Artif Intell Rev (2010) 33:275–306
DOI 10.1007/s10462-010-9156-z

A study of the effect of different types of noise on the precision of supervised learning techniques

David F. Nettleton · Albert Orriols-Puig · Albert Fornells

Published online: 27 January 2010

Abstract Machine learning techniques often have to deal with noisy data, which may affect the accuracy of the resulting data models. Therefore, effectively dealing with noise is a key aspect in supervised learning to obtain reliable models from data. Although several authors have studied the effect of noise for some particular learners, comparisons of its effect among different learners are lacking. In this paper, we address this issue by systematically comparing how different degrees of noise affect four supervised learners that belong to different paradigms. Specifically, we consider the Naïve Bayes probabilistic classifier, the C4.5 decision tree, the IBk instance-based learner and the SMO support vector machine. We have selected four methods which enable us to contrast different learning paradigms, and which are considered to be four of the top ten algorithms in data mining (Yu et al. 2007). We test them on a collection of data sets that are perturbed with noise in the input attributes and noise in the output class. As an initial hypothesis, we assign the techniques to two groups, NB with C4.5 and IBk with SMO, based on their proposed sensitivity to noise, the first group being the least sensitive. The analysis enables us to extract key observations about the effect of different types and degrees of noise on these learning techniques. In general, we find that Naïve Bayes appears as the most robust algorithm, and SMO the least, relative to the other two techniques. However, we find that the underlying empirical behavior of the techniques is more complex, and varies depending on the noise type and the specific data set being processed. In general, noise in the training data set is found to give the most difficulty to the learners.

Keywords Attribute noise · Class noise · Machine learning techniques · Noise impacts

D. F. Nettleton (✉)
Department of Technology, Pompeu Fabra University, Tanger, 122-140, 08018 Barcelona, Spain
e-mail: david.nettleton@upf.edu

D. F. Nettleton · A. Orriols-Puig · A. Fornells
Grup de Recerca en Sistemes Intel·ligents, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Quatre Camins 2, 08022 Barcelona, Spain

1 Introduction

Supervised learning techniques are applied to extract novel, interesting and useful data from real-world problems. An important characteristic of real-world problems is that the training and/or test data frequently contains noise. That is, the quality of the data may be decreased by errors or deviations produced in the data collection phase, as a consequence of human error in translating information or due to limitations in the tolerance of the measurement equipment. This may result in errors in the values of the attributes (attribute noise) or in the class of the instances (class noise). In general, noisy data may bias the learning process, making it more difficult for learning algorithms to form accurate models from the data. Therefore, developing learning techniques that effectively and efficiently deal with these types of data is a key aspect in machine learning. Recently, several works have studied the effect of noise, including attribute noise and class noise, for specific learners such as C4.5 (Zhu et al. 2003; Zhu and Wu 2004). Nevertheless, comparisons of the effect of noise on different learning paradigms are lacking.

The purpose of this work is to provide a fair comparison of the effect of attribute noise and class noise on the models created by four learning techniques that belong to different learning paradigms: (a) the Naïve Bayes probabilistic classifier (John and Langley 1995); (b) the C4.5 decision tree (Quinlan 1993); (c) the IBk instance-based algorithm (Aha et al. 1991) and (d) the SMO (Platt 1998) support vector machine (Vapnik 1995). We have selected four methods which enable us to contrast different learning paradigms, and which are considered to be four of the top ten algorithms in data mining (Yu et al. 2007). The performance of these four systems is compared on a collection of thirteen classification problems which are systematically perturbed with different proportions of attribute noise and class noise. The results enable us to highlight the strong and weak characteristics of the different approaches when presented with different degrees of noise, and to group the techniques in terms of their relative behaviors. The whole analysis is developed from the following initial hypothesis.

Initial hypothesis: Given the characteristics of each learning technique, we initially propose that NB and C4.5 have specific properties which should make them less sensitive to noise. On the other hand, we initially propose that SMO and IBk will be the two learners most sensitive to noise, again as a consequence of specific properties of these techniques. In Sect. 3 of the paper, we will go into the details of the properties of each learning technique, with respect to their robustness to noise, and to justify the initial hypothesis. Therefore, we initially group C4.5 with NB, expecting them to give better results relative to IBk and SMO. However, we highlight that this initial grouping is a theoretical characterization for convenience, and the specific noise tolerance characteristics of the algorithms described in Sect. 3, together with the empirical results of Sect. 5, will bring to evidence a more complex situation.

The structure of the present paper is as follows: in Sect. 2 we summarize related work in noise analysis and different approaches. In Sect. 3 we present, in a summarized form, the four chosen learning techniques and we discuss the theoretical noise tolerance mechanisms of each technique. Section 4 describes the data sets which have been used for the experiments and explains the methodology used for the noise generation. Section 5 presents and analyzes the results, identifying any trends and reaching some conclusions about the performance of the different algorithms for different noise types and percentages. We analyze the results from different points of view: arithmetic mean, geometric mean, rankings, class imbalance, type of noise and specific data sets. Section 6 summarizes the results in terms of the different points of view which give different groupings for the four techniques. Finally in Sect. 7 we give the overall conclusions.

2 Related work

The presence of noise in data is mainly related to the way in which it is acquired and pre-processed from the environment. In this sense, two main sources of noise can be identified: (1) the implicit measurement error introduced by sensors such as in electrical engineering, signal processing, or hardware sensor calibration domains; and, (2) the random error introduced by batch processes or experts when they gather data such as in a document digitalization process. The impact of noise in a data set has as result a partial alteration of the attributes' values and, as a consequence, it influences the algorithms' performance. Therefore understanding its impact in the performance of machine learning algorithms is a key issue for improving their reliability, and it has motivated the study of how to generate and introduce noise in data with the aim of measuring its impact in algorithms. The next sections describe different ways of generating noise and review some of the most relevant papers in the literature related to the effect of noise on machine learning algorithms, with special emphasis on the work of Quinlan (1986, 1993) and Zhu et al. (2003), Zhu and Wu (2004).

2.1 Noise generation

The generation of noise can be characterized in different ways. Firstly, it can be characterized by its distribution, which can be, for example, normal (Gaussian) (Zhu et al. 2003; Zhu and Wu 2004). Secondly, noise can be characterized by where the noise is introduced, which can be in the input attributes, in the output class, in the training data, in the test data, or a combination of all of these. When the noise is in the training data, as a consequence, the learning process and resulting model is impaired by imperfect examples. This situation has been studied previously in the literature for specific algorithms, such as ID3 tree induction (Quinlan 1986) and C4.5 tree induction (Quinlan 1993; Zhu et al. 2003; Zhu and Wu 2004). Finally, noise can be characterized by distinguishing if the magnitude of the generated noise values is relative to each data value of each variable, or if it is relative to the min, max and standard deviation for each variable (Zhu et al. 2003; Zhu and Wu 2004).

Our study offers an additional aspect to the analysis of the effects of noise on supervised learning, given that many studies tend to use just one machine learning method, and do not tend to compare results from different methods. Also, some studies do not clearly explain how they define and generate noise, and why they use one type of noise, and not another.

2.2 The impact of noise in algorithms

In Quinlan (1986), a study is made of the effect of noise on machine learning techniques, and specifically on the ID3 induction algorithm. This paper analyzed how ID3 should handle noise, from a heuristic point of view, for different noise levels which were applied in three different ways: (a) to the values of the most noise-sensitive attribute; (b) to the values of all attributes simultaneously, and (c) to the class information.

In Zhu et al. (2003), Zhu and Wu (2004), Zhu analyzed different types of noise: in the input attributes, in the output class, in the test data, in the training data, in both the training and test data. Zhu realized an exhaustive analysis of noise using the C4.5 algorithm. Some comparative testing is done using different techniques (C4.5, HCV, 1R and Prism).

In current data mining philosophy, some machine learning techniques are considered more "robust" to noise, errors and missing values than others. For example, ID3 cannot process missing values, whereas C4.5 has a heuristic which incorporates a "lost information" calculation. Another approach is to estimate "a priori" the "risk" of there being noise or errors,

and assigning "weights" to variables and/or data ranges to proportionally compensate the risk. Following this approach, one can define a weighting vector which corresponds to given data ranges, for each input variable, and which can be interpreted as the "reliability" of each data range of each variable. In Nettleton (2001) and Nettleton and Muñiz (2001)), weighting coefficients were learned by a genetic algorithm in order to construct a predictive data model, which were used to quantify the relevance of the input variables and the reliability of the data values, respectively. Also, in the context of aggregation operators, Torra (1997) defined the WOWA aggregation operator which uses two weighting vectors to model a data set: vector ρ can be interpreted as corresponding to the "relevance" of the variables, and vector ω can be interpreted as corresponding to the "relevance" of the data values.

