *negative* class.

A related problem is that the class of interest often has a higher misclassification cost than that of the majority class — in other words, incorrectly predicting an instance that actually belongs to the minority class as a majority class example is more costly than incorrectly predicting a majority class example as belonging to the minority class. This holds true in the domain of software quality where misclassifying an *fp* module as *nfp* is more expensive than the reverse, since this type of misclassification is likely to cause faults to go undetected.

Several *data sampling* (or resampling) techniques have been proposed to alleviate the problem of class imbalance. These techniques attempt to reduce the severity of imbalance within the data by removing examples from the majority class, or adding examples to the minority class. Some techniques do so randomly, while others attempt to "intelligently" augment or subtract from the dataset in a clever way to benefit the classifier.

Incorrect values in a dataset are called noise. Noise, especially when it occurs in the class attribute (class noise) can have a negative impact on classification performance [27,7]. An instance contains class noise if the true class of the instance is different from the recorded class. The most basic strategy for dealing with noise is to use a classification algorithm that is robust [17].

In this study, we examine the impact of noise and imbalance on a variety of classification algorithms including decision trees, nearest neighbors, neural networks and Bayesian learners. The objective of this work is to analyze the relationship between classification performance, data sampling, learner selection, class imbalance and class noise. From a real-world software quality dataset, we derive 12 datasets with different levels of noise and imbalance. We apply 11 classification algorithms to these datasets combined with seven different sampling techniques, varying sampling technique parameters when applicable. We examine the interaction between the choice of classifier and sampling technique on each of the 12 derived datasets. Each classifier/sampling technique combination was evaluated using 10 runs of 10-fold cross validation. In total, 1,267,200 classifiers were built to produce the results presented in this work.

In this work, we address the following research questions:

- What is the relative impact of class noise vs. class imbalance? Which has a more severe impact on the performance of the different classification algorithms and sampling techniques?
- How do different classification algorithms react to the application of different sampling techniques? Are some classifiers more significantly improved by the use of sampling techniques? Do certain sampling techniques work better when used in conjunction with specific classification algorithms?
- What benefit do sampling techniques provide at different levels of class imbalance and noise? When the data is highly imbalanced or very noisy, do certain sampling techniques perform better than others?
- How do classification algorithms perform at different levels of class imbalance and noise after sampling techniques have been applied to the data? When the data is highly imbalanced or very noisy, do certain classification algorithms perform better than others?

The results of our experiments show two sampling techniques, Wilson's editing and random undersampling, perform very well. Some learners are impacted by the use of sampling, while others are unaffected. Generally speaking, class noise is also shown to have a more significant impact on learners than imbalance. Section 4 presents much more detailed analysis, along with the supporting results from our experimentation.

2. Related work

Sampling techniques have received significant attention in recent research, however most of this research does not take the quality of data into consideration. Drummond and Holte [13] found that majority undersampling is more effective at dealing with the imbalance problem using C4.5, but Maloof's research [30] shows that undersampling and oversampling produce roughly equivalent classifiers using Naive Bayes and C5.0 (the commercial successor to C4.5). Weiss and Provost [39] examine the impact of class distribution, finding that the ideal class distribution is dependent on domain, though in general the natural distribution yields the best overall accuracy, while a balanced distribution results in the highest AUC. Barandela et al. [3] and Han et al. [18] examine the performance of more "intelligent" data sampling techniques such as SMOTE, Borderline SMOTE, and Wilson's Editing. Weiss and Provost [39], Japkowicz and Stephan [21], Elkan [14] and Kolcz et al. [28] study the impact of sampling on different classifiers finding that some can benefit greatly by using data sampling techniques while other classifiers are relatively unaffected by sampling. Other methods have also been proposed to handle class imbalance [10].

In addition, numerous studies have been performed to analyze the impact of noise and countless techniques have been proposed to cope with it. Zhu and Wu [43] show that noise, especially class noise, can negatively impact classification performance. Three types of techniques have been proposed to alleviate the problem of noise in data. The first is to use a robust classification algorithm [17]. A classification algorithm is said to be robust in the presence of noise if it is able to maintain a high classification accuracy despite the low quality of data. Another common technique to overcome the problem of noisy data is to apply a filter designed to remove noisy instances from the dataset. For example, Brodley and Friedl present an ensemble classification technique to remove noisy instances from the data [7]. The third technique is to attempt to correct the noisy data. Noise correction is often considered superior to filtering, especially in small datasets where the removal of records could result in the loss of important data which could be beneficial to the learning process. Teng proposed *Polishing* [38] to accomplish this goal. In this study, we focus only on the first technique. We attempt to identify which classification algorithms are most robust in the presence of class noise, especially when data is imbalanced. Classifiers are used both with and without data sampling. Future work will investigate the effect of noise filtering/correcting techniques when applied to imbalanced data. In addition, our research group has published many works in this area including the application of a hybrid cleansing procedure [26] which can be used to identify noise instances and cleanse noisy datasets. While our research group has performed many recent experiments in this area, we are unaware of any external work that studies the combined effects of both noise and imbalance in such detail as we present in this study.

3. Experimental design

3.1. CCCS dataset

The CCCS dataset used in our experiments is a large military command, control and communications system written in Ada [23]. CCCS contains 282 instances (program modules), where each instance is an Ada package consisting of one or more procedures. CCCS contains eight software metrics which are used as independent variables or attributes. An additional attribute, *nfaults* (the dependent variable), indicates the number of faults attributed to a module during the system integration and test phases, and during the first year of deployment. Table 1 lists the software metrics in the CCCS dataset. Additional information about the software metrics used in this study can be found in Fenton and Pfleeger [15]. The intent of this study is not to evaluate the utility of these specific software metrics, and other studies may choose to utilize additional or alternative metrics. For simplicity, CCCS will be denoted $\mathcal{C}^o$ in these experiments.

A histogram is provided in Fig. 1, with *nfaults* on the *x*-axis, while the *y*-axis is the percentage of program modules with a particular value for *nfaults*. For example, over 50% of the program modules have no faults (i.e., *nfaults* = 0), and approximately 19% of the program modules contained exactly one fault. The median value of *nfaults* is 0, the largest value is 42 and the mean is 2.369.

3.2. Cleansing CCCS

Since CCCS is a real-world dataset, $\mathcal{C}^o$ contains some instances which have a noisy value for *nfaults*, so-called *inherent* noise, which can be attributed to many different factors (for example, a faulty data collection process resulting in transcription errors of manually recorded attribute values). *Injected* noise, on the other hand, is noise that is artificially introduced into the dataset, primarily for use in experimentation, by replacing the original attribute values for some of the instances with noisy, synthetic values. In other words, inherent noise is naturally occurring in the dataset while injected noise is artificially introduced by the researcher strictly for experimental purposes.

When analyzing the impact of noise, the distinction between inherent and injected noise is very important. It may be difficult to inject noise that represents the types of noise typically found in a dataset from the application domain. Generating realistic examples of noise in a domain-sensitive manner is a difficult research issue, and is critically important because measuring the results of any technique using unrealistic noise can be misleading since it may not represent the types of noise found in real-world domain specific datasets. On the other hand, inherent noise is representative of the noise typically found in a domain-specific dataset, but the use of a domain expert is required to identify the noisy instances in the dataset since they are not known beforehand. This can be a time-consuming and tedious task since it requires the manual inspection of the

Table 1
CCCS dataset software metrics.

Independent variables
Logical operators
Total lines of code
Executable lines of code
Unique operands
Total operands
Unique operators
Total operators
Cyclomatic complexity
Dependent variable
nfaults

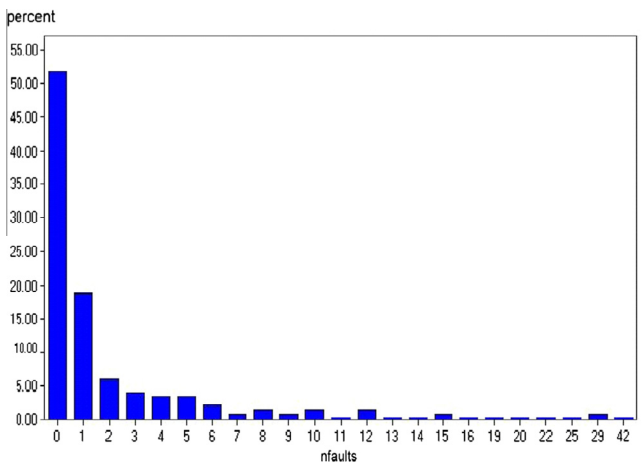

Fig. 1. Distribution of *nfaults* in the CCCS dataset.

data. Our experiments utilize both inherent and injected noise, which is unique among related work and further demonstrates the breadth of our experiments. In this respect $\mathcal{C}^o$ is an ideal dataset because it is of reasonable size (282 instances, 8 attributes and 1 dependent variable), allowing for a detailed inspection by a domain expert. In addition to having considerable software engineering domain experience, the expert employed in this study has worked for many years with the CCCS dataset and therefore has a deep understanding of the data. The expert employed in these experiments also has considerable knowledge and experience using a variety of statistical and data analysis tools.

With the help of a hybrid procedure introduced for cleansing noise from a continuous dependent variable [26], a software engineering expert oversaw the cleansing of $\mathcal{C}^o$. The hybrid procedure, under the supervision of a domain expert, also determined a clean value for *nfaults* (denoted $nfaults^c$) for the instances deemed to be noisy. A total of 81 instances were identified as containing inherent noise in the dependent variable. For these 81 instances, the original value for *nfaults* that was in the dataset is noisy and is denoted $nfaults^n$. A detailed description of the cleansing process is presented in our previous work [26].

3.3. Injected noise

Two additional datasets were derived from $\mathcal{C}^o$, denoted $\mathcal{C}^{5p}$ and $\mathcal{C}^{10p}$. Note that all three of these datasets have the same number of instances (282) and the same number of attributes (9). These datasets were created as follows:

- $\mathcal{C}^{5p}$: Starting with the original dataset $\mathcal{C}^o$, the software engineering expert inspected the dataset and identified 14 instances (or 5% of the instances in $\mathcal{C}^o$) that were relatively clean. These 14 instances were then corrupted (with respect to *nfaults*) in an expert-supervised manner such that the noise was reasonable for the given dataset and that the noisy value was different than the clean value.
- $\mathcal{C}^{10p}$: In addition to the 14 clean instances with injected noise in the dependent variable in $\mathcal{C}^{5p}$, the expert identified 14 more relatively clean instances for corruption. Noise was injected in a manner similar to that of the first 14 instances, resulting in the $\mathcal{C}^{10p}$ dataset. $\mathcal{C}^{10p}$ therefore has 28 instances (or 10% of the dataset) with injected noise in *nfaults*.

$\mathcal{C}^o$, $\mathcal{C}^{5p}$ and $\mathcal{C}^{10p}$ each have 81 inherently noisy examples with respect to *nfaults*. For these 81 instances both the noisy value $nfaults^n$ and clean value $nfaults^c$ are known. For the injected noise in datasets $\mathcal{C}^{5p}$ and $\mathcal{C}^{10p}$, $nfaults^c$ is the value that was originally in the dataset, and $nfaults^n$ is the corrupted value. For the remaining examples in each of these three datasets, $nfaults^n = nfaults^c$. Therefore, both a noisy value $nfaults^n$ and a clean value $nfaults^c$ are available for all examples of each dataset.

3.4. Level of imbalance

The second factor considered in our experiments is the level of imbalance in the class attribute. From $\mathcal{C}^o$, $\mathcal{C}^{5p}$ and $\mathcal{C}^{10p}$, a total of 12 new datasets with a binary class L are derived. Let $\mathcal{C}^*$ denote any of the three initial datasets, and let $\lambda \in \{4, 6, 8, 12\}$ denote a threshold on the dependent variable *nfaults* in $\mathcal{C}^*$. $\mathcal{C}^*$ is transformed to the dataset $\mathcal{C}^*_\lambda$ by replacing *nfaults* with a binary class attribute, L. $L(x)$ identifies an instance x as either *fault-prone* (*fp*) or *not fault-prone* (*nfp*) according to the following rules:

$$L^c_\lambda(x) = \begin{cases} nfp & \text{If } nfaults^c(x) < \lambda \\ fp & \text{otherwise} \end{cases}$$

$$L^n_\lambda(x) = \begin{cases} nfp & \text{If } nfaults^n(x) < \lambda \\ fp & \text{otherwise} \end{cases}$$

$L^n_\lambda(x)$ is the class of instance x based on the noisy value $nfaults^n(x)$, while $L^c_\lambda(x)$ is the class of x using the clean value $nfaults^c(x)$. From the perspective of software quality prediction, instances are generally divided into two classes, fault-prone (*fp*) and not fault-prone (*nfp*). In general, *fp* is the positive or minority class, while *nfp* is the negative or majority class. Since the purpose of software quality prediction is to identify modules that are likely to contain faults, classifiers are constructed with the objective of detecting as many *fp* instances as possible in a dataset while incorrectly detecting as few *nfp* instances as possible. Resources can then be directed toward at-risk program modules.