With respect to other studies of noise in data, in the literature (Sloan 1988; Angluin and Laird 1988; Goldman and Sloan 1995; Sloan 1995; Meeson et al. 1996; Kearns 1998; Nelson 2005), we have observed that many approaches tend to reflect and be specific to the domain and problem.

In Goldman and Sloan (1995) a study is made of the robustness of Probably Approximately Correct (PAC) learning algorithms when the instance space is $\{0,1\}^n$, and the examples are corrupted by purely random noise affecting only the attributes (and not the labels). The results for *uniform attribute noise*, in which each attribute is flipped independently at random with the same probability, are compared with the results for *product random attribute noise*, where each attribute$_i$ is flipped randomly and independently with its own probability$_i$. PAC represents a formalism for deciding how much data has to be collected in order for a given classifier to achieve a given probability of correct predictions on a given proportion of unseen test data. In Sloan (1995) an evaluation is made of the effect on the quality learning by a PAC algorithm, of the introduction of four types of noise in the data.

Another consideration is whether the noise is present in the training data, or if we can guarantee that the training data is good, but the subsequent test (field) data is that which contains the noise. In Angluin and Laird (1988) a study is made of the problem of learning from noisy examples, that is, how a learning algorithm can cope with incorrect training examples. Angluin states that when a "teacher" makes independent random errors in classifying the example data, the strategy of selecting the most consistent rule for the sample is sufficient, and generally requires a relatively small number of examples, in a situation in which the noise affects less than half the examples. They state that it was possible to estimate the rate of noise using only the knowledge that the rate is less than one half. Finally, in Kearns (1998), a statistical approach to noise tolerant learning is proposed, which considers the problem of learning in the presence of classification noise for the probabilistic learning model of Valiant and its variants. A general formalization of the class of "robust" learning algorithms is defined, from which a novel model is derived for learning from statistical queries. In this model, a learning algorithm is restricted from examining individual examples of the unknown target function.

On the other hand, in the context of data acquisition systems, Nelson (2005) presents a study to evaluate if the operation of near DC or upwards of 1 GHz, can suffer adverse effects from seven types of noise: quantization noise, internal analog-to-digital converter (ADC) noise, power-line noise, time skew, aliasing noise, common-mode noise, and radiated noise (Nelson 2005). In Meeson et al. (1996) a study is made of the quality of data and its effect on the reliable interpretation of EIT images in the clinical environment. A clinical imaging system has to be assessed for each clinical investigation, with comparison to a known standard, or with previous studies. The method depends on calculating the variances of the differences in reciprocity measurements as a function of the distance between the current drive electrodes and the receive voltage electrodes. These measurements fit the 'noise model',

with minimal interference from physiological variability, and permit a figure of merit to be calculated which is a representation of the noise presented at the inputs to the system.

3 Heuristics of chosen algorithms

The present work focuses on analyzing the influence and impact of the noise on different types of learning algorithms. For the purposes of the present study, we have grouped the learning algorithms into two families, as mentioned in the introduction to the paper. Group 1 is comprised of Naïve Bayes and C4.5, and Group 2 is comprised of IBk and SMO. Now we will comment the justification for grouping the learning algorithms in this manner. With reference to Group 1, NB is a Bayes learner which uses conditional probabilities to derive posterior probabilities. As conditional probability values are relatively less sensitive to data errors, we would expect that NB would perform relatively well when compared with the other chosen methods. C4.5 includes some specific aspects to make it less sensitive to noise, such as the detection of over fitting and the gain ratio heuristic.

On the other hand, with reference to Group 2, SMO and IBk rely on each single instance to derive a decision model, and therefore we would expect that SMO and IBk will be the two learners which are most sensitive to noise. Thus, it seems reasonable to group SMO and IBk together in Group 2 as the most sensitive to noise (when compared with Group 1), and NB and C4.5 together in Group 1 as the least sensitive to noise (when compared with Group 1).

The noise tolerant properties of each technique are explained in more detail in the following Sects. 3.1 and 3.2.

We have used the versions of these algorithms which are available in Weka Version 3.5.5 (Hall et al. 2009): (1) Naïve Bayes probabilistic classifier (John and Langley 1995); (2) C4.5 decision tree (Quinlan 1993); (3) IBk instance-based algorithm (Aha et al. 1991) and (4) SMO (Platt 1998) support vector machine (Vapnik 1995). We have selected four methods which enable us to contrast different learning paradigms, and which are considered to be four of the top ten algorithms in data mining (Yu et al. 2007), thus allowing other researchers to compare results.

This section briefly describes how they work, with special reference to each method's inherent noise handling mechanisms, allowing us a better understanding of the results of the impact of noise on them in the experimentation section. We only briefly introduce each learning method given that we assume that the reader already has a basic knowledge of these popular methods, before going into a description of what noise handling capabilities, if any, are inherent in each technique. If the reader wishes more detail about the basic functionality of each technique, s/he can refer to the given corresponding references in the literature.

3.1 Group 1

Group 1 represents the two types of algorithms which we initially presume will be the *least* sensitive to noise.

3.1.1 Naïve Bayes

The Naïve Bayesian (NB) classifier (John and Langley 1995) is a simple probabilistic classifier based on the application of the Bayes' theorem which represents and processes knowledge which includes probabilistic information about data. It is called 'naïve' because it makes two key simplifying assumptions: (a) the predictive attributes are conditionally independent with

respect to the output class; (b) there are no latent or hidden attributes which influence the prediction process. Considering these two assumptions, NB estimates the parameters of a simplified Bayes probabilistic model by considering the relative frequencies of each attribute and value per class in the training data set.

Noise Tolerance: one significant characteristic of Naïve Bayes in general is the assumption of the independence of the different input attributes, which may be an advantage when noise is introduced into the data set. NB is also a Bayes learner which uses conditional probabilities to derive posterior probabilities. As conditional probability values are relatively less sensitive to data errors, this would lead us to expect that NB would perform more favorably with respect to the other chosen learning methods. The numeric estimator precision values are chosen based on the analysis of the training data. As a consequence, the classifier is not an "updateable classifier", which is normally initialized with zero training instances.

In John and Langley (1995), a benchmarking was carried out of two versions of a Bayesian Classifier, both of which model a probability distribution. They are proposed as solutions to the handling of continuous variables by density estimation: the first version assumes normality and models each conditional distribution using a single Gaussian; the second version uses a non-parametric kernel density estimation. Their results demonstrate a significant reduction in error on several of the test data sets.

Version used: We have used the "Naive Bayes" classifier (John and Langley 1995), which uses estimator classes, as implemented in Weka 3.5.5. The following Weka parameters were used: debug = false; useKernelEstimator = false; useSupervisedDiscretization = false. The Weka class is "weka.classifiers.bayes.NaiveBayes".

3.1.2 C4.5—decision tree generator

C4.5 (Quinlan 1993) is an induction algorithm based on an iterative process in which a decision tree that correctly classifies a successively greater number of cases (defined by the 'window size') is iteratively built. In each iteration, a sub-model is executed against the remaining cases (those which are not in the window) and the incorrectly classified cases are given precedence to be included in the next window (which will be $x\%$ bigger than the previous window). It is assumed that the distribution of cases and attribute values is well balanced.

Noise Tolerance: As indicated in Fürnkranz (1997), a principal problem with learning techniques which rely on windowing is that, in noisy domains, the process will eventually incorporate all noisy examples into the learning window, because they will be 'misclassified by a good theory.' Also, the window will typically contain a subset of the original learning examples, which after a few iterations, can give arise to a situation where the proportion of noisy examples in the learning window is much higher than the noise level in the entire data set, which will make learning considerably more difficult.

It is stated in Quinlan (1986), that, "for higher noise levels, the performance of the correct decision tree on corrupted data was found to be inferior to that of an imperfect decision tree formed from data corrupted to a similar level."

It is concluded that a worse result is obtained if the noise is filtered out of the attribute information in the training set, when the test set contains similar levels of noise in the same attributes. Also, it is better to keep the instances which contain attribute noise in the training data set, so as not to eliminate valuable information in other attributes of the instance.

In general, in domains with noise or uncertainty, the system may try to decrease the training error by completely fitting all the training examples. Therefore we should try to avoid 'over-fitting' by not including branches that fit the data too specifically. In order to avoid

'over-fitting' to the noise, two options are open when building a decision tree: (a) pre-prune: stop growing a branch when information becomes unreliable; (b) post-prune: take a fully-grown decision tree and discard unreliable parts. C4.5's method derives a confidence interval from the training data and uses a heuristic limit, derived from this, for pruning, following a standard Bernoulli-process-based method. But, this process is based on possibly "unreliable" statistical assumptions, given that it depends on the training data.

In the context of the partition of the training data set, the heuristic of C4.5 has a key dependence on an information gain calculation to evaluate which attribute to incorporate next, and where to incorporate it in the induction tree. But, any disturbance to the values of an attribute may change its correlation with other attributes and therefore provoke an erroneous decision by C4.5. This problem is addressed in Zhu and Wu (2004), making reference to the 'Information Gain' measure (Hunt et al. 1966), used to classify the correlation direction between attribute A and attribute B, which they also understand as the mutual information between A and B. This measure is given by Eq. 1:

$$\text{Gain}_B(A) = I(A; B) = \sum_a \sum_b P(a, b) \log \frac{P(a, b)}{P(a)P(b)} \tag{1}$$

where a and b indicate the attribute values of attributes A and B, respectively.

Version used: We have used the "J48" classifier (Quinlan 1993), implemented in Weka 3.5.5. The following Weka parameters have been used: binary splits = false; confidence factor = 0.25; debug = false; minNumObj = 2; numFolds = 3; reducedErrorPruning = False; saveInstanceData = False; Seed = 1; subtreeRaising = True; unpruned = False; useLaplace = False. The Weka class is "weka.classifiers.trees.J48"

3.2 Group 2

Group 2 represents the two types of algorithms which we propose will be the *most* sensitive to noise.

3.2.1 IBk—instance based learning

The Instance Based Learning algorithm (IBk) (Aha et al. 1991), retrieves the K most similar cases for proposing a new solution. The basic similarity function used is the Euclidean distance. This mechanism is also known as k-NN, or K-Nearest-Neighbors classifier. We have used the method with $K = 1$.

Noise Tolerance: with reference to the family of 'IBN' (Instance Based) algorithms, the IB1 method relies on each single instance to derive the decision model, hence the decision regions of IB1 can be easily altered by the inclusion or exclusion of a single noisy instance. IB2 was an improved version of IB1 in terms of memory storage requirements; IB3 was an extension to improve tolerance to noisy data by only retaining some exemplars of each class and discarding ones that did not have a good classification history. IB4 and IB5 were described in Aha (1992) as extensions to IB3, which handled irrelevant and novel attributes.

On the other hand, IBk in Weka has a similar approach but with the following differences. The 'IBN' algorithms employ the nearest neighbor approach, which classifies an instance as being a member of the same concept as its most similar instance. The k-nearest neighbor classification function does the same, but takes a majority vote among its k most similar instances (we set k to 1)."

The difference between IBk using 1 nearest neighbor and IB1 in Weka is that IB1 always uses one training instance to make a prediction, whereas IBk might use more than 1. For

example, if there are ties in the distance for the nearest neighbor, they are voted to produce a prediction. On the other hand, IB1 just uses the first one found. Therefore, in terms of noise tolerance, we would expect IBk to perform better than IB1.

Version used: We have used the "IBk" classifier (Aha and Kibler, 1991), implemented in Weka 3.5.5. The following Weka parameters have been used: KNN = 1; Crossvalidate = false; debug = false; distanceWeighting = no distance weighting; meanSquared = false; nearestNeighborSearchAlgorithm = LinearNN—A weka.core.EuclideanDistance; windowSize = 0. The Weka class is "weka.classifiers.lazy.IBk"

3.3 SMO

Support Vector Machines (SVMs) (Vapnik 1995) represent a family of supervised learning methods for classification and regression. Focusing on classification tasks, the objective of SVM is to build a hyperplane that separates instances of different classes and that maximizes the distance (or the margin) with the nearest data points, which are addressed as support vectors. In order to calculate the margin, it calculates two parallel hyper planes, one on each side of the separating hyper plane. These parallel hyper planes are used to process the two data sets. In general, a good separation is achieved when a hyper plane has a greater distance from the neighboring point data values of both classes. This is so because in general a greater margin implies a reduction in the generalization error of the classifier. In Platt (1998), a new algorithm for training SVMs was presented, which is called 'Sequential Minimal Optimization' (SMO).

Noise Tolerance: SMO relies on the support vectors, which are identified by considering each single instance, to derive the decision model. Thus, as in the case of IB1, the hyperplanes of SMO can be easily altered by inclusion or exclusion of a single noisy instance. Also, the implicit interdependence of the input attributes (as they are fused into just two factors as required by the SVM), may also cause difficulties when noise is introduced into the training data, which disrupts the interrelations and correlations between the attributes. Thus, we would expect that SMO will be a learning technique which is more sensitive to noise than the Group 1 techniques (NB and C4.5).

Version used: We have used the SMO algorithm (Witten and Frank 2005), which implements Platt's sequential minimal optimization algorithm (Platt 1998) for training a support vector classifier (Vapnik 1995), implemented in Weka 3.5.5. The following parameters have been used: buildlogisticModels = false; $C = 1.0$; checksTurnedOff = false, debug = false; epsilon = 1.0E-12; filtertype = normalized training data; kernel = Polykernel—C 250007-E1.0; numfolds = −1; randomseed = 1; tolerance parameter = 0.0010. The Weka class is "weka.classifiers.functions.SMO"

3.4 Summary of the noise tolerance characteristics of the techniques

Naïve Bayes: has two strong points: (*s1*) the assumption of the conditional independence (Naive) of the different input attributes; (*s2*) a Bayes learner uses conditional probabilities to derive posterior probabilities.

C4.5: the first strong point (*s1*) is that it derives a confidence interval from the training data and uses a heuristic limit, derived from this, for pruning, following a standard Bernoulli-process-based method. However, a weak point (*w1*) is that this process is based on possibly "unreliable" statistical assumptions, given that it depends on the training data. The second strong point (*s2*) would be the use of the Information Gain measure to classify the correlation

direction between attribute A and attribute B, when choosing the next decision attribute for the tree.

IBk: the main weak point would be (*w1*) that the basic method relies on each support vector (and by extension on each single instance) to derive the decision model, hence the decision regions can be easily altered by the inclusion or exclusion of a single noisy instance. However, as a strong point, (*s1*) if we use IBk, it may use more than one training instance to make a prediction, when there are ties.

SMO: the main weak point (*w1*) would be that the basic method relies on each single instance to derive the decision model. Thus, as in the case of IBk, the hyper planes of SMO could be easily altered by inclusion or exclusion of a single noisy instance. A second weak point (*w2*) would be the implicit interdependence of the input attributes, in contrast to the independence assumption of NB.

4 Design of experiments

To create the test bed, we designed a synthetic data set and selected 6 problems from the UCI repository (Asuncion and Newman 2007). The synthetic data set was designed to have no noise, a high class imbalance, and one redundant variable; so, this problem permitted testing the ability of the learners to deal with under-sampled concepts and to generalize useless attributes regardless of the degree of noise. The other data sets were selected for their different data structures and complexities.

All the data sets consisted of only numerical attributes. Data sets that contained m classes (where $m > 2$) were transformed to m 2-class data sets. To achieve this transformation, we applied the following procedure. For each one of the m classes, we created a data set that contained the instances of the selected class and labeled all the other instances with a new class value. Therefore, this procedure enabled us to analyze whether the learners could extract accurate models for each of the classes of the problem. This process resulted in a total of 13 data sets, whose characteristics are summarized in Table 1. In addition, all the data sets were perturbed with different degrees of attribute and class noise.

In the remainder of this section we explain how the synthetic data set was generated and how the attribute noise and the class noise were added to the problems. Furthermore, we provide a case study that shows the effect of noise. This case study is not to extract general conclusions, but to intuitively show some possible difficulties that learners may tackle to extract accurate models from noisy problems.