Since an instance x is labeled as *fp* only when $nfaults(x) \geqslant \lambda$, increasing the value of λ reduces the number of instances labeled as *fp* in the dataset. Therefore, increasing the value of λ also increases the level of imbalance in L. Four values of λ are used in our experiments, resulting in four levels of imbalance. Therefore, in our experiments a total of 12 derived datasets were created. $\mathcal{C}^o_4$, $\mathcal{C}^o_6$, $\mathcal{C}^o_8$ and $\mathcal{C}^o_{12}$ are the four binary classification datasets derived from $\mathcal{C}^o$ using the threshold $\lambda \in \{4, 6, 8, 12\}$. $\mathcal{C}^{5p}_4$, $\mathcal{C}^{5p}_6$, $\mathcal{C}^{5p}_8$ and $\mathcal{C}^{5p}_{12}$ were derived from dataset $\mathcal{C}^{5p}$, while $\mathcal{C}^{10p}_4$, $\mathcal{C}^{10p}_6$, $\mathcal{C}^{10p}_8$ and $\mathcal{C}^{10p}_{12}$ were derived from dataset $\mathcal{C}^{10p}$.

Table 2 shows the severity of imbalance in the 12 derived datasets. The first set of four datasets in this table is derived from $\mathcal{C}^o$. As the threshold (λ) increases from 4 to 12, the percentage of minority class examples in the data (%*fp*) decreases from 19.5% to 5.67%, demonstrating an increase in the severity of imbalance. This also holds true for datasets $\mathcal{C}^{5p}$ and $\mathcal{C}^{10p}$. Similar information is provided in Table 3, which provides the number of *fp* and *nfp* instances in the data with respect to L^c, the clean class value. Note that since this is based on L^c, the number of *fp* and *nfp* are the same for datasets with a given λ.

3.5. Class noise

The number of noisy instances in the datasets prior to the transformation of *nfaults* to L is not necessarily the same as the number of noisy instances in the post-transformation datasets. In fact, the number of noisy instances is much lower since an incorrect *nfaults* label does not guarantee an incorrect L value. Specifically, an L value will only be incorrect, or noisy, if the value of λ falls between $nfaults^n$ and $nfaults^c$. For example, an instance with $nfaults^n = 7$ and $nfaults^c = 10$ will not be identified as noisy for $\lambda = 6$. Since both $nfaults^n$ and $nfaults^c$ are greater than 6, the instance will be correctly labeled as *fp* even though $nfaults^n \neq nfaults^c$. However, if $\lambda = 8$ this same instance will be incorrectly labeled as *nfp* (since $nfaults^n < \lambda$) when it should be

Table 2
Dataset characteristics: level of imbalance in L^n.

Dataset	#*fp*	#*nfp*	%*fp*	%*nfp*
$\mathcal{C}^o_4$	55	227	19.5	80.5
$\mathcal{C}^o_6$	35	247	12.41	87.59
$\mathcal{C}^o_8$	27	255	9.57	90.43
$\mathcal{C}^o_{12}$	16	266	5.67	94.33
$\mathcal{C}^{5p}_4$	63	219	22.34	77.66
$\mathcal{C}^{5p}_6$	43	239	15.25	84.75
$\mathcal{C}^{5p}_8$	33	249	11.7	88.3
$\mathcal{C}^{5p}_{12}$	18	264	6.38	93.62
$\mathcal{C}^{10p}_4$	73	209	25.89	74.11
$\mathcal{C}^{10p}_6$	53	229	18.79	81.21
$\mathcal{C}^{10p}_8$	38	244	13.48	86.52
$\mathcal{C}^{10p}_{12}$	22	260	7.8	92.2

Table 3
Dataset characteristics: level of imbalance in L^c.

Dataset	#*fp*	#*nfp*	%*fp*	%*nfp*
$\mathcal{C}^*_4$	56	226	19.86	80.14
$\mathcal{C}^*_6$	35	247	12.41	87.59
$\mathcal{C}^*_8$	27	255	9.57	90.43
$\mathcal{C}^*_{12}$	19	263	6.74	93.26

labeled as *fp* (since $nfaults^c \geqslant \lambda$). Finally, this instance would be correctly labeled as *nfp* for $\lambda = 12$, since both $nfaults^n$ and $nfaults^c$ are less than 12.

Table 4 shows the actual quantity of noise in each of the 12 derived datasets. Noisy instances are denoted as $x \to y$, where x and y indicate whether L^c and L^n, respectively, belong to the negative (n) or positive (p) class. The table provides both the number of noisy instances and the percentage of instances containing noise. The columns $\#n \to p$, $\#p \to n$ and $\#noise$ show the number of false positives (instances that should be labeled *nfp* but are incorrectly labeled *fp*, i.e., $nfaults^n \geqslant \lambda$ and $nfaults^c < \lambda$), the number of false negatives (instances that should be labeled *fp* but are incorrectly labeled *nfp*, i.e., $nfaults^n < \lambda$ and $nfaults^c \geqslant \lambda$) and the total number of noisy instances ($\#n \to p + \#p \to n$), respectively. Column $\%n \to p/p$ shows the percentage of instances labeled *fp* that should have been labeled *nfp*. Column $\%n \to p/n$ shows the percentage of instances with $L^c = nfp$, but $L^n = fp$. Column $\%p \to n/n$ shows the percentage of instances labeled *nfp* that should have been labeled *fp*. Column $\%p \to n/p$ shows the percentage of instances with $L^c = fp$, but $L^n = nfp$. That is:

$$\%n \to p/p = \frac{\#n \to p}{\#p \to p + \#n \to p} \times 100 \tag{1}$$

$$\%n \to p/n = \frac{\#n \to p}{\#n \to n + \#n \to p} \times 100 \tag{2}$$

$$\%p \to n/n = \frac{\#p \to n}{\#n \to n + \#p \to n} \times 100 \tag{3}$$

$$\%p \to n/p = \frac{\#p \to n}{\#p \to p + \#p \to n} \times 100 \tag{4}$$

Finally, column *%noise* shows the percentage of all instances that are incorrectly labeled. For example, dataset C^o_{12} has 15 noisy instances, 6 of which are incorrectly labeled as *fp* and 9 of which are incorrectly labeled as *nfp*. These 15 instances represent 5.32% of the entire dataset. The 6 false positives represent 37.5% of instances with $L^n = fp$ and 2.28% of the instances with $L^c = nfp$. The 9 false negatives represent 3.38% of the instances with $L^n = nfp$ and 47.37% of the instances with $L^c = fp$.

3.6. Dataset characteristics

It is important to note that as we change the level of noise in our data ($\mathcal{C}^o, \mathcal{C}^{5p}, \mathcal{C}^{10p}$), the severity of imbalance in L^n also changes. However, the imbalance in L^c remains the same. Similarly, as we change the level of imbalance ($\lambda = 4, 6, 8, 12$), the amount of noise changes as well. One must keep this in mind when analyzing the effects of noise and imbalance on classification performance.

A change in the severity of imbalance within the class label, L, as we vary the level of $noise(\mathcal{C}^o, \mathcal{C}^{5p}, \mathcal{C}^{10p})$ is shown in Table 2 and is summarized in Fig. 2. For simplicity, datasets $\mathcal{C}^o$, $\mathcal{C}^{5p}$, $\mathcal{C}^{10p}$ are simply represented as O, 5P and 10P in many tables throughout this paper. Fig. 2 shows the change in *%fp* (the percentage of instances that are labeled *fp*) for the three different levels of noise. For each dataset, C^*_4, C^*_6, C^*_8 and C^*_{12}, as the level of noise increases, *%fp* also increases. Since *fp* is the minority class, an increase in *%fp* results in a decrease in the severity of imbalance within L. Note that this is an increase in the percentage of instances that are labeled *fp*, not the percentage of instances where the $L^c = fp$. The number of instances where $L^c = fp$ does not change with the increase in noise level, as shown in Table 3. Only L^n changes with noise level.

A change in the amount of actual noise in L as we vary the level of imbalance ($\lambda = 4, 6, 8, 12$) can be seen in Table 4 and is summarized in Fig. 3. Fig. 3 shows that as we increase the level of imbalance, the percentage of noisy instances in our data decreases. This figure shows only the change in overall noise, however the change in the number of false positives and false

Table 4
Dataset characteristics: level of noise.

Dataset	$\#n \to p$	$\#p \to n$	$\#noise$	$\%n \to p/p$	$\%n \to p/n$	$\%p \to n/n$	$\%p \to n/p$	$\%noise$
C^o_4	14	15	29	25.45	6.19	6.61	26.79	10.28
C^o_6	8	8	16	22.86	3.24	3.24	22.86	5.67
C^o_8	8	8	16	29.63	3.14	3.14	29.63	5.67
C^o_{12}	6	9	15	37.5	2.28	3.38	47.37	5.32
C^{5p}_4	24	17	41	38.1	10.62	7.76	30.36	14.54
C^{5p}_6	18	10	28	41.86	7.29	4.18	28.57	9.93
C^{5p}_8	15	9	24	45.45	5.88	3.61	33.33	8.51
C^{5p}_{12}	9	10	19	50	3.42	3.79	52.63	6.74
$C_{4^{10p}}$	36	19	55	49.32	15.93	9.09	33.93	19.5
C^{10p}_6	30	12	42	56.6	12.15	5.24	34.29	14.89
C^{10p}_8	22	11	33	57.89	8.63	4.51	40.74	11.7
C^{10p}_{12}	13	10	23	59.09	4.94	3.85	52.63	8.16

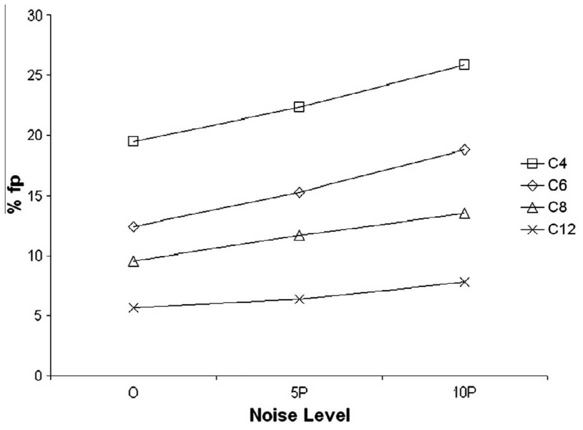

Fig. 2. Effect of changing the level of *noise* on the severity of class imbalance.

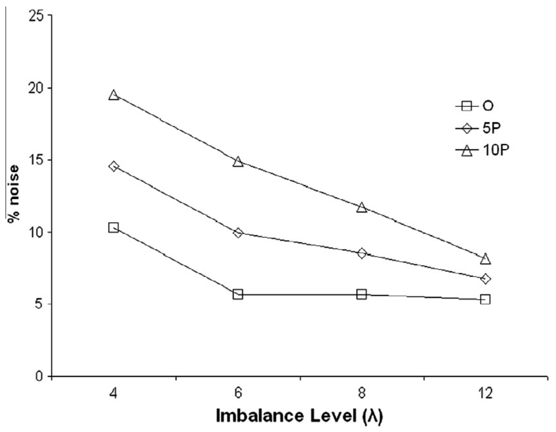

Fig. 3. Effect of changing imbalance level (λ) on %noise.

negatives can be seen in Table 4. These relationships between noise and imbalance within the datasets have a significant role in our results and analysis.

3.7. Learners

This section provides brief descriptions of the 11 classification algorithms along with an explanation of the parameters used in our experiments. For additional information, please see the referenced works. These classifiers were considered because they are the most commonly used in class imbalance research, and they are widely known in the data mining community. All learners were built using the WEKA machine learning tool [41]. Unless otherwise specified, the default parameters (as provided by WEKA) were used.