With reference to Table 1, we can distinguish three main groups of data sets: (1) those with a greater complexity due to the number of attributes and the relations between them (wbcd, bpa and pim); (2) those with a class imbalance problem (INI, thy2c1, th2yc2, th2yc3, bal2c1); (3) other data sets (bal2c2 and bal2c3, iris2c1, iris2c2, iris2c3). This enables us to analyze the results also in terms of which methods give the best results for more complex data sets, and which methods give the best results for data sets with a class imbalance problem. The chosen data sets are frequently used in the literature for benchmarking, which will facilitate the comparison of our results with those of other investigators In Fig. 1 we see a scenario of over fitting to noisy instances by a rule induction technique.

4.1 INI synthetically generated data set

We have designed a simple artificial problem based on a series of intervals. By creating the data set ourselves with fixed rules/ranges, we can guarantee that the data set is 100%

Table 1 Benchmark data sets for our experiments

Data set	Number of instances	Number of attributes	Number of instances (class1, class2)
wbcd	699	9	458, 241
bpa	345	6	200, 145
pim	768	8	500, 268
INI	210	4	170, 40
thy2c1	215	5	185, 30
thy2c2	215	5	180, 35
thy2c3	215	5	150, 65
bal2c1	625	4	576, 49
bal2c2	625	4	337, 288
bal2c3	625	4	337, 288
iris2c1	150	4	100, 50
iris2c2	150	4	100, 50
iris2c3	150	4	100, 50

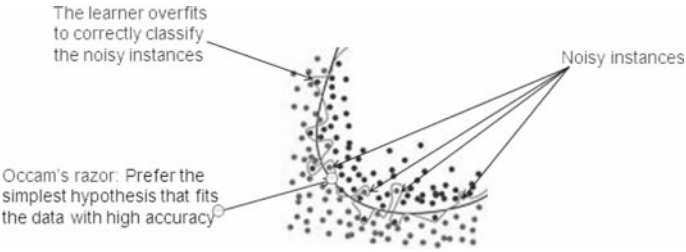

Fig. 1 Scenario of over fitting to noisy instances by a rule induction technique

```
If attribute0 ≤ 0.3 → class 0
       Else If attribute0 ≤ 0.7 and attribute1 ≤ 0.3 → class 1
       Else If attribute0 ≤ 0.7 and 0.33 < attribute1 ≤ 0.7
            → class 0
       Else If attribute0 ≤ 0.7 and attribute1 > 0.7 → class 1
       Else If attribute3 ≤ 0.3 → class 0
       Else If 0.3 < attribute3 ≤ 0.7 → class 1
       Else If attribute3 > 0.7 → class 0
```

Fig. 2 Pseudo-code of rules used to generate instances for the INI synthetic data set

free from noise. On the other hand, the UCI data sets, as they are based on real data (data descriptions of real objects or events), may contain a certain amount of background noise, erroneous values, and so on. We have created the INI data set with 4 attributes which are partitioned so as to create three intervals of equal length, assuming that the attribute has a minimum value of 0 and a maximum value of 1. The intervals go from 0 to 0.3, from 0.3 to 0.7, and from 0.7 to 1. In Fig. 2 we see the rules which were applied to generate the instances. If we run C4.5 on the training data set (fold 8), it produces the pruned tree shown in Fig. 3.

We observe that for this data set there is a significant imbalance of the number of instances per class: the ratio of class1 to class 2 is approx. 3:1. In the training set (fold 8), there are 153 instances of class 1 and 36 instances of class 2. In the test data set (fold 8) there are 17

Fig. 3 Pruned tree generated by rule induction on INI training data set (fold 8)

```
v3 ≤ 0.6
|   v3 ≤ 0.3
|   |   v1 ≤ 0.6
|   |   |   v1 ≤ 0.3: A (26.0/3.0)
|   |   |   v1 > 0.3: B (14.0)
|   |   v1 > 0.6: A (77.0)
|   v3 > 0.3: B (19.0)
v3 > 0.6: A (53.0)
```

Table 2 Basic statistics for INI data set

	Min	Max	Mean	SD
V_1	0.0	1.0	0.752	0.266
V_2	0.0	1.0	0.554	0.359
V_3	0.0	1.0	0.368	0.336
V_4	0.0	0.7	0.177	0.167

Table 3 Variable correlations for INI data set

	V_1	V_2	V_3	V_4
V_1	1	0.64	0.37	0.18
V_2		1	0.66	0.33
V_3			1	0.57
V_4				1

instances of class 1 and 4 instances of class 2. This represents an approximate ratio of 81% to 19%. This imbalance is introduced intentionally to observe the performance of the different algorithms for the minority class.

With reference to Tables 2 and 3, we can see a summary of basics statistics and correlations between variables for the INI data set. We observe that all values for the data variables are in the range from 0.0 to 1.0 and that the highest correlations are V_2 with V_3 (0.66) and V_1 with V_2 (0.64).

4.2 Generation of noise—general discussion

The generation of noise can be classified in different ways: (1) by its distribution, which can be, for example, normal or Gaussian; (2) by where the noise is introduced, which can be in the input attributes, in the output class, in the training data, in the test data, or a combination of all of these; (3) if the magnitude of the generated noise values is relative to each data value of each variable, or if it is relative to the min, max and standard deviation for each variable.

When we consider the distribution (1), values are perturbed under some constraints such as for example altering values according to some data distributions like Normal or Gaussian distribution. In the case of numerical attributes, the distribution is typically Gaussian, whereas for categorical attributes (such as the output class), a Normal distribution is typically used.

For example, if we are modifying a value “14” which corresponds to variable C, in the second case we establish the min (for example, 10) and max(for example, 85) of variable C. Then we generate a random number with Gaussian distribution, between the ranges 10 and 85. This may result in the value “14” being overwritten with the value “55”, for example. In the first case (the generated noise is proportional to the value to be modified), the random

Table 4 Possible situations of noise in data sets

Input data	Output class
A*	D
B	
C	

* A = noise in train, B = noise in test, C = noise in train and test

number generated would be related to the value "14", and so a Gaussian distribution based on the value of "14" as the mean value, would in the majority of cases generate a noise value between 10 and 18. The alteration of numerical values using a Gaussian distribution and based on the min/max values of the corresponding variable will be the method used for generating noise in our experiments (Zhu et al. 2003; Zhu and Wu 2004).

Once we have decided upon the way in which values are altered, the next step is to decide which attributes have to be modified. This is usually done by generating a random integer (which serves as an index into the data vector) using a normal distribution. Noise can occur in different combinations of situations, some of which are listed as follows and summarized in Table 4.

1. Noise in the test data which affects the input variables.
2. Noise in the training data which affects the input variables.
3. Noise in both the train and the test data which affects the input variables.
4. Noise in the training data which affects the output class.

From a practical point of view, it is interesting to study different perspectives of noise presence. This perspective reasons that in the case of data such as that from sensors, we assume that we train in a controlled environment but with real data, which includes any possible errors or background noise in the sensor data.

The training noise may affect the input attributes and/or the output class. When we consider the test data, if the sensor is functioning in the "field" (and therefore generating "test" data), it may start to malfunction progressively over time, due to physical deterioration. The sensor may also temporarily receive "noise" due to some external agent, after which it returns to correct operation.

4.3 Generation of noise—implementation details

In this section we describe how two different types of noise are generated: input attribute noise and output class noise.

4.3.1 Attribute noise

The noise is generated and introduced in the test data sets in the following manner. Firstly, for each variable, we generate N random numbers R_v with a *Gaussian* distribution and within a range between the *max* and *min* of the corresponding variable, to modify the data values, where N is equal to the number of examples in the test data set multiplied by the fraction of noise (F_n) we wish to introduce in the data set. Then we generate random numbers R_c with a *uniform* distribution within the range 1 to the number of cases, which indicates the case whose data value is to be modified. The number of random numbers generated is equal to the number of cases multiplied by the desired fraction of noise F_n . This is repeated for each variable. This procedure for generating *input attribute noise*, is adopted from similar

Table 5 First 5 random values generated for Iris2c1 variables

Value	Variable 1		Variable 2		Variable 3		Variable 4	
	Original value, random value generated	Example to be modified	Original value, random value generated	Example to be modified	Original value, random value generated	Example to be modified	Original value, random value generated	Example to be modified
1	7.7, 6.1	15	3.8, 2.9	12	1.6, 1.5	20	0.2, 2.5	15
2	6.8, 5.0	19	3.0, 3.3	7	4.4, 2.3	14	1.8, 0.3	1
3	5.1, 5.4	6	2.3, 3.1	11	4.4, 6.0	6	1.2, 1.8	20
4	4.4, 5.6	4	2.6, 2.9	18	5.7, 5.6	13	2.0, 0.8	7
5	4.6, 5.2	17	3.2, 4.0	6	5.6, 1.6	4	1.0, 1.2	16

widely referenced work with machine learning algorithms and artificially generated noise, specifically that of Zhu et al. (2003), Zhu and Wu (2004) and Quinlan (1986).

Depending on the noise percentage we wish to achieve, we modify the corresponding number of cases with respect to the complete test data set. For example, if the test data set has 20 examples, a 10% noise would modify 2 examples randomly chosen in the manner we have described, for each variable. Each randomly chosen example would have the data value corresponding to the current variable modified and the randomly generated value between the given ranges as described previously.

In Table 5 we can see an example of random data value generation (first 5 values per variable) for the Iris2c1 data set. Both the data value itself, and the example number to be modified are randomly generated, the former with a Gaussian distribution and the latter with a Normal distribution. For example, in the case of Variable 2, the first example to be modified will be example 12 and its current value (3.8) will be overwritten with the noisy value (2.9). Then, the second example to be modified for Variable 2 will be example 7 whose original value (3.0) will be assigned a noise value of 3.3, and so on. This is repeated for the number of values determined by the current noise percentage, and for each variable.

With reference to Fig. 5a, b, we see the distribution of each of the four variables of the INI data set before and after the introduction of noise. In this case the noise introduced was 50%. Also, the proportions of the output class in each range block are indicated in dark grey and light grey. Black indicates class 1 and grey indicates class 2. We observe that, with the introduction of 50% noise, the distribution of each variable in specific ranges is changed to a certain degree, although the general forms of the distributions remain quite similar.

In the case of 10% noise for a given data set of 20 instances, the pseudo-code the random noise generator is shown in Fig. 4.

Likewise, in Fig. 5c, d, we see the distribution of each of the four variables of the Iris2c1 data set before and after the introduction of noise (50%). This time, we observe that with the introduction of 50% noise, the distributions of variables 1, 3 and 4 are changed to a greater degree, whereas for variable 2, the distribution is very similar.

4.3.2 Class noise

The class noise is generated in the following manner: First we identify (by a simple frequency count), which is the majority class, assuming our data sets only have two output

```
For i = 1 to number_of_variables
num_test_cases=20;
For j = 1 to (num_test_cases * 0.1)
      1. Randomly generate a data value Rv with a Gaussian
         distribution and lying between the min and max
         value for V(i).
      2. Randomly generate an instance number Rc between 1
         and num_test_cases, with a uniform distribution.
      3. Check that the example number Rc has not been
         chosen previously for this variable. If it has
         return to Step 2.
      4. Assign random value to chosen case and variable:
         V(i)C( Rc) = Rv .
```

Fig. 4 Pseudo-code of random noise generator for example of 10% noise and 20 instances

classes, which is the case. For example, if there are 20 1's and 10 2's in a data set of 30 instances, then the 1's will be the majority class. Then, a noise level, N_c is assigned, between 0 and 1.0. For example, if N_c equals 0.1, this implies a noise level of 10%. Now we repeat the following for each instance in the data set: for each instance, a random uniform number R_u is generated, between 0 and 1. If R_u is less than N_c, then we modify the corresponding instance. For a chosen instance, we then check if its class is the majority class (in the example, the 1's), and if so, we change the class value from 1 to the other class (in the example, 2). Otherwise (if the class is not the majority class), we do not make any change. We observe that the distribution of the noise is uniform. This procedure of generating noise for the *output class* is adopted from similar widely referenced work with machine learning algorithms and artificially generated noise, specifically that of Zhu and Wu (2004) and Quinlan (1986).

5 Results

With the experimental methodology and the aspects highlighted by the case studies of the previous section in mind, we are now in position to start with the experimental analysis. As aforementioned, we are interested in comparing the performance of the different learning techniques on data with different proportions of attribute noise and class noise in both the training and the test data sets. We have initially grouped the learning techniques into two groups, based on their characteristics and their hypothetical sensibility to noise. Group 1 (NB and C4.5) represents techniques which it is proposed will be more robust to noise, and Group 2 (IBk and SMO) represents techniques which it is proposed will be more sensitive to noise. In our experiments, we use two measures to evaluate the learners' performance: (1) the average test accuracy and (2) the geometric test accuracy. In the following subsections, these two comparisons are elaborated in more detail, and a discussion of the fundamental differences observed in both analyses is conducted throughout this section. Furthermore, the effect of noise on particular problems (data sets) and noise types, is also studied.

The results are structured in the following manner: in Sect. 5.1, we evaluate the overall performance of the methods using the arithmetic mean of the precision, and the corresponding relative rankings of the methods for each noise combination. In Sect. 5.2, we evaluate the overall performance of the methods using the geometric mean of the precision, and the corresponding relative rankings of the methods for each noise combination. In this section,

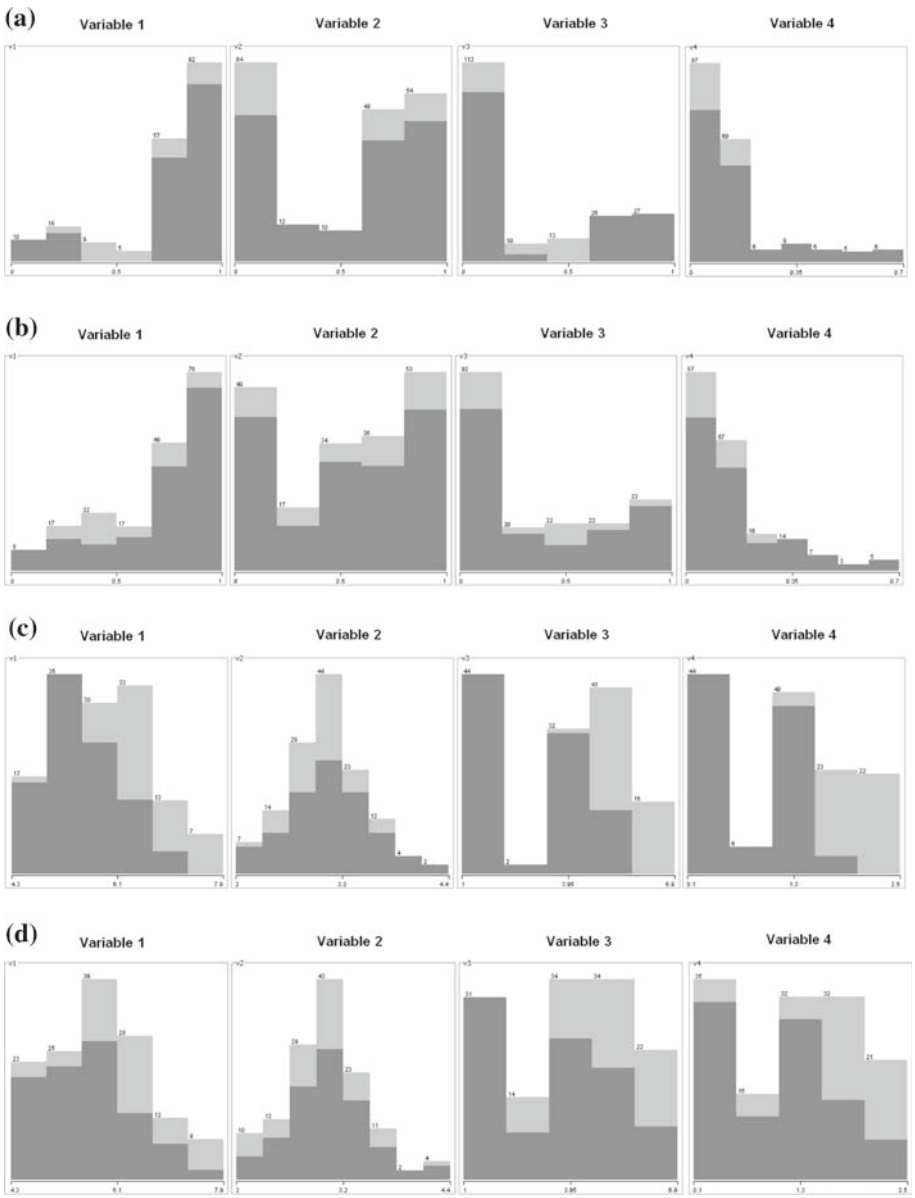

Fig. 5 **a** INI data variable distributions for 0% noise. **b** INI data variable distributions for 50% noise. **c** Iris2c1 data variable distributions for 0% noise. **d** Iris2c1 data variable distributions for 50% noise

we also evaluate the performance of the different methods for data sets with class imbalance. In Sect. 5.3 we evaluate the geometric mean ranking results in more detail in terms of the type and incidence of noise. Finally, in Sect. 5.4 we evaluate the effect of noise in more detail by data set, and for two specific data sets, 'wbcd' and 'thy2c1'.