We use two versions of Quinlan's popular *C4.5* decision tree learner [34], denoted C4.5(D) and C4.5(N). The C4.5 algorithm constructs a decision tree using an entropy-based splitting criteria stemming from information theory [1]. C4.5(D) uses the default parameters provided by WEKA. C4.5(N) disables pruning and enables Laplace smoothing [39]. Breiman's *Random Forest* classifier [6] (RF) utilizes bagging [5] as well as the 'random subspace method' [20] in order to construct an ensemble of randomized decision trees. Two versions of the *k-nearest neighbors technique* (kNN) [2] were also used. This technique classifies examples by finding the *k* examples that are closest to it in feature space. We used two values of *k*, 2 and 5, and the resulting learners are denoted 2NN and 5NN, respectively. For these kNN learners, the 'distanceWeighting' parameter was set to 'Weight by 1/distance.' *Repeated Incremental Pruning to Produce Error Reduction* (RIPPER) is a rule-based learner proposed by Cohen [11] that improved upon the IREP algorithm [16] in order to improve accuracy without sacrificing efficiency. *Logistic regression* (LR) is a statistical regression model for categorical prediction [19,24]. *Naive Bayes* (NB) [31] is a simple classifier that utilizes Bayes's rule of conditional probability. Although NB assumes all predictor variables to be conditionally independent, NB has been shown to perform well even in the presence of strong attribute dependencies [12].

Multilayer Perceptron (MLP) [35] attempts to artificially mimic the functioning of a biological nervous system. Two MLP parameters were changed from their 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 reserve 10% of the training data as a validation set to determine when to stop the iterative training process. Radial basis function networks [32] are, like Multilayer Perceptron, a type of artificial neural network. The only parameter change for RBF was to set the parameter 'numClusters' to 10. A *support vector machine* (SVM) [37] is a linear classifier which attempts to learn the maximum margin hyperplane separating two classes in the data. Two changes to the default parameters were made: the complexity constant 'c' was changed from 1.0 to 5.0, and the 'buildLogisticModels' parameter, which allows proper probability estimates to be obtained, was set to 'true'.

3.8. Sampling techniques

Seven data sampling techniques were used in this study. This section contains a brief description of each, along with the parameters used. We examine a wide variety of data sampling techniques including both random and intelligent versions of under and over sampling. In addition, all experiments were performed without data sampling and are denoted 'None'.

The two simplest data sampling techniques used in this study are *random undersampling* (RUS) and *random oversampling* (ROS). Minority oversampling randomly duplicates examples of the minority class, while majority undersampling randomly discards examples of the majority class. Each of these techniques have their benefits and drawbacks. For example, oversampling may lead to overfitting [9] while undersampling may result in the loss of valuable information.

In addition to random sampling, many more 'intelligent' data sampling techniques are also investigated. Two intelligent undersampling techniques are used: *One-Sided Selection* [29] (OSS) attempts to intelligently undersample the majority class by removing examples which are considered either redundant or 'noisy'. *Wilson's editing* [3] (WE) is another intelligent under sampling technique based on Wilson's technique for pruning datasets [40]. WE undersamples the majority class by removing majority class examples that are misclassified using a *k*-nearest neighbors learner.

In addition, we investigate the performance of three intelligent oversampling techniques. Cluster-Based Oversampling [22] (CBOS) attempts to even out both between-class and within-class imbalance. Using this technique, the minority and majority class examples are separated into clusters, and each cluster (except the largest) is randomly oversampled with replacement until they have the same number of examples as the largest majority class cluster. Synthetic Minority Oversampling Technique [8] or SMOTE (SM) adds new minority class examples to the dataset. This is done by finding the *k*-nearest neighbors of minority examples and extrapolating between these neighbors. Borderline-SMOTE [18] (BSM) performs oversampling in the same way, but favors creating new examples based on minority class examples that lie near the border of the minority decision region.

While many previous studies chose to test random sampling strategies by sampling the data until the classes had equal numbers, we chose not to do this because previous work [39] has shown that an even class distribution is not always optimal and that the optimal distribution can vary with the problem domain. Thus, for sampling techniques that could be applied with varying magnitudes, we tested a range of different values. RUS was performed at 5%, 10%, 25%, 50%, 75%, and 90% of the majority class. These values indicate the percentage of majority class examples that are removed from the training dataset. ROS, SM and BSM were performed with oversampling rates 50%, 100%, 200%, 300%, 500%, 750%, and 1000%. These values indicate how many examples (as a percentage of the minority class) are added to the training datset. When performing Wilson's editing, we utilized both the weighted and unweighted (standard Euclidean) versions. A total of 31 combinations of sampling techniques and parameters were utilized. In addition, we built a classifier with no sampling, which we denote 'None'. All of these sampling techniques were implemented in Java in the framework of the WEKA machine learning tool [41].

3.9. Evaluation metric: ROC curves

One of the most popular methods for performance evaluation of learners for class imbalance cases is *Receiver Operating Characteristic*, or *ROC* curves. *ROC* curves, graph true positive rates on the *y*-axis vs. the false positive rates on the *x*-axis. The resulting curve illustrates the trade-off between detection rate and false alarm rate. The *ROC* curve illustrates the performance of a classifier across the complete range of possible decision thresholds, and accordingly does not assume any particular misclassification costs or class prior probabilities.

For a single numeric measure, the *area under the ROC curve* (*AUC*) is widely used, providing a general idea of the predictive potential of the classifier. A higher *AUC* is better, as it indicates that the classifier, across the entire possible range of decision thresholds, has a higher true positive rate. The *AUC* is the performance metric used for this study. Provost and Fawcett [33] give an extensive overview of ROC curves and their potential use for creating optimal classifiers.

4. Empirical results

4.1. Learners and sampling

In this section, we examine the effect of the seven sampling techniques on the 11 classification algorithms. In Section 4.1.1 we examine the impact of using sampling on each of the classifiers. We identify which algorithms are most improved by performing sampling, and which are either not improved by sampling, or in some cases which classification algorithms can actually have their performance hindered by performing sampling. In section 4.1.2 we take a closer look at the individual sampling techniques and identify which sampling techniques perform best with each classification algorithm. Finally, we summarize our experimental results in Section 4.1.3.

4.1.1. Improvement over not performing sampling

Not all learners can be improved upon equally by applying sampling techniques. Tables 5 and 6 show the best improvement achieved for each learner on each dataset. Improvement is calculated as:

$$\text{improvement} = \frac{AUC^{best} - AUC^{None}}{AUC^{None}} \times 100$$

where AUC^{None} is the AUC achieved without any sampling and AUC^{best} is the AUC achieved by the best sampling/parameter combination. A negative value in Table 5 or 6 indicates that not performing sampling yielded a higher AUC than any of the sampling techniques when combined with the given learner and applied to the specified dataset. The column labeled *Average* in Table 6 shows the average over all datasets in Tables 5 and 6.

Each learner is improved by sampling when the results are averaged across all datasets. However, some are improved more than others. For example, RBF benefits greatly from sampling, especially at higher levels of imbalance. At the lowest level of imbalance ($\lambda = 4$), RBF's improvement is between 1.7% and 3.6%. At the highest level of imbalance ($\lambda = 12$), RBF's improvement is much greater, ranging from 26.49% to 38.54%. Other techniques, such as 2NN and NB do not benefit as much from sampling. For example, 2NN actually performs better without sampling than with sampling in 5 of the 12 datasets. The 5 datasets where 2NN performs better without sampling are the datasets with lower levels of imbalance ($\lambda \in \{4,6\}$). At higher levels of imbalance the performance of 2NN is enhanced by performing sampling, but in all cases the increase in AUC is less than 1%. The average improvement in 2NN performance across all datasets is only 0.1%. Unlike 2NN, the performance of NB is never hindered by the application of sampling. That is, the results are always at least as good with sampling as without. However, its average improvement in AUC (0.04%) is even smaller than 2NN. NB appears to be relatively unaffected by sampling, with a maximum improvement of 0.14% (on dataset C_4^{10p}) and an improvement of 0.01% or less on 4 of the 12 datasets. The only datasets where NB's performance is improved by more than 0.10% are those datasets with $\lambda = 4$. Other classification algorithms, such as C4.5 (both) and RIPPER show improvement when sampling is performed, but the improvement is much smaller than that of RBF.

Table 5
Improvement over not performing sampling: $\lambda = \{4,6\}$.

Learner	C_4^o (%)	C_4^{5p} (%)	C_4^{10p} (%)	C_6^o (%)	C_6^{5p} (%)	C_6^{10p} (%)
2NN	−0.31	−0.50	−0.89	0.50	−0.35	−0.66
5NN	0.68	2.30	3.35	0.27	0.33	0.45
C4.5(D)	0.43	−1.10	−1.10	6.13	3.87	6.19
C4.5(N)	2.84	1.10	−0.39	6.88	2.04	3.08
LR	2.31	0.21	−0.30	3.63	2.81	2.94
MLP	0.30	0.66	0.21	0.49	0.70	1.24
NB	0.11	0.13	0.14	0.02	0.02	0.02
RBF	1.70	2.44	3.60	7.96	19.24	24.91
RF	1.83	2.52	3.55	0.93	−0.06	1.38
RIPPER	4.42	9.31	10.38	5.44	3.21	5.55
SVM	−0.08	0.05	1.43	0.69	2.92	3.20

Table 6
Improvement over not performing sampling: $\lambda = \{8,12\}$.

Learner	C_8^o (%)	C_8^{5p} (%)	C_8^{10p} (%)	C_{12}^o (%)	C_{12}^{5p} (%)	C_{12}^{10p} (%)	Average (%)
2NN	0.43	0.68	0.97	0.15	0.42	0.77	0.10
5NN	0.28	0.39	0.70	0.33	0.23	0.29	0.80
C4.5(D)	5.90	6.21	10.25	10.49	7.04	9.26	5.30
C4.5(N)	2.87	1.94	7.32	0.49	7.61	4.66	3.37
LR	1.38	−0.39	−0.49	0.46	−0.71	−0.75	0.92
MLP	0.34	0.39	1.65	1.06	−0.22	0.36	0.60
NB	0.01	0.00	0.00	0.00	0.05	0.04	0.04
RBF	27.22	30.63	24.00	38.32	26.49	38.54	20.42
RF	0.39	0.98	2.53	0.33	−0.18	1.05	1.27
RIPPER	3.33	2.02	5.12	6.05	4.52	3.46	5.23
SVM	0.00	0.02	0.24	−0.12	−0.06	0.54	0.74

4.1.2. Learner/sampling technique pairs

Tables 5 and 6 show the improvement of each learner with sampling over the performance without applying sampling. Tables 7 and 8 identify the sampling technique that was used to achieve the results in Tables 5 and 6. For example, on the C_{12}^{o} dataset using C4.5(D), RUS achieved the best performance improvement over no sampling with a 10.49% improvement in AUC.

The *Most Common* column in Table 8 identifies the sampling technique that performed best for the given learner on the most datasets. For example, ROS performed best on 5 of the 12 datasets using C4.5(N), more than any other sampling technique. WE and RUS tied when using the learner SVM, each achieving the highest AUC in 5 of the 12 datasets. WE performed the best on 5 (2NN, 5NN, MLP, RF and SVM) of the 11 learners, while RUS performed best on 4 (C4.5(D), RBF, RIPPER and SVM) of the learners. C4.5(N) and LR were most improved by ROS. NB was improved most by SM, but as shown in Tables 5 and 6, this improvement was minimal.

Even though WE performed better on more learners than RUS, the learners where RUS performed best showed relatively more improvement than those where WE performed best. For example, WE performed best on 2NN, 5NN, MLP, RF and SVM. These learners improved by 0.10%, 0.80%, 0.60%, 1.27% and 0.74%, respectively. The average improvement across all datasets for these 5 learners is 0.7%. RUS performed best when combined with C4.5(D), RBF, RIPPER and SVM, improving the performance of these learners by 5.3%, 20.42%, 5.23% and 0.74%, respectively. The average improvement across all datasets for these 4 learners is 7.92%. So, while WE outperformed RUS for more learners, those learners where RUS performed best achieved a higher increase in AUC than those where WE outperformed RUS. Counting the number of times each sampling technique resulted in the best AUC for each learner/dataset pair, we find that RUS resulted in the best performance most frequently, 50 times. WE performed comparably with 41 first place finishes, while the next best technique (ROS) only had 14 first place finishes. Out of 132 learner/dataset combinations, CBOS never resulted in the highest AUC.

4.1.3. Summary of results

In summary, not all learners are equally affected by applying sampling techniques. In fact, some learners are actually negatively affected by sampling, especially at less severe levels of imbalance. For example, even the highest AUC achieved using sampling with 2NN was lower than the AUC achieved using no sampling in 5 of the 12 datasets. These 5 datasets were all among the 8 datasets with the lowest levels of imbalance (C_4^* and C_6^*). So, for these 8 datasets when using 2NN as the classifier, *None* (not performing sampling) outperformed all sampling techniques 62.5% of the time. Other learners, such as RBF, were greatly improved by sampling, especially at higher levels of imbalance. The improvement in AUC achieved by RBF when using the sampling technique RUS was as much as 38.54% higher than not performing sampling. NB was relatively unaffected

Table 7
Best learner/sampling technique combinations: $\lambda = \{4, 6\}$.

Learner	C_4^o	C_4^{5p}	C_4^{10p}	C_6^o	C_6^{5p}	C_6^{10p}
2NN	RUS	RUS	RUS	WE	RUS	RUS
5NN	OSS	OSS	OSS	WE	OSS	WE
C4.5(D)	SM	RUS	RUS	RUS	RUS	RUS
C4.5(N)	WE	WE	RUS	ROS	WE	SM
LR	ROS	ROS	ROS	WE	WE	WE
MLP	ROS	ROS	RUS	WE	WE	WE
NB	BSM	BSM	BSM	SM	SM	SM
RBF	SM	WE	WE	BSM	RUS	RUS
RF	WE	BSM	BSM	RUS	ROS	WE
RIPPER	OSS	RUS	RUS	RUS	RUS	RUS
SVM	RUS	WE	ROS	WE	RUS	RUS

Table 8
Best learner/sampling technique combinations: $\lambda = \{8, 12\}$.

Learner	C_8^o	C_8^{5p}	C_8^{10p}	C_{12}^o	C_{12}^{5p}	C_{12}^{10p}	Most common
2NN	WE	WE	WE	WE	WE	WE	WE
5NN	RUS	WE	WE	WE	WE	WE	WE
C4.5(D)	WE	RUS	RUS	RUS	RUS	RUS	RUS
C4.5(N)	ROS	ROS	ROS	WE	BSM	ROS	ROS
LR	ROS	ROS	RUS	RUS	BSM	RUS	ROS
MLP	WE	WE	WE	RUS	RUS	RUS	WE
NB	SM	BSM	SM	SM	BSM	SM	SM
RBF	BSM	RUS	RUS	RUS	RUS	RUS	RUS
RF	WE	WE	WE	RUS	RUS	WE	WE
RIPPER	RUS	RUS	RUS	RUS	RUS	RUS	RUS
SVM	WE	OSS	WE	RUS	RUS	WE	RUS/WE

by sampling. The performance of NB was not hindered by sampling the data, but it was not improved much either. Most classifiers fell somewhere between these extremes, but all reacted differently to different sampling techniques at different levels of noise and imbalance.

Note that these conclusions relate to the AUC of the learners. In other words, for many learners, the AUC cannot be improved through the use of sampling. In general, this implies that the ranking of examples (which is used to calculate the true positive and false positive rates at different decision thresholds), based on the posterior probabilities calculated by each of the learners, does not significantly change due to the application of sampling. The results for threshold-dependent measures such as the geometric mean or *F*-measure may be different, as those utilize the (possibly) inappropriate threshold of 0.5 to determine the predicted class of each example.

4.2. Sampling technique results

In this section, we examine the effects of *noise* and *imbalance* on the performance of the eight sampling techniques: BSM, CBOS, OSS, ROS, RUS, SM, WE and None. In Section 4.2.1 we look at relative impact of imbalance vs. that of noise on sampling technique performance. In Section 4.2.2 we examine the overall effect of noise on the performance of sampling techniques. In Section 4.2.3 we take a closer look at the effect of noise and imbalance by examining the effect of increasing the level of noise at each level of imbalance.

4.2.1. Sampling technique selection and imbalance level

In this section, we examine the relative impact of imbalance vs. noise when applying the seven sampling techniques as well as performing no sampling. Tables 9–12 provide the results of our experiments and form the basis of our conclusions. These tables show the average AUCs achieved by each sampling technique on each dataset summarized as follows:

1. For each *classifier*, *sampling* technique, *noise* level and *imbalance* level combination, 10-fold cross validation is repeated 30 times. The AUC values of these 30 repetitions are averaged, resulting in 4224 average AUC values (32 *sampling*/parameter combinations $\times$ 11 *learners* $\times$ 4 *imbalance* levels $\times$ 3 *noise* levels).
2. For each of the 7 *sampling* techniques, consider only the parameter which yields the best results (highest AUC). The result is 1056 averaged AUC values (8 *sampling* techniques $\times$ 11 *classifiers* $\times$ 4 *imbalance* levels $\times$ 3 *noise* levels).
3. Average the best *sampling* parameter results across all 11 *classifiers*. The result is 96 average AUC values (8 *sampling* techniques $\times$ 4 *imbalance* levels $\times$ 3 *noise* levels). These values are provided in Tables 9–12, in the columns labeled $noise = x$, where $x \in \{O, 5P, 10P\}$.
4. Average the results across all *noise* levels. The result is 32 average AUC values (8 *sampling* techniques $\times$ 4 *imbalance* levels). These values are provided in Tables 9–12 in the columns labeled *Average*.

Tables 9–12 show the results of our experiments for each of the 12 derived datasets, and provide a relative ranking of the sampling techniques at each level of imbalance. Fig. 4 summarizes the data in Tables 9–12 (CBOS is omitted from Fig. 4 to improve readability). Using the relative ranking provided in Tables 9–12, we can easily compare the performances of the different data sampling techniques at different levels of imbalance. For example, Table 12 sampling techniques when data is most imbalanced ($\lambda = 12$) are RUS, WE, BSM and SM (ranked 1, 2, 3 and 4, respectively). RUS and WE consistently outperform the other techniques (ranked 1 and 2, respectively) for all levels of imbalance. BSM outperforms the remaining techniques at higher levels of imbalance ($\lambda = \{8, 12\}$) but is outperformed by OSS and None at the lowest level of imbalance ($\lambda = 4$). OSS, which is ranked 3rd at the lowest level of imbalance, is ranked 7 at all other levels of imbalance. SM does not perform as well as None for $\lambda = 4$, but does outperform None at higher levels of imbalance. CBOS consistently performs worse than the other techniques.

The results in Fig. 4 may, at first, seem somewhat unexpected. Previous works have shown that classification performance usually degrades with an increase in imbalance [39]. Fig. 4 shows a different result. Most sampling techniques actually show improved performance as λ is increased from 4 to 6, and from 6 to 8. It is not until λ is increased from 8 to 12 that the

Table 9
Average AUC values for each sampling technique at various noise levels for $\lambda = 4$.

Sampling	*Noise* = O	*Noise* = 5P	*Noise* = 10P	Average	Rank
BSM	0.9469	0.9240	0.8915	0.9208	5
CBOS	0.8932	0.8276	0.7674	0.8294	8
None	0.9429	0.9241	0.8987	0.9219	4
OSS	0.9421	0.9278	0.9014	0.9238	3
ROS	0.9422	0.9072	0.8764	0.9086	7
RUS	0.9476	0.9320	0.9076	0.9291	1
SM	0.9476	0.9191	0.8888	0.9185	6
WE	0.9490	0.9276	0.9024	0.9263	2
Average	0.9389	0.9112	0.8793	–	–

Table 10
Average AUC values for each sampling technique at various noise levels for $\lambda = 6$.

Sampling	*Noise* = O	*Noise* = 5P	*Noise* = 10P	Average	Rank
BSM	0.9523	0.9229	0.8807	0.9186	4
CBOS	0.9153	0.8411	0.7872	0.8479	8
None	0.9392	0.9148	0.8771	0.9104	6
OSS	0.9309	0.9120	0.8810	0.9079	7
ROS	0.9516	0.9154	0.8762	0.9144	5
RUS	0.9598	0.9344	0.9053	0.9331	1
SM	0.9549	0.9205	0.8817	0.9190	3
WE	0.9582	0.9339	0.8934	0.9285	2
Average	0.9453	0.9119	0.8728	–	–

Table 11
Average AUC values for each sampling technique at various noise levels for $\lambda = 8$.

Sampling	*Noise* = O	*Noise* = 5P	*Noise* = 10P	Average	Rank
BSM	0.9637	0.9378	0.9127	0.9380	3
CBOS	0.9145	0.8804	0.8476	0.8808	8
None	0.9421	0.9303	0.9051	0.9258	6
OSS	0.9371	0.9258	0.9079	0.9236	7
ROS	0.9513	0.9359	0.9123	0.9331	5
RUS	0.9632	0.9537	0.9360	0.9510	1
SM	0.9586	0.9346	0.9134	0.9355	4
WE	0.9597	0.9500	0.9308	0.9468	2
Average	0.9488	0.9311	0.9082	–	–

Table 12
Average AUC values for each sampling technique at various noise levels for $\lambda = 12$.

Sampling	*Noise* = O	*Noise* = 5P	*Noise* = 10P	Average	Rank
BSM	0.9340	0.9250	0.9089	0.9226	3
CBOS	0.8904	0.8594	0.8334	0.8611	8
None	0.9229	0.9164	0.9022	0.9138	5
OSS	0.9142	0.9134	0.8991	0.9089	7
ROS	0.9248	0.9159	0.8995	0.9134	6
RUS	0.9598	0.9436	0.9321	0.9452	1
SM	0.9352	0.9157	0.8993	0.9167	4
WE	0.9367	0.9277	0.9242	0.9295	2
Average	0.9273	0.9147	0.8998	–	–

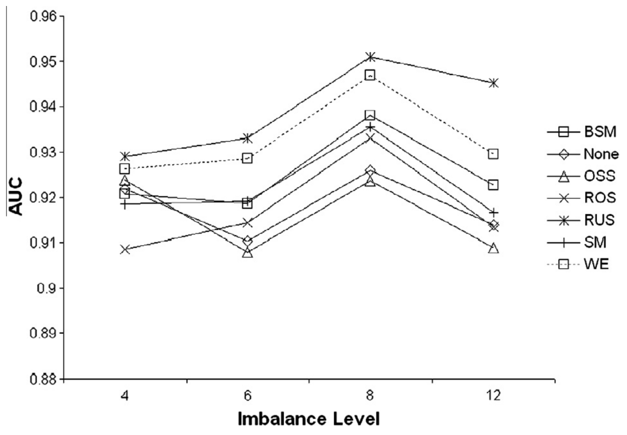

Fig. 4. Sampling techniques performance for different levels of *imbalance*.

expected decrease in performance is observed. This can be explained based on the information provided in Sections 3.5 and 3.6 and Fig. 3. Specifically, as we increase λ we are also decreasing the amount of noise in the data. While the increased imbalance would normally lead to a decrease in AUC, the decrease in noise offsets this effect and leads to an increase in AUC. It is not until $\lambda = 12$ that the increase in imbalance is enough to overcome the decrease in noise and result in a decrease in AUC. This result highlights the importance of ensuring the quality of data before applying sampling techniques. The relationship between noise and imbalance and their effects on sampling techniques is examined closer in Sections 4.2.3.

4.2.2. Sampling technique and noise level

In this section we will examine how different sampling techniques perform in the presence of different levels of noise. Tables 13–15 show the experimental results that will form the basis of this comparison. These results are collected in the same manner as those in Section 4.2.1 but are grouped differently, and the averages are calculated across all imbalance levels, rather than all noise levels. Specifically, Steps 1 and 2 are identical to Section 4.2.1. Steps 3 and 4 are changed to:

3. Average the best *sampling* parameter results across all 11 *classifiers*. The result is 96 average AUC values (8 *sampling* techniques $\times$ 4 *imbalance* levels $\times$ 3 *noise* levels). These values are provided in Tables 13–15 in the columns labeled $\lambda = x$, where $x \in \{4, 6, 8, 12\}$.
4. Average the results across all *imbalance* levels. The result is 24 average AUC values (8 *sampling* techniques $\times$ 3 *noise* levels). It is this interaction between *sampling* and *noise* that we will examine in this section. These values are provided in Tables 13–15 in the columns labeled *Average*.

Table 13
Average AUC values for *noise* = O.

Sampling	$\lambda = 4$	$\lambda = 6$	$\lambda = 8$	$\lambda = 12$	Average	Rank
BSM	0.9469	0.9523	0.9637	0.934	0.9492	3
CBOS	0.8932	0.9153	0.9145	0.8904	0.9033	8
None	0.9429	0.9392	0.9421	0.9229	0.9368	6
OSS	0.9421	0.9309	0.9371	0.9142	0.931	7
ROS	0.9422	0.9516	0.9513	0.9248	0.9425	5
RUS	0.9476	0.9598	0.9632	0.9598	0.9576	1
SM	0.9476	0.9549	0.9586	0.9352	0.9491	4
WE	0.949	0.9582	0.9597	0.9367	0.9509	2
Average	0.9389	0.9453	0.9488	0.9273	–	–

Table 14
Average AUC values for *noise* = 5P.