5.1 Overall performance of the methods: arithmetic mean

As most often done in the literature for these types of comparisons (Zhu et al. 2003; Zhu and Wu 2004), in our first analysis we compare the performance of the learning systems according to the test accuracy of their models. For this purpose, Table 7 represents the summarized results obtained by Naïve Bayes, C4.5, IBk and SMO, respectively, on the original data sets and the data sets perturbed with attribute noise and class noise. More specifically, for each learning method, the systems were trained with the original data set with (1) 0% added noise (train0), (2) 10% added noise (train10), and (3) 50% added noise (train50) in the input attributes. Each one of these configurations was tested with a data set with (1) 0% added noise (test0), (2) 10% added noise (test10), and (3) 50% added noise in the input attributes (test50). This resulted in nine combinations of training/test data sets with which we evaluated the capability of the learners to deal with noise in both the training and the test examples. In addition, we also included two configurations that were used to evaluate the learners' performance in data sets with (1) 10% of class noise and (2) 50% of class noise in the training instances. Therefore, these two last configurations enable us to study whether the learners can obviate a certain proportion of mislabeled training instances. As we proceed, several conclusions are drawn from these results. That is, we first statistically compare the results and then we analyze the results obtained with particular data sets.

Our first concern was to analyze which algorithms performed the best for each type of noise. For this purpose, we first compared the results obtained by the different methods for each proportion of noise in the training and test instances. This resulted in 11 different comparisons of the four learners. Then, for each comparison, we applied the Friedman's procedure to test the null hypothesis that all the learners performed the same, on average (Friedman 1937, 1940). In addition, we computed the average rank of each method as follows. For each data set, we ranked the learning algorithms according to their performance; the learner with highest accuracy held the 1st position, whilst the learner with the lowest accuracy held the last position of the ranking. If a subset of learners had the same performance, we assigned the average rank of the subset to each of the learners in the subset. Table 6 summarizes the p-values returned by the Friedman's procedure and the average rank and the standard deviation of the rank obtained by each learning system for each degree of noise.

With reference to Table 6, the Friedman's test enabled us to reject the null hypothesis that all the learners performed the same on average, at $\alpha = 0.10$, for the problems with (1) 0% noise in training and 50% noise in test, (2) 50% noise in training and 0%, 10%, and 50% noise in test, and (3) 50% class noise in training.

It is important to note that the results shown in Tables 6 and 7 are calculated using the arithmetic mean of the precision for the output classes. This will give a biased result when the learner favors a majority output class in order to improve the overall precision. In Sects. 5.2 to 5.4, we calculate the precisions using the geometric mean, which considers the precision for each output class, individually, and we compare the results.

In terms of the arithmetic mean precisions, we can see, in Table 6, that the first and second best techniques for each category of noise (reading horizontally) are indicated with a grey background. Thus, for example, it can be clearly seen that Naïve Bayes is the method that presents better average ranks in general (is the best method on average in five degrees of noise), followed by IBk and SMO (being the best method on average in three degrees of noise), and C4.5 (which is only the best method for one degree of noise). Therefore, we could say, in terms of rankings, that these initial results support the hypothesis that the Group 2 techniques (IBk, SMO) give similar overall performance, whereas the Group 1 techniques

Table 6 Comparison of the average rank (and standard deviation) of the test accuracy obtained by Group 1 (Naïve Bayes and C4.5) and Group 2 (IBk and SMO)

	Rankings based on Arithmetic Test Accuracy (1=highest, 4=lowest)				
	NB	**C4.5**	**Ibk**	**SMO**	**Friedman**
tr0 ts0[†]	**2.23 ± 0.98**	2.77 ± 1.10	2.46 ± 1.09	2.54 ± 1.22	0.746
tr0 ts10	2.35 ± 0.96	3.00 ± 1.05	**2.15 ± 1.13**	2.50 ± 1.10	0.348
tr0 ts50	2.46 ± 1.22	2.92 ± 0.92	2.85 ± 1.12	**1.77** ± 0.73	***0.074***
tr10 ts0	**2.23 ± 0.99**	2.54 ± 1.03	2.65 ± 1.13	2.58 ± 1.25	0.830
tr10 ts10	2.69 ± 0.93	**2.42 ± 1.03**	**2.42 ± 1.24**	2.46 ± 1.22	0.936
tr10 ts50	2.50 ± 1.16	2.65 ± 0.81	2.85 ± 1.13	**2.00 ± 1.02**	0.329
tr50 ts0	**1.92 ± 1.02**	2.04 ± 0.88	3.42 ± 0.75	2.62 ± 1.11	***0.009***
tr50 ts10	**1.92 ± 0.84**	2.35 ± 0.76	3.35 ± 0.90	2.38 ± 1.20	***0.024***
tr50 ts50	**1.81 ± 0.77**	2.42 ± 0.91	3.73 ± 0.54	2.04 ± 1.03	***0.000***
tr10c ts0[‡]	2.38 ± 1.08	2.31 ± 1.08	3.23 ± 0.73	**2.08 ± 1.18**	0.105
tr50c ts0	1.77 ± 0.91	2.62 ± 0.50	**1.62 ± 0.63**	4.00 ± 0.00	***0.000***
Mean	**2.21**	**2.55**	**2.79**	**2.45**	

The last column provides the *p*-value returned by Friedman's test
Bold values represent the most significant values
[†] tr 0 = training set with 0% noise, ts 0 = test set with 0% noise, ts 10 = test set with 10% noise, etc
[‡] tr10c = output class of training set with 10% noise, etc

Table 7 Summarized test accuracies (in percentage) for all data sets and noise configurations (arithmetic means)

	Noise in test			Noise in train			Noise in train and test			Noise in train label		Mean precision
	tr0 ts0	*tr0 ts10*	*tr0 ts50*	*tr10 ts0*	*tr10 ts10*	*tr10 ts50*	*tr50 ts0*	*tr50 ts10*	*tr50 ts50*	*tr10c ts0*	*tr50c ts0*	
NB	90.10	87.71	73.73	89.23	87.29	75.84	88.10	86.43	77.12	86.91	68.22	82.78
C45	89.30	86.73	73.67	90.05	87.81	75.84	86.30	84.77	75.68	89.09	53.99	81.20
IBk	90.80	88.56	74.26	88.95	86.92	75.50	82.30	81.16	72.60	86.16	60.54	80.70
SMO	87.20	85.82	77.27	87.39	85.76	77.91	84.80	83.44	77.03	86.57	39.44	79.33

give the best (NB) and the worst (C4.5) performance. However, no strong grouping trends for the techniques are evident over all noise types.

We also note that SMO gives relatively high rankings for high noise percentages in the data attributes, but this is due to the strategy of the method, which is to favor the majority

Table 8 Summarized test accuracies (in percentage) for all data sets and noise configurations (geometric means)

	Noise in test			Noise in train			Noise in train and test			Noise in train label		Mean Precision
	tr0 ts0	*tr0 ts10*	*tr0 ts50*	*tr10 ts0*	*tr10 ts10*	*tr10 ts50*	*tr50 ts0*	*tr50 ts10*	*tr50 ts50*	*tr10c ts0*	*tr50c ts0*	
NB	77.85	75.20	52.66	76.57	73.94	56.73	71.74	69.50	53.09	76.33	67.59	68.29
C45	80.23	77.77	61.39	80.94	77.77	59.54	64.69	61.59	46.93	80.15	46.98	67.09
IBk	81.99	79.29	58.50	79.87	76.53	60.49	69.46	66.34	53.10	79.15	63.81	69.87
SMO	60.62	59.81	45.06	60.65	58.57	45.43	46.97	43.91	32.91	65.76	17.42	48.83

Table 9 Comparison of the average rank (and standard deviation) of the geometric test accuracy obtained by Group 1 (Naïve Bayes and C4.5), and Group 2 (IBk and SMO)

	Rankings based on Geometric Test Accuracy (1=highest, 4=lowest)				
	NB	**C4.5**	**Ibk**	**SMO**	**Friedman**
tr0 ts0†	**1.96 ± 1.05**	2.88 ± 0.96	2.35 ± 0.85	2.81 ± 1.22	0.182
tr0 ts10	**1.96 ± 0.88**	2.81 ± 1.15	2.31 ± 0.88	2.92 ± 1.21	0.163
tr0 ts50	2.31 ± 1.25	**2.23 ± 0.93**	2.38 ± 1.04	3.08 ± 1.12	0.300
tr10 ts0	**2.12 ± 0.96**	2.58 ± 1.04	2.50 ± 1.04	2.81 ± 1.22	0.537
tr10 ts10	**1.96 ± 0.88**	2.42 ± 0.95	2.65 ± 1.31	2.96 ± 1.01	0.212
tr10 ts50	**1.96 ± 1.05**	2.35 ± 1.03	2.50 ± 0.96	3.19 ± 1.03	0.082
tr50 ts0	**1.92 ± 1.02**	2.54 ± 0.78	2.54 ± 1.27	3.00 ± 1.14	0.188
tr50 ts10	**1.92 ± 0.93**	2.69 ± 0.75	2.54 ± 1.20	2.85 ± 1.33	0.260
tr50 ts50	**1.96 ± 0.92**	2.46 ± 0.78	2.31 ± 1.32	3.27 ± .1.01	0.059
tr10c ts0‡	**2.19 ± 0.99**	2.50 ± 1.12	2.58 ± 0.95	2.73 ± 1.30	0.729
tr50c ts0	1.69 ± 0.85	2.69 ± 0.48	**1.62 ± 0.65**	4.00 ± 0.00	0.000
Mean	**1.99**	**2.56**	**2.39**	**3.05**	

The last column provides the *p*-value returned by Friedman's test
Bold values represent the most significant values
† tr0 = training set with 0% noise, ts0 = test set with 0% noise, ts 10 = test set with 10% noise, etc
‡ tr10c = output class of training set with 10% noise, etc

class. This fact becomes clearly evident in Sect. 5.2 (Table 9) when we use the geometric mean, instead of the arithmetic mean, to calculate the overall precisions.

Table 7 is a summary of the detailed results tables for each learning method. We observe that in terms of the mean arithmetic precision for all data sets and noise permutations, Group 1 (NB and C45) have a slight advantage over Group 2 (IBk and SMO). However, for noise in training and test data, SMO is the best classifier for high noise percentages (due to bias

on the majority class), followed by NB. For a high noise percentage in the training label, NB (68.22) and IBk are the best classifiers, and SMO (39.44) is significantly the worst.

Note that in Tables 7 and 8, we can take the mean of the average precisions, given that all the original values have the same cardinality, that is, the same number and type of *N*-fold cross-validation tests, data sets and noise configurations.

5.2 Noise combined with the class imbalance problem: geometric mean

The results provided in the previous section enabled us to extract valuable observations of the performance of the different algorithms on noisy data. However, as we used the arithmetic test accuracy as the performance measure, we could not evaluate how the different learners modeled the different classes of the data sets. In addition, it is well known that the arithmetic test accuracy is biased toward the majority class in the training data set, which may lead to the extraction of poor conclusions.

For example, in problems with 90% of instances of class A and 10% of instances of class B, a model that always predicts the majority class would have 90% accuracy. Nevertheless, the model would not reflect any novel knowledge. This effect may be more severe when noise is added; that is, as the instances of different classes begin to overlap due to the added noise, the learning system may decide to predict the majority class to minimize the global error. Therefore, the final model, which appears quite accurate, would not reflect knowledge that is useful for human experts. As we proceed, it is proposed to use the test geometric mean as a more reliable performance measure, repeat the same experiments of the previous section but using the new performance metric, and study the differences between them.

Tables 16 to 19 give the detailed geometric test accuracies obtained by the four learners on the data sets with different degrees of noise, and Table 8 is a summary of these four tables. With reference to Table 8, we observe that in terms of the average geometric precision for all data sets and noise permutations, IBk, NB and C.45 have the best results, with 69.87, 68.29 and 67.09, respectively, and SMO is the worst performer with a significantly lower precision (48.83). However, for noise in training or test data, C45 and IBk are the best classifier for high noise percentages, followed by NB. For a high noise percentage in the training and test, IBk (53.10) and NB are the best classifiers, and SMO (32.91) is significantly the worst. For a high noise percentage in the training label, NB (67.59) and IBk are the best classifiers, and SMO (17.42) is significantly the worst.

The geometric mean is said to be nonbiased toward the majority class in the learning data set since it considers the accuracy of each class regardless of the number of examples of the class in the training data set. Table 9 summarizes the statistical analysis of the results in terms of rankings.

With reference to Table 9, several conclusions can be drawn from the statistical analysis and the particular results on each noise combination. First, note that Friedman's test only rejects the null hypothesis that all learners perform the same on average, with α=0.10, for (1) the problems with 10 and 50% noise in the training data set and 50% noise in the test data set and (2) for the problem with 50% class noise and 0% noise in the test data set. For all the other cases, the models created by all the learners are not statistically different. Nevertheless, note that, for all the configurations, except for (1) 0% noise in the training data set and 50% noise in the test data set and (2) 50% class noise in the training data set and 0% noise in the test data set, the most accurate models are built by Naïve Bayes.

In general, in terms of the first and second rankings, indicated in grey in Table 9 we observe that NB and IBk are the best, followed by C4.5. SMO comes last in the ranking for all noise types, and has therefore clearly suffered a major loss in precision when we use the

geometric mean when compared to the arithmetic mean used in Sect. 5.1 (Tables 6 and 7). As we mentioned, this is due to the technique having a low precision on the minority class label. We observe that although IBk wins second ranking place on more occasions, C4.5 wins for some of the greater noise percentages (tr0, ts50; tr10, ts10; tr10, ts50), that is, when the noise is in the training or in the test data sets, but not in both.

IBk, on the other hand, gives better results for lower noise percentages and when the noise is in both the train and the test (tr50, ts0; tr50, ts10; tr50, ts50). In terms of general groupings of the techniques, a strong trend is not clear, although we could isolate NB (designated as Group 1) on the one hand and SMO (designated as Group 2) on the other. In terms of secondary rankings, we could then group together IBk and C4.5.

5.3 Analysis of the results in terms of the type of noise: noise in the test data set, noise in the training data set, noise in training and test, noise in the output class used for training

With reference to Table 9, we now perform a more detailed analysis of the relative rankings, considering where the noise is (in the test, training or output class), reveals the following trends:

1. For noise only in the test data (first 3 rows of Table 9), the relative rank position of C4.5 with respect to the other techniques gets significantly better for the highest noise percentage (50), whereas the relative ranking of the other three techniques gets progressively worse.
2. For noise only in the training data (rows 1, 4 and 7 of Table 9), we again observe a progressive relative improvement of C4.5 with respect to the other techniques, for increasing test noise, whereas IBk and SMO lose relative ranking. In the case of Naïve Bayes, it loses ranking (1.96 → 2.12) from 0 to 10% noise, but recovers again (2.12 → 1.91) for 10–50% noise.
3. For noise in the training and test data (rows 1, 5 and 9 of Table 9), we observe that C4.5's ranking gets significantly better (2.88 → 2.42) for 0–10% noise and slightly worse (2.42 → 2.46) for 10–50% noise. In contrast, NB has a very good resistance in the same noise situation, maintaining as constant its high ranking (1.96). On the other hand, IBk gets worse for 0–10% noise (2.35 → 2.65) then better for 10–50% noise (2.65 → 2.31). Finally, SMO gets progressively worse for increasing noise in training and test (2.81 → 2.96 → 3.27).
4. For noise in the output class (rows 1, 11 and 12 of Table 9), we observe that NB and IBk worsen their relative performance for 0–10% noise (2.35 → 2.58 for IBk and 1.96 → 2.19 for NB), and then improve for 10–50% noise (2.58 → 1.62 for IBk and 2.19 → 1.69 for NB). In the case of C4.5, it improves ranking from 0–10% noise and then loses slightly for 10–50% noise. Finally, SMO improves slightly to 2.73 for 0–10% noise, and then worsens significantly to 4.00 for 10–50% noise.