Sampling	$\lambda = 4$	$\lambda = 6$	$\lambda = 8$	$\lambda = 12$	Average	Rank
BSM	0.924	0.9229	0.9378	0.925	0.9274	3
CBOS	0.8276	0.8411	0.8804	0.8594	0.8522	8
None	0.9241	0.9148	0.9303	0.9164	0.9214	5
OSS	0.9278	0.912	0.9258	0.9134	0.9198	6
ROS	0.9072	0.9154	0.9359	0.9159	0.9186	7
RUS	0.932	0.9344	0.9537	0.9436	0.9409	1
SM	0.9191	0.9205	0.9346	0.9157	0.9225	4
WE	0.9276	0.9339	0.95	0.9277	0.9348	2
Average	0.9112	0.9119	0.9311	0.9147	–	–

Table 15
Average AUC values for *noise* = 10P.

Sampling	$\lambda = 4$	$\lambda = 6$	$\lambda = 8$	$\lambda = 12$	Average	Rank
BSM	0.8915	0.8807	0.9127	0.9089	0.8985	3
CBOS	0.7674	0.7872	0.8476	0.8334	0.8089	8
None	0.8987	0.8771	0.9051	0.9022	0.8958	5
OSS	0.9014	0.881	0.9079	0.8991	0.8973	4
ROS	0.8764	0.8762	0.9123	0.8995	0.8911	7
RUS	0.9076	0.9053	0.936	0.9321	0.9203	1
SM	0.8888	0.8817	0.9134	0.8993	0.8958	6
WE	0.9024	0.8934	0.9308	0.9242	0.9127	2
Average	0.8793	0.8728	0.9082	0.8998	–	–

Tables 13–15 show results grouped by *noise* level. The results in Table 13 are for *noise* = O. Tables 14 and 15 provide results for *noise* = 5P and 10P, respectively. The first column of each of these tables lists the *sampling* technique. The second through fifth columns list the results for the indicated noise level at each imbalance level $\lambda \in \{4,6,8,12\}$. The sixth column shows the average AUC for each sampling technique across all imbalance levels. The last column provides the relative rank of the sampling technique with 1 indicating the best performance (highest AUC) and 8 being the worst.

For *noise* = O, RUS performs the best with an average AUC of 0.9576 followed by WE (0.9509), BSM (0.9492) and SM (0.9491). ROS is ranked 5 with an average AUC of 0.9425. CBOS and OSS perform worst, with average AUCs of 0.9033 and 0.931, respectively. These two techniques yield worse results than performing no sampling at all (0.9368). These rankings are based on the average performance across all levels of imbalance. Note, for example, that while RUS achieves a better rank than WE when all levels of imbalance are considered, WE actually outperforms RUS at the lower levels of imbalance. More details regarding the combined effect of noise and imbalance are presented in Section 4.2.3.

Similar results are achieved for *noise* = 5P. Again, RUS, WE, BSM and SM outperform the other sampling techniques and CBOS yields the lowest average AUC. However, at this level of noise, ROS no longer performs better than no sampling. In fact, ROS drops in rank from 5 (*noise* = O) to 7 (*noise* = 5P), suggesting that it is more severely impacted by the increase noise than both no sampling and OSS. As with *noise* = O, OSS is also outperformed by no sampling.

Finally, for *noise* = 10P, RUS, WE and BSM continue to outperform the other sampling techniques. However, SM drops from 4 (*noise* $\in \{O, 5P\}$) to 6 at this noise level. Unlike the two lower noise levels, SM is now outperformed by both OSS and no sampling. In fact, at this noise level, OSS now provides better results than no sampling, suggesting it may be more robust in the presence of noise. This makes sense, since OSS is designed to avoid including noise in the sampled data. ROS once again outperforms only CBOS, which consistently performs worst among all sampling techniques used in this work.

Fig. 5 shows the average AUC for each sampling technique across all levels of imbalance at each noise level. CBOS is not shown in this figure since it is significantly worse than the other techniques at all noise levels. Again, RUS and WE perform best at all levels of noise. In addition, the lines for RUS and WE are relatively parallel, suggesting that these two techniques have a similar sensitivity to noise. BSM is ranked 3 at all levels of noise, but its performance degrades more severely than RUS and WE as the noise level increases. SM and ROS both perform better than OSS and no sampling on the original data, but when noise is increased, the performance of SM and ROS both degrade more quickly than OSS and no sampling, further suggesting that SM and ROS are more sensitive to noise than the other techniques.

4.2.3. Combined impact of noise and imbalance on sampling techniques

Figs. 6–9 take a closer look at the effects of noise on the various sampling techniques at different levels of imbalance. These figures show the change in performance of each sampling technique at each level of imbalance as noise is increased. In addition to judging overall performance, these figures allow us to determine which sampling techniques are more sensitive to noise at a given level of imbalance. A sampling technique that is sensitive to noise will result in a line with a steeper slope than a technique that is robust in the presence of noise. By presenting results for each level of imbalance we can observe the combined effect of imbalance and noise on the sampling techniques. CBOS is omitted from these figures to enhance readability. In all cases, CBOS performed significantly worse than the other techniques.

Fig. 6 shows the performance of each sampling technique averaged across all learners at the imbalance level $\lambda = 4$. At *noise* = O (the original data with no injected noise) sampling techniques BSM, RUS, SM and WE perform similarly, with AUCs between 0.9469 (BSM) and 0.9490 (WE). None, OSS and ROS do not perform as well as the others with AUCs between 0.9421 (OSS) and 0.9429 (None). As the level of noise in the dataset is increased, the difference in performance between the sampling techniques also increases. For *noise* = 5P the AUCs for the different techniques range from 0.9072 (ROS) to 0.932 (RUS). CBOS, which is not included in the figure yielded an AUC of 0.8726. For *noise* = 10P the range in values is even greater, ranging from 0.8764 (ROS) to 0.9076 (RUS). At this level of imbalance, RUS performs best as noise is increased. Although WE outperforms RUS at *noise* = O, RUS outperforms WE and all other techniques for *noise* $\in \{5P, 10P\}$. OSS does not perform well at *noise* = O, but demonstrates robustness as noise is increased, performing almost as well as WE at the higher noise levels. SM and ROS are both affected by noise resulting in the most severe decrease in performance as noise is increased at this level of

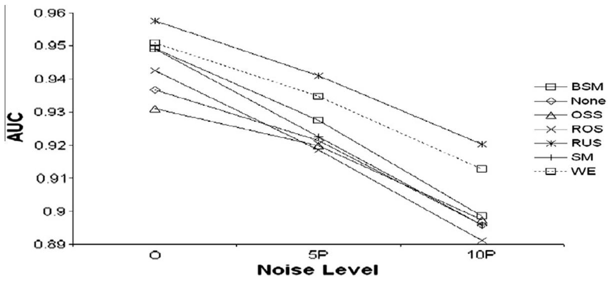

Fig. 5. Sampling technique performance (AUC) for different levels of noise.

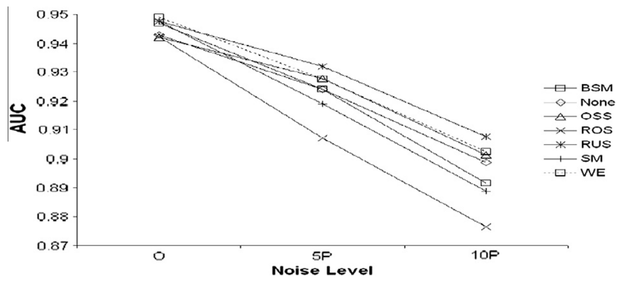

Fig. 6. Sampling technique performance (AUC) for imbalance level $\lambda = 4$.

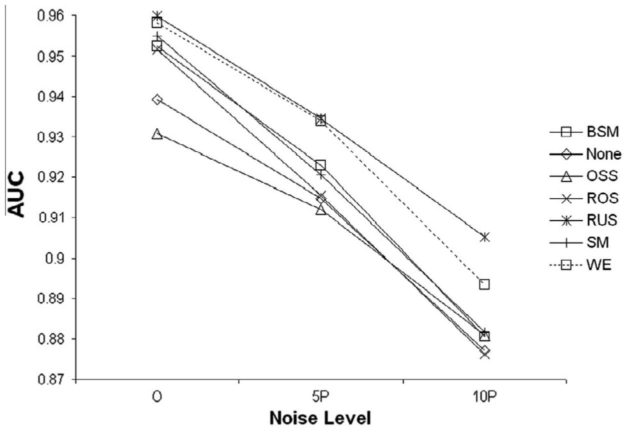

Fig. 7. Sampling technique performance (AUC) for imbalance level $\lambda = 6$.

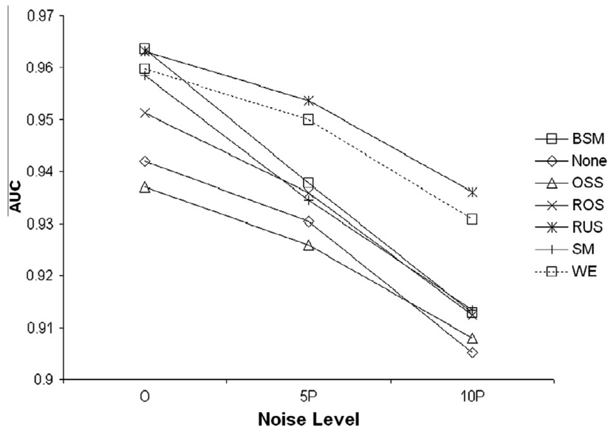

Fig. 8. Sampling technique performance (AUC) for imbalance level $\lambda = 8$.

imbalance. Note that CBOS, which is not included in Fig. 6, yielded a more severe drop in performance than both SM and ROS as noise was increased.

Fig. 7 shows the change in performance as noise is increased at imbalance level $\lambda = 6$. At this level of imbalance, RUS again performs best at all noise levels and proves itself to be relatively robust in the presence of noise. The only technique whose performance degrades less than RUS as noise is increased is OSS. OSS only experiences a 5.35% decrease in performance as noise is increased from O to 10P. This is not to say OSS performs best at this level of imbalance. At *noise* = O, OSS performs the worst of all techniques except CBOS which is not included in the figure. At *noise* = 10P, OSS is still outperformed by RUS, WE and SM. Once again, WE ranks 2 in overall performance behind only RUS. Also ROS and SM continue to be most affected as noise is increased, resulting in the sharpest decline in performance as indicated by the large slope of their respective lines in Fig. 7. Only CBOS, which is not included in the figure, results in a larger decrease in performance as noise is increased.

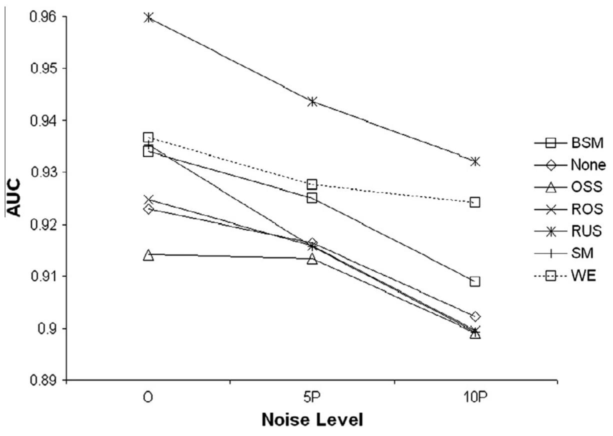

Fig. 9. Sampling technique performance (AUC) for imbalance level $\lambda = 12$.

Fig. 8 shows the change in performance as noise is increased at imbalance level $\lambda = 8$. At this level of imbalance, the lines for WE and RUS are nearly parallel suggesting that these two sampling techniques are similarly affected by the increase in noise. This is different than lower levels of imbalance where RUS appeared to be less affected by noise than WE. Also unlike the lower imbalance levels, BSM is among the most affected by the increase in noise, second only to CBOS in performance degradation. SM continues to be severely impacted by the increase in noise. Once again, OSS is relatively unaffected by noise, but this does not make up for its relatively weak performance overall. At this imbalance level OSS only outperforms CBOS (not included in figure) at *noise* $\in \{O, 5P\}$. At *noise* = 10P, OSS also outperforms no sampling.

Finally, Fig. 9 shows the change in performance as noise is increased at imbalance level $\lambda = 12$. At this most severe level of imbalance, all techniques except RUS suffer a large decrease in performance at *noise* = O. Unlike the lower imbalance levels, while RUS does perform best overall, its decrease in performance as the level of noise is increased is among the worst of all sampling techniques. This decrease in performance is not enough to drop its overall performance below that of any other sampling technique at any level of noise. However, it does suggest that at very high levels of imbalance RUS is more affected by noise than many other techniques. On the other hand, WE is relatively unaffected by the increase in noise at this level of imbalance. WE is designed to prevent the use of noisy examples in the training data, and it appears to be successful, especially at the highest level of imbalance. SM and ROS continue to be relatively sensitive to noise at this level of imbalance.

4.3. Learning algorithm results with sampling

In this section, we examine the effects of *imbalance* and *noise* on the 11 classification algorithms: 2NN, 5NN, C4.5(D), C4.5(N), LR, MLP, NB, RBF, RF, RIPPER and SVM, after sampling techniques have been applied. In section 4.3.1 we examine the impact of the level of imbalance on each algorithm. In section 4.3.2 we study the impact of increasing noise on the performance of each classification algorithm. In section 4.3.3 we take a closer look at the combined impact of noise and imbalance on classifier performance. In each case, sampling is performed (on only the training dataset) prior to building the models that will be used to evaluate classification algorithm performance.

4.3.1. Learning algorithms and imbalance level

In this section we examine the effect of imbalance on the 11 learning algorithms. Fig. 10 shows the change in performance of all learners as imbalance is increased. The AUCs values in this figure are calculated as follows:

1. For each *classifier*, *sampling* technique, *noise* level and *imbalance* level combination, 10-fold cross validation is repeated 30 times. The 30 repetitions are averaged, resulting in 4224 average AUC values (32 *sampling*/parameter combinations $\times$ 11 *classifiers* $\times$ 4 *imbalance* levels $\times$ 3 *noise* levels).
2. For each of the 7 *sampling* types, consider only the parameter which yields the best results (highest AUC). The result is 1056 averaged AUC values (8 *sampling* techniques $\times$ 11 *classifiers* $\times$ 4 *imbalance* levels $\times$ 3 *noise* levels).
3. Average the best *sampling* parameter results across all 8 *sampling* techniques. The result is 132 average AUC values (11 *classifiers* $\times$ 4 *imbalance* levels $\times$ 3 *noise* levels).
4. Average the results across all *noise* levels. The result is 44 average AUC values (11 *classifiers* $\times$ 4 *imbalance* levels).

Fig. 10 shows a similar trend as Fig. 4. Once again, as the level of imbalance increases, the performance of many learning algorithms also increases until the highest level of imbalance, $\lambda = 12$ is reached. At this highest level of imbalance, the performance of most learners decreases. As discussed in Section 4.2.1 this is a result of the relationship between noise levels and

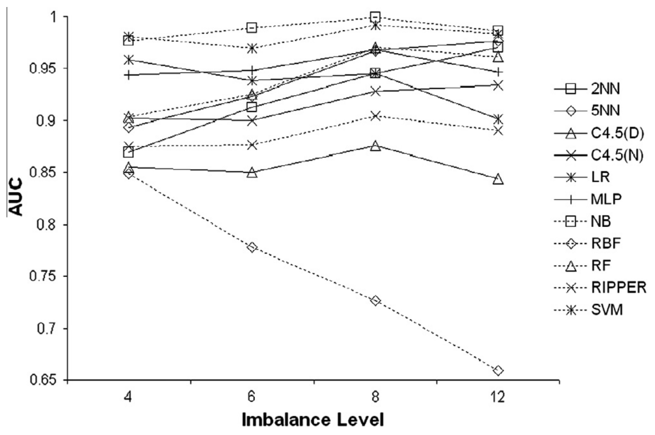

Fig. 10. Classification algorithm performance for different levels of *imbalance*.

imbalance levels in our data. As we increase λ, the level of imbalance increases but the amount of noise in the data decreases. This decrease in noise is responsible for an increase in performance where one might expect performance to degrade.

Some learners continue to improve even at the highest level of imbalance. 2NN and 5NN both consistently improve as imbalance is increased, suggesting that these "nearest neighbor"-based algorithms are relatively unaffected by imbalance and that they are helped by the decrease in noise that occurs as a side effect of increasing imbalance. C4.5(N) also performs better at higher levels of imbalance than at lower levels, demonstrating that it is also more sensitive to noise than imbalance. RBF, on the other hand, is shown to be very sensitive to imbalance in data. Even though the noise level in the data decreases as imbalance increases, the performance of RBF consistently decreases with increased imbalance. This change in AUC is much greater than the change in AUC for other learning algorithms.

4.3.2. Learning algorithms and noise levels

In this section, we examine the effect of noise level on each of the learning algorithms. Tables 16–18 provide the experimental results that will be used to perform this analysis. The values in these tables are calculated in the same manner as those in Section 4.3.1, but they are grouped by noise levels and averaged across all levels of imbalance. Specifically, Steps 3 and 4 change to:

3. Average the best *sampling* parameter results across all 8 *sampling* techniques. The result is 132 average AUC values (11 *classification* algorithms $\times$ 4 *imbalance* levels $\times$ 3 *noise* levels). These values are provided in Tables 16–18, in the columns labeled $\lambda = x$, where $x \in \{4,6,8,12\}$.
4. Average the results across all *imbalance* levels. The result is 33 average AUC values (11 *classification* algorithms $\times$ 3 *noise* levels). These values are provided in Tables 9–12 in the columns labeled *Average*.

Fig. 11 plots the average AUC values from Tables 16–18. RBF is consistently outperformed by the other learning algorithms and is not included in this figure. NB performs better than all other classification algorithms for all noise levels and is relatively unaffected by the increase in noise as indicated by the nearly horizontal line in Fig. 11. SVM also performs very well, achieving a higher AUC at all noise levels than all learning algorithms with the exception of NB. While the line

Table 16
Average AUC values for *noise* = O.

Learner	$\lambda = 4$	$\lambda = 6$	$\lambda = 8$	$\lambda = 12$	Average	Rank
2NN	0.9158	0.9554	0.9659	0.9791	0.9540	7
5NN	0.9394	0.9654	0.9839	0.9831	0.9680	3
C4.5(D)	0.8860	0.8920	0.8979	0.8522	0.8820	10
C4.5(N)	0.9487	0.9519	0.9667	0.9688	0.9590	6
LR	0.9566	0.9524	0.9481	0.9035	0.9401	8
MLP	0.9621	0.9688	0.9807	0.9535	0.9663	5
NB	0.9806	0.9908	0.9993	0.9827	0.9883	1
RBF	0.9063	0.8711	0.7979	0.7321	0.8268	11
RF	0.9509	0.9665	0.9845	0.9653	0.9668	4
RIPPER	0.9001	0.9015	0.9184	0.8944	0.9036	9
SVM	0.9819	0.9823	0.9929	0.9852	0.9856	2
Average	0.9389	0.9453	0.9488	0.9273	–	–

Table 17
Average AUC values for *noise* = 5P.

Learner	$\lambda = 4$	$\lambda = 6$	$\lambda = 8$	$\lambda = 12$	Average	Rank
2NN	0.8741	0.9168	0.9451	0.9697	0.9264	7
5NN	0.8968	0.9295	0.9704	0.9759	0.9432	6
C4.5(D)	0.8527	0.8560	0.8856	0.8458	0.8600	10
C4.5(N)	0.9072	0.9119	0.9365	0.9335	0.9223	8
LR	0.9664	0.9519	0.9560	0.9107	0.9463	4
MLP	0.9491	0.9555	0.9710	0.9474	0.9557	3
NB	0.9775	0.9895	0.9990	0.9898	0.9890	1
RBF	0.8379	0.7411	0.6953	0.6464	0.7302	11
RF	0.9066	0.9292	0.9739	0.9643	0.9435	5
RIPPER	0.8707	0.8776	0.9139	0.8925	0.8887	9
SVM	0.9839	0.9714	0.9950	0.9851	0.9839	2
Average	0.9112	0.9119	0.9311	0.9147	–	–

Table 18
Average AUC values for *noise* = 10P.

Learner	$\lambda = 4$	$\lambda = 6$	$\lambda = 8$	$\lambda = 12$	Average	Rank
2NN	0.8188	0.8667	0.9260	0.9627	0.8936	7
5NN	0.8438	0.8757	0.9468	0.9698	0.9090	6
C4.5(D)	0.8262	0.8020	0.8436	0.8335	0.8263	10
C4.5(N)	0.8507	0.8352	0.8811	0.8998	0.8667	9
LR	0.9541	0.9107	0.9326	0.8889	0.9216	4
MLP	0.9213	0.9194	0.9516	0.9385	0.9327	3
NB	0.9726	0.9878	0.9994	0.9871	0.9867	1
RBF	0.8020	0.7202	0.6871	0.5982	0.7019	11
RF	0.8521	0.8804	0.9519	0.9544	0.9097	5
RIPPER	0.8539	0.8493	0.8812	0.8853	0.8674	8
SVM	0.9764	0.9537	0.9893	0.9801	0.9749	2
Average	0.8793	0.8728	0.9082	0.8998	–	–

representing SVM is not completely horizontal, it is flatter than the lines for most other techniques suggesting that SVM is also relatively robust in the presence of noise, even when data is imbalanced.

LR is unique in that it achieves a higher AUC at *noise* = 5P than *noise* = O. This increase in performance is likely due to the decrease in imbalance as noise is increased. Fig. 10 supports this, showing that LR is one of the few learners that does not increase in performance as imbalance is increased (for $\lambda \in 4, 6, 8$). LR seems to be more sensitive to imbalance than noise in data. The results in Section 4.3.3 support this and future research will be performed to test this hypothesis.

5NN performs well at *noise* = O, but when the noise level is increased its rank among the 11 classification algorithms drops from rank 3 to 6. MLP, RF and LR all outperform 5NN at noise levels 5P and 10P. Other than RBF which is not included in Fig. 11, C4.5(D) performs worst at all noise levels. C4.5(N) is ranked 6 in performance at *noise* = O, but its performance declines more quickly than any other learning algorithm as noise is increased. At *noise* = 10P, C4.5(N) ranks 9, performing

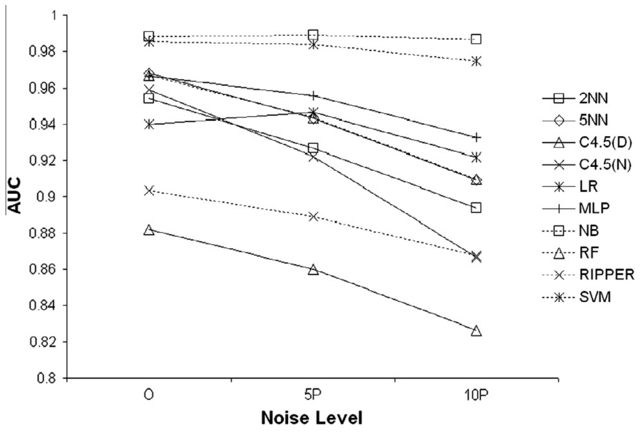

Fig. 11. Classification algorithm performance at different levels of noise.

better than only RBF and C4.5(D). The remaining learning algorithms show a degradation in performance with increase in noise level that falls between these extremes.

4.3.3. Impact of noise and imbalance on post-sampling classifier performance

Figs. 12–15 take a closer look at the effects of noise on the various classification algorithms at different levels of imbalance. These figures show the performance of each classifier at each level of imbalance as noise is increased. In addition to

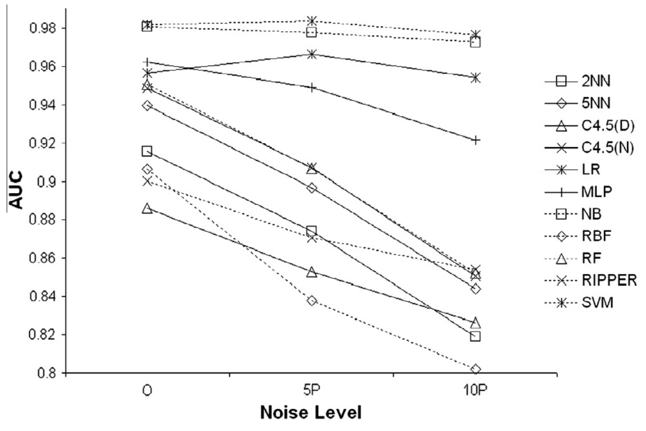

Fig. 12. Classification algorithm performance at different levels of noise for $\lambda = 4$.

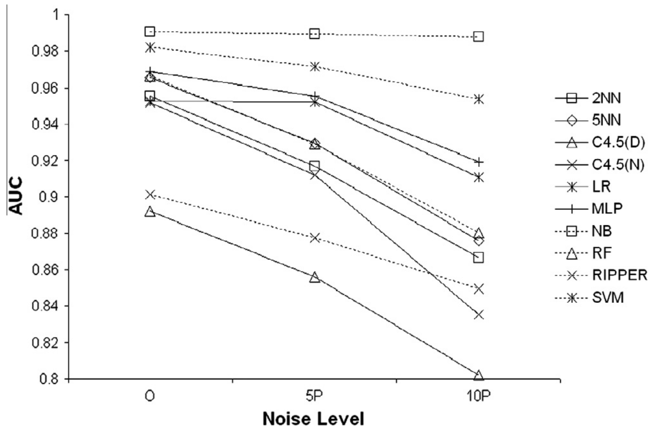

Fig. 13. Classification algorithm performance at different levels of noise for $\lambda = 6$.

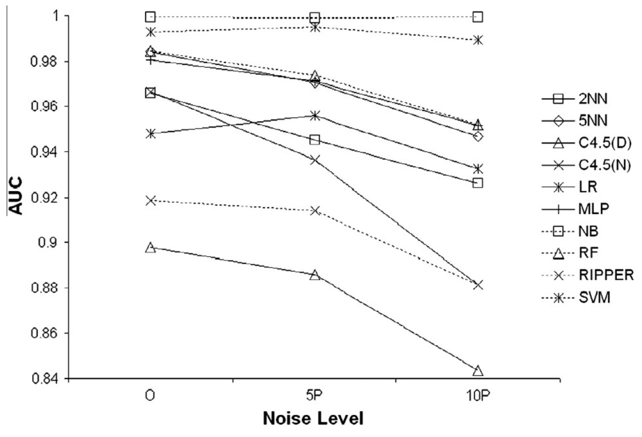

Fig. 14. Classification algorithm performance at different levels of noise for $\lambda = 8$.

judging overall performance of the classifiers, these figures allow us to determine which are more sensitive to noise at a given level of imbalance. A relatively steeper slope in these figures suggests that the corresponding classification algorithm is more sensitive to noise at that level of imbalance than those classification algorithms exhibiting a more horizontal line. By presenting results for each level of imbalance, we can observe the combined effect of imbalance and noise on the classification algorithms.

Fig. 12 shows the change in performance for all classification algorithms as noise is increased for imbalance level $\lambda = 4$. NB and SVM perform the best with the highest AUC values and relatively horizontal lines suggesting robustness as noise is increased. RIPPER also appears to be relatively unaffected by the increase in noise at this imbalance level, although its performance is lower overall than many other algorithms, especially at low levels of noise. LR is unique in that its performance actually improves slightly as noise is increased from O to 5P, and although its performance degrades as noise is increased from 5P to 10P, its AUC is higher at noise = 10P than at noise = O. This supports our previous statement that LR is more sensitive to imbalance than noise. Recall that as noise is increased for any value of λ the actual severity of imbalance within the data is decreased slightly as discussed in section 3.6. In addition to RBF's apparent high sensitivity to imbalance, it also appears to be highly sensitive to the increase in noise at this level of imbalance as indicated by the steep slope of the line representing its performance. C4.5(D) appears to be less sensitive to the increase in noise than most, but its overall performance is among the worst at this imbalance level. The two nearest neighbor techniques are similarly affected by noise as indicated by the nearly parallel lines representing these two classifiers. RF and C4.5N also show very similar results, both overall and with respect to sensitivity to noise.

Fig. 13 shows the change in performance for all classification algorithms as noise is increased for imbalance level $\lambda = 6$. RBF performed significantly worse than the other algorithms and is not included in this figure. Unlike $\lambda = 4$, at this level of imbalance NB outperforms SVM at all levels of noise. Also, while NB continues to appear relatively unaffected by the increase in noise, SVM begins to show sensitivity as noise is increased. Once again, LR is unique in that it performs better when *noise* = 5P than *noise* = O. C4.5(N) once again outperforms C4.5(D), but C4.5(N) demonstrates a steeper slope indicating a relatively greater sensitivity to noise than C4.5(D). 5NN and 2NN continue to show similar sensitivity to noise, with 5NN outperforming 2NN at this level of imbalance. MLP ranks third in performance at all levels of noise, but its sensitivity to noise appears to be greater than that of NB and SVM.

Fig. 14 shows the change in performance for all classification algorithms as noise is increased for imbalance level $\lambda = 8$. Once again, NB performs best at all levels of noise, followed by SVM. These two algorithms continue to show robustness as the level of noise is increased. Also, like previous levels of imbalance, LR stands out as the only technique to perform better when *noise* = 5P than when *noise* = 10P. RF, 5NN and MLP perform similarly at this level of imbalance. Once again, C4.5(N) outperforms C4.5(D), but also shows a greater sensitivity to noise. 2NN does not appear to be more sensitive to noise than 5NN, but its performance is, once again, worse than 5NN.

Fig. 15 shows the change in performance for all classification algorithms as noise is increased for imbalance level $\lambda = 12$. NB and SVM continue to outperform the other algorithms. Unlike previous levels of imbalance the performance of these algorithms is actually better when *noise* = 5P or 10P than when *noise* = O, suggesting that the decrease in imbalance as noise increases is playing a more significant role in the performance of these classifiers than the noise at this level of imbalance. LR continues to demonstrate this trend, but performs noticeably worse at this level of imbalance than at the lower levels of imbalance further supporting our claim that LR is relatively more sensitive to imbalance than most other algorithms. Most other algorithms demonstrate similar sensitivities to noise with the exception of C4.5(N) which appears to be much more sensitive to noise at this level of imbalance than the other algorithms.

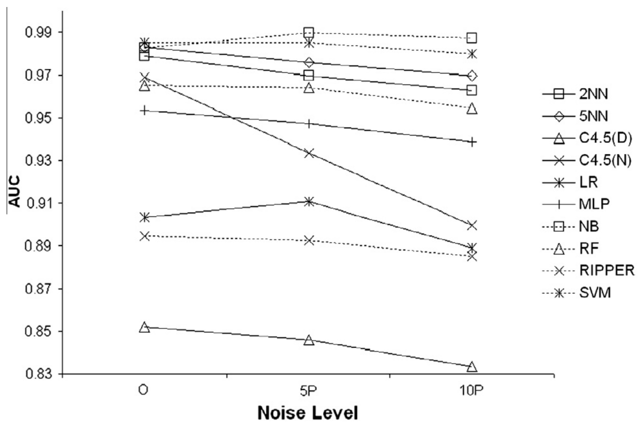

Fig. 15. Classification algorithm performance at different levels of noise for $\lambda = 12$.

4.4. Four-factor ANOVA model

A four-factor analysis of variance or ANOVA model [4] can be represented as:

$$AUC_{jklmn} = \mu + \alpha_j + \beta_k + \gamma_l + \eta_m + \epsilon_{jklmn}$$

where

- AUC_{jklmn} is the nth value of the response variable (AUC) for the jth level of α, kth level of β, lth level of γ and mth level of η,
- μ is the overall mean of the response variable,
- α_j (resp., β_k, γ_l, η_m) is the mean *err* of level j (resp., k, l, m) for attribute α (resp., β, γ, η),
- ϵ_{jklmn} is the random error.

Attributes α, β, γ and η are called *main* effects or factors. Crossed effects, which consider the interaction between one or more main effects, can also be included. An example of a crossed effect is $(\alpha\beta)_{jk}$. The main effects in our study are the learning algorithm (*learner*, α), sampling technique (*samp*, β), level of imbalance (*imb*, γ) and noise level (*noise*, η).

The ANOVA model can be used to test the hypothesis that the *AUC* for each level of the main factors *learner*, *samp*, *noise* and *imb* are equal against the alternative hypothesis that at least one is different. If the alternative hypothesis (i.e., that at least one *AUC* is different) is accepted, numerous procedures can be used to determine which of the *AUC* values are significantly different from the others. This involves a comparison of two *AUC* values, with the null hypothesis that they are equal. A Type I error occurs when the null hypothesis is incorrectly rejected. In this study, we use Tukey's Honestly Significant Difference (HSD) test [36], which controls the Type I experiment-wise error rate [4].

Table 19 shows the analysis of the main effects *learner*, *noise*, *samp* and *imb* and the cross effects *learner* × *imb*, *learner* × *noise*, *imb* × *samp*, *noise* × *samp* and *imb* × *noise*. The first column lists the experimental factors, while the remaining columns provide the number of degrees of freedom, the *F*-statistic, and the *p*-value (i.e., significance level of each experimental factor). Each of these main effects and cross effects are significant at the $\alpha = 5\%$ significance level, supporting our findings that learner selection, sampling technique selection, and the levels of both imbalance and noise can have an impact on classification performance as measured using AUC. Based on the *F*-statistic, noise has a relatively greater effect on performance than imbalance when learning from our derived software quality datasets. In addition, this ANOVA analysis supports our conclusion that the interaction between the level of imbalance and the level of noise within a dataset is a significant factor, and that studying these two main effects only in isolation may not be sufficient.

Table 20 shows the analysis of the main effect *learner*. This analysis supports our conclusions drawn in Sections 4.3 and 4.1 that not all classification algorithms perform equally in the presence of class noise and imbalance. NB and SVM perform similarly, both belonging to group A. While their mean AUCs were not significantly different from each other, they were statistically better than the remaining classifiers. MLP, RF and 5NN all belong to group B, meaning that their mean AUCs were not significantly different from each other. Since MLP is a member of only group B, we can say that it performs better than all LR and all learners that are lower than LR in Table 20. While RF and 5NN do not perform significantly worse than MLP, since they are members of both groups B and C, we cannot say they these two learners performed better than LR, which is also a member of group C. Also, our conclusion that RBF does not perform well in the presence of imbalance and noise is supported by this analysis.

Table 21 shows the analysis of the main effect *imb*. The column *imb* provides the λ value used to create the given level of imbalance. Although *imb* is shown to be a significant effect by Table 19, this table shows that only $\lambda = 8$ results in an AUC that is significantly different from the remaining values of λ. This supports our conclusion that *noise* has a greater impact than *imb* in our datasets. Also note that the order of the λ values in Table 21 is almost the opposite of what might be expected. This is due to the fact that as we increase the level of imbalance (λ) the amount of noise in the data decreases, as explained in Section 3.6.

Table 22 shows the analysis of the main effect *noise*. Each level of noise results in a mean AUC that is significantly higher than the next most severe level of noise. That is, our cleanest dataset, C^o results in a higher mean AUC than C^{5p} which results

Table 19
Analysis of effects and some cross effects.

Source	DoF	*F*-statistic	*p*-Value
Learner (α)	10	541.35	<0.0001
Samp (β)	7	115.48	<0.0001
Imb (γ)	3	29.41	<0.0001
Noise (η)	2	286.9	<0.0001
Learner × *imb* ($\alpha\gamma$)	30	35.81	<0.0001
Learner × *noise* ($\alpha\eta$)	20	15.76	<0.0001
Imb × *samp* ($\gamma\beta$)	21	2.32	0.0007
Noise × *samp* ($\eta\beta$)	14	5.56	<0.0001
imb × *noise* ($\gamma\eta$)	6	11.4	<0.0001

Table 20
Main effect: *learner*.

Learner	Mean	*n*	HSD
NB	0.98801	96	A
SVM	0.981434	96	A
MLP	0.951564	96	B
RF	0.939985	96	CB
5NN	0.93981	96	CB
LR	0.935994	96	CD
2NN	0.924669	96	ED
C4.5(N)	0.916005	96	E
RIPPER	0.886565	96	F
C4.5(D)	0.856139	96	G
RBF	0.752962	96	H

Table 21
Main effect: *imb*.

Imb	Mean	*n*	HSD
8	0.929329	264	A
12	0.913899	264	B
6	0.909938	264	B
4	0.909793	264	B

in a higher AUC than C^{10p}. Combined with the results in Table 19 this clearly demonstrates that the level of noise has a significant impact on classification performance.

Finally, Table 23 shows the analysis of the main effect *samp*. As concluded in Section 4.2, RUS significantly outperforms all other sampling techniques with the exception of WE. These two sampling techniques both belong to group A, and therefore we cannot say that one performs significantly better than the other. This supports our conclusion that RUS consistently performs very well, and is matched only by WE, and only at certain levels of noise and imbalance. Although WE belongs to group A, it also belongs to group B along with BSM and SM. Therefore we cannot conclude that WE significantly outperforms these two techniques on average, across all datasets. Sampling technique None (not performing any data sampling) performs as well as BSM, SM, ROS and OSS, leading us to conclude that these sampling techniques do not result in a significant improvement in classification performance averages across all 12 of our datasets. Finally, CBOS performs significantly worse than all other sampling techniques when applied to our datasets.

5. Threats to validity

Experimental research commonly includes a discussion of two different types of threats to validity [42]. 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 [41] was used for the construction of all classifiers, and we have included all of the parameter settings to ensure repeatability. The ANOVA

Table 22
Main effect: *noise*.

Noise	Mean	*n*	HSD
C^o	0.940026	352	A
C^{5p}	0.917183	352	B
C^{10p}	0.890009	352	C

Table 23
Main effect: *samp*.

Samp	Mean	*n*	HSD
RUS	0.9396	132	A
WE	0.932799	132	BA
BSM	0.924853	132	BC
SM	0.922445	132	BC
NONE	0.917989	132	C
ROS	0.91739	132	C
OSS	0.916051	132	C
CBOS	0.85479	132	D

model presented in this work was built using the SAS GLM procedure [36], and the assumptions for constructing valid ANOVA models were validated. All output was validated for accuracy by members of our research group, giving us confidence in the legitimacy of the results. Seven different sampling techniques were included in this 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. We believe that the results presented in this work are valid for other datasets. Our experiments were conducted under the careful supervision of a software engineering expert, which is a very important component of empirical work in the data quality domain. Randomly injecting noise, without regard for the nature of the application domain from which the dataset was derived, can be very problematic and potentially lead to invalid results. Further, an important dimension of software quality classification problems is the conversion of a continuous attribute such as *nfaults* to a binary classification attribute. In many situations, *nfaults* can be very noisy, and we study in this work the impact of noisy *nfaults* on binary classification by considering different thresholds. The performance measure considered in this work is the AUC, which is widely used in many application domains. As mentioned previously, different results may be obtained when different performance metrics are utilized, especially if those metrics are threshold-dependent such as the geometric mean or *F*-measure. As with any empirical work, additional experimentation is required to confirm our conclusions.

6. Conclusions

We have presented a comprehensive study on the effects of class imbalance and class noise on different classification algorithms and data sampling techniques when used to predict software quality. Using 12 datasets derived from real world software quality data with different levels of class noise and imbalance, we compared the performance of seven sampling techniques (plus no sampling), described in Section 3.8, and 11 classification algorithms, described in Section 3.7, on data containing both noise and imbalance. Based on our empirical evaluation we draw the following conclusions:

1. What is the relative impact of noise vs. imbalance? Which has a more severe impact on the performance of the different classification algorithms and sampling techniques.
   (a) Although RBF proved to be most sensitive to imbalance, most learners and sampling techniques actually improved in performance as imbalance was increased. Again, this is due to the fact that the level of noise in our data dropped as imbalance was increased. It is the reduction in the amount of noise that causes this improvement in performance, suggesting that many of the classification algorithms are more sensitive to noise than imbalance. However, as imbalance increases in severity, it plays a larger role in the performance of classifiers and sampling techniques.
   (b) The results for each sampling technique were similar to those of the classification algorithms. Due to the fact that the level of noise in the data decreases as we increase the level of imbalance, most sampling techniques performed better as λ was increased from 4 to 6 and from 6 to 8. This suggests that the reduction of noise had a more significant impact on sampling technique performance than the increase in imbalance. At the highest level of imbalance, however, the performance of all sampling techniques dropped.
2. How do different classification algorithms react to the application of different sampling techniques? Are some classifiers more significantly improved by the use of sampling techniques? Do certain sampling techniques work better with specific classifiers?
   (a) Some classification algorithms are greatly impacted by the application of sampling techniques, while others are relatively unaffected. Those classification algorithms that are affected by performing sampling are not always positively impacted. In some cases the application of sampling can actually hinder the performance of a classification algorithm, especially when the level of imbalance is not severe.
   (b) 2NN is an example of a classification algorithm whose performance can be worse when sampling is applied than when no sampling is used. For those datasets containing the lowest levels of imbalance, the best AUC achieved by any sampling technique with 2NN was lower than the AUC achieved when no sampling was used.
   (c) NB was relatively unaffected by applying sampling techniques. Among all datasets, the largest difference in AUC with and without sampling was only 0.14%. The average improvement using sampling was 0.04%.
   (d) RBF was significantly improved by applying sampling, especially at higher levels of imbalance. At lower levels of imbalance the largest improvement was only 3.6%. At higher levels of imbalance, improvement in AUC was as high as 38.54%. The average improvement over all datasets was 20.42%.
   (e) WE proved to be the best sampling technique (achieving the highest improvement in AUC the most often) when combined with five of the 11 classification algorithms. This is not to say WE is the best sampling technique overall, just that it was the best when combined with the most classification algorithms. While WE performed the best when combined with the most classifiers, those classifiers (2NN, 5NN, MLP, RF, SVM) did not show large improvements in AUC over no sampling. The improvement in AUC using these classification algorithms ranged from 0.10% to 1.27%.
   (f) RUS performed best when combined with four classification algorithms (C4.5(D), RBF, RIPPER and SVM). Although RUS was the best sampling techniques for fewer classifiers than WE, the classifiers that RUS improved the most had a much higher improvement in AUC. The improvement in AUC using these classification algorithms ranged from 1.27% to 20.42%.

(g) C4.5(D), RBF and RIPPER performed best when combined with RUS. 2NN, 5NN, MLP and RF performed best when combined with WE. RUS and WE were tied for best when combined with SVM. C4.5(N) and LR performed best with ROS and NB performed best when paired with SM. Note that NB was relatively unaffected by sampling, so while SM did yield the best results, the improvement was minimal.

3. What benefit do sampling techniques provide at different levels of class imbalance and noise? When the data is highly imbalanced or very noisy, do certain sampling techniques perform better than others?
   (a) RUS performed the best overall at all levels of noise and imbalance. WE also performed very well, outperforming all sampling techniques except RUS at all levels of imbalance and noise.
   (b) At lower levels of imbalance, WE seems to be hindered more by noise than RUS. This is somewhat surprising since WE is designed to create a noise-free training dataset. However, as the level of imbalance is increased, WE proves to be more robust in the presence of noise. At the highest level of imbalance the performance of WE degrades the least as the level of noise is increased compared to other sampling techniques.
   (c) OSS consistently proves itself to be relatively unaffected by an increase in noise level. At all levels of imbalance OSS's performance remains relatively stable as noise is injected into the data. Even though the performance does not degrade as much as most other techniques as noise is injected, the overall performance of OSS is not very good when compared to most other techniques. At low levels of noise, OSS performs better than only CBOS. Therefore, although its performance does not degrade much as noise is increased, it still does not perform as well as most other techniques.
   (d) CBOS consistently performs worse than any other sampling technique on our datasets.
4. How do classification algorithms perform at different levels of class imbalance and noise after sampling techniques have been applied to the data? When the data is highly imbalanced or very noisy, do classification algorithms perform better than others?
   (a) NB and SVM consistently perform best on all datasets for all levels of imbalance and noise. NB slightly outperforms SVM in our experiments.
   (b) Although our data demonstrates a characteristic where the level of noise in the data decreases as we increase the level of imbalance, we can draw one important conclusion about the classification algorithm RBF. Unlike most other classification algorithms, the performance of RBF is significantly hindered by an increase in imbalance. Most other algorithms experience only small changes in AUC when imbalance was increased, but RBF's AUC dropped from 0.8587 at imbalance level $\lambda = 4$ to 0.6589 at imbalance level $\lambda = 12$. This suggests that RBF is severely hindered by data imbalance.
   (c) C4.5(N) performs relatively well at the lowest noise level, but when noise is increased C4.5(N)'s performance degrades more quickly than others. At *noise* = O, C4.5(N) is ranked 6th out of 11 classifiers. At *noise* = 10P, however, it is ranked 9th.

Future work will extend this study by using additional datasets to verify our conclusions. In addition this work will be extended by applying noise handling techniques (to filter or correct noisy examples in a dataset) in conjunction with sampling techniques to enhance classification performance. In doing so, we will study the impact of class imbalance on noise handling techniques.

References

[1] J. Aczel, J. Daroczy, On Measures of Information and Their Characterizations, Academic Press, New York, NY, 1975.
[2] D.W. Aha, Lazy Learning, Kluwer Academic Publishers, Norwell, MA, USA, 1997.
[3] 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, vol. 3138, 3138/2004, 2004, pp. 806–814.
[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, Bagging predictors, Machine Learning 26 (2) (1996) 123–140.
[6] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[7] C.E. Brodley, M.A. Friedl, Identifying mislabeled training data, Journal of Artificial Intelligence Research 11 (1999) 131–167.
[8] 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.
[9] N.V. Chawla, N. Japkowicz, A. Kolcz, Editorial: special issue on learning from imbalanced data sets, SIGKDD Explorations 6 (1) (2004) 1–6.
[10] M.-C. Chen, L.-S. Chen, C.-C. Hsu, W.-R. Zeng, An information granulation based data mining approach for classifying imbalanced data, Information Sciences 178 (16) (2008) 3214–3227.
[11] W.W. Cohen, Fast effective rule induction, in: Proceedings of the 12th International Conference on Machine Learning, Morgan Kaufmann, 1995, pp. 115–123.
[12] P. Domingos, M. Pazzani, On the optimality of the simple bayesian classifier under zero-one loss, Machine Learning 29 (2–3) (1997) 103–130.
[13] 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.
[14] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the Seventeenth International Conference on Machine Learning, 2001, pp. 239–246.
[15] N.E. Fenton, S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, second ed., PWS Publishing Company, ITP, Boston, MA, 1997.
[16] J. Furnkranz, G. Widmer, Incremental reduced error pruning, in: International Conference on Machine Learning, 1994, pp. 70–77.
[17] D. Gamberger, N. Lavrač, C. Grošelj, Experiments with noise filtering in a medical domain, in: Proceedings of the 16th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1999, pp. 143–151.

[18] 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, vol. 3644, Springer-Verlag, 2005, pp. 878–887.
[19] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, 2001.
[20] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[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] T.M. Khoshgoftaar, E.B. Allen, Classification of fault-prone software modules: prior probabilities, costs and model evaluation, Empirical Software Engineering 3 (3) (1998) 275–298.
[24] T.M. Khoshgoftaar, E.B. Allen, Logistic regression modeling of software quality, International Journal of Reliability, Quality, and Safety Engineering 6 (4) (1999) 303–317.
[25] 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.
[26] T.M. Khoshgoftaar, J. Van Hulse, C. Seiffert, A hybrid approach to cleansing software measurement data, in: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006), Washington, DC, 2006, pp. 713–722.
[27] T.M. Khoshgoftaar, S. Zhong, V. Joshi, Enhancing software quality estimation using ensemble-classifier based noise filtering, Intelligent Data Analysis: An International Journal 6 (1) (2005) 3–27.
[28] A. Kolcz, A. Chowdhury, J. Alspector, Data duplication: an imbalance problem?, in: ICML'2003 Workshop on Learning from Imbalanced Datasets, 2003.
[29] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: one sided selection, in: Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 179–186.
[30] 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.
[31] T.M. Mitchell, Machine Learning, McGraw-Hill, New York, NY, 1997.
[32] J. Moody, C.J. Darken, Fast learning in networks of locally tuned processing units, Neural Computation 1 (2) (1989) 281–294.
[33] F. Provost, T. Fawcett, Robust classification for imprecise environments, Machine Learning 42 (2001) 203–231.
[34] J.R. Quinlan, C4.5: Programs For Machine Learning, Morgan Kaufmann, San Mateo, California, 1993.
[35] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, UK, 1996.
[36] SAS Institute, SAS/STAT User's Guide, SAS Institute Inc., 2004.
[37] B. Schlkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, Massachusetts, 1999.
[38] C.M. Teng, Correcting noisy data, in: Proceedings of the Sixteenth International Conference on Machine Learning, 1999, pp. 239–248.
[39] 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.
[40] D. Wilson, Asymptotic properties of nearest neighbor rules using edited data sets, IEEE Transactions on Systems, Man and Cybernetics 2 (1972) 408–421.
[41] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufmann, San Francisco, California, 2005.
[42] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering: An Introduction, Kluwer International Series in Software Engineering, Kluwer Academic Publishers, Boston, MA, 2000.
[43] X. Zhu, X. Wu, Class noise vs attribute noise: a quantitative study of their impacts, Artificial Intelligence Review 22 (3–4) (2004) 177–210.