Technique groupings: From the observations of the tendencies of each technique for different percentages and types of noise, we can summarize the following as possible groupings: for noise in the training data (point 2, above) we can group IBk and SMO as having similar behavior; for noise in the output class (point 4) we can group NB with IBk. SMO seems to have a singular behavior relative to the other three techniques. C4.5 also behaves differently for noise in the test data (point 1) and in the training data (point 2), in which cases its relative ranking with respect to the other three techniques progressively improves.

Table 10 Ranking of the noise types with the highest precisions per technique

	Ranking of noise types with highest precisions per technique			
	1st	**2nd**	**3rd**	**4th**
NB	tr10te0	tr10cte0	tr0te10	tr10te10
C4.5	tr10te0	tr10cte0	tr10te10	tr0te10
IBk	tr10te0	tr0te10	tr10cte0	tr10te10
SMO	tr10cte0	tr10te0	tr0te10	tr10te10

Table 11 Ranking of the noise types with the lowest precisions per technique

	Ranking of noise types with lowest precisions per technique			
	1st	**2nd**	**3rd**	**4th**
NB	tr0te50	tr50te50	tr10te50	tr50cte0
C4.5	tr50te50	tr50cte0	tr10te50	tr0te50
IBk	tr50te50	tr0te50	tr10te50	tr50cte0
SMO	tr50cte0	tr50te50	tr50te10	tr0te10

With reference to Table 10, we observe that all techniques predict best tr10te0; tr10cte0 comes 2nd for NB and C4.5, and 3rd for IBk; tr0te10 comes 2nd, 3rd or 4th for all techniques; tr10te10 comes 3rd or 4th for all techniques. It can be seen that the ranking of noise types with highest precision is not very discriminative for the techniques. The only aspect we can differentiate is that SMO had tr10cte0 as 1st ranked, although this can be partially explained by the relatively low precisions obtained in other noise types.

With reference to Table 11, as is to be expected, we observe that high noise percentages (50%) predominate the most difficult noise types to predict. C4.5 and IBk are the only two techniques which coincide in their most difficult noise type (tr50te50), although this data type comes in the first two ranked, for all techniques. SMO and C4.5 have particular difficulties with tr50cte0 (noise in the output label, also taking into account the mean precision).

In order to try to group the techniques in terms of these rankings we will calculate the 'Hamming distance'.

Hamming distance: we define the hamming distance in the following manner, for the first four ranked noise types: if a noise type for technique A does not appear for technique B, distance = 10; if a noise type for technique A appears for technique B and also is in the same ranked position, distance = 0; if a noise type for technique A appears for technique B but in another rank position, the distance is equal to the difference between the rankings. For example, if noise type 'tr0te50' is ranked in 1st position for technique NB, and this same noise type is ranked in 4th position for technique C4.5, then the distance is calculated as 4−1 = 3. The distances are summed for each of the four rankings, for each technique with every other technique.

If we calculate the hamming distance for the lowest precision rankings (Table 11), the following distances are obtained: NB with C4.5, 6; NB with IBk, 2; NB with SMO, 23; C4.5

with IBk, 4; C4.5 with SMO, 22; IBk with SMO, 23. Therefore, in terms of these distances, NB would be grouped with IBk (lowest distance of 2), C4.5 with IBk (distance of 4), and SMO would be in its own group.

5.4 The effect of noise on specific data sets

Having extracted conclusions on the general performance of the different learners, in this section we now focus on how the noise affects the different techniques in particular data sets.

5.4.1 Results for data sets with the class imbalance problem

We now study the detailed data set results technique by technique with reference to Tables 16 to 19 (included at the end of the paper). Interesting conclusions can be extracted from the analysis of the results obtained with each individual data set. The results denote that there are some data sets that are especially difficult for all learners. For example, note that all of the techniques have had great difficulties with the 'bal2c1' data set. As a first hypothesis we can attribute this to the high class imbalance of this data set (576 : 49), and a zero precision for the minority class which gives an overall geometric precision of zero (rounded to two places after the decimal point). Similarly, the 'INI' problem poses particular difficulties to Naïve Bayes (Table 16) which was one of the data sets with the lowest precision for this technique. Finally, for SMO (Table 19), some of the worst results are again obtained with the INI data set. Finally, we observe some that SMO had some specific problems for 'thy2c1' and 'thy2c3' data sets, for 50% noise in the train only, 50% noise in training and test, and 50% noise in the output class.

If we look at the results for 'INI' when the noise is in both the training and the test data sets, IBk (Table 18) gave the best overall results, where the other techniques achieved a zero precision (NB, SMO) or below 20% (C4.5).

5.4.2 Results for data sets without the class imbalance problem

Looking in more detail at the case of C4.5 (Table 17), we observe that the data sets with the lowest precisions are 'bpa' and 'pim'. These data sets do not have any significant class imbalance. In the case of IBk (Table 18), we observe that the data sets with the lowest overall precisions are again 'bpa' and 'pim'. In the case of Naïve Bayes (Table 16), the data set with the lowest precision is 'bpa'. C4.5 had a particular difficulty with 'bpa' for noise in the train and test with precisions below 12%. The precisions of C4.5 for 'pim' are also inferior the general precision of the other data sets. Finally, for SMO (Table 19), the worst results are again obtained with the 'bpa' data set. SMO also had particular difficulties with the 'iris2c3' data set, which did not occur for the other three techniques.

5.4.3 Results for all data sets

With reference to Table 12, if we group the techniques by a ranking based on the data sets with the lowest precisions per technique, we could say that C4.5 and IBk have the most similarity. We also note that the two techniques which gave relatively good results with the 'INI' data set, were IBk and C4.5, whereas for this data set, NB and SMO had very low precisions for all noise types.

If we calculate the Hamming distance on Table 12, using the same procedure as used for the noise types in Sect. 5.3, we obtain the following 'distances': NB with C4.5, 4; NB

Table 12 Ranking of the data sets with the lowest precisions per technique

	Ranking of data sets with lowest precisions per technique			
	1st	2nd	3rd	4th
NB	bal2c1	ini	bpa	pim
C4.5	bal2c1	bpa	pim	ini
IBk	bal2c1	bpa	pim	ini
SMO	ini	bal2c1	iris2c3	bpa

Table 13 Ranking of the data sets with the highest precisions per technique

	Ranking of data sets with highest precisions per technique			
	1st	2nd	3rd	4th
NB	wbcd	iris2c2	thy2c2	bal2c2
C4.5	wbcd	iris2c2	iris2c1	bal2c3
IBk	iris2c2	wbcd	iris2c3	iris2c1
SMO	wbcd	iris2c2	bal2c2	bal2c3

with IBk, 4; NB with SMO, 13; C4.5 with IBk, 0; C4.5 with SMO, 16; IBk with SMO, 16. Therefore, in terms of these distances, NB would be grouped with IBk and C4.5 (lowest distance of 4), C4.5 with IBk (distance of 0), and SMO would be in its own group.

With reference to Table 13, we observe that two data sets which are easiest to predict are 'wbcd' and 'iris2c2'. If we look at the third and fourth ranked data sets, NB and SMO have 'bal2c2' in common, whereas C4.5 and IBk have 'iris2c1' in common and C4.5/SMO have 'bal2c3' in common.

5.4.4 'Wbcd' data set

For the sake of compactness, we only consider two particular data sets in detail: (1) the 'wbcd' and (2) 'thy2c1' data sets. The former represents a data set without a class imbalance problem (66 /34%), and the latter represents a data set which has a class imbalance problem (86/14%). For this purpose, Figs. 6, 7 show tendencies for the four machine learning techniques, contrasted for different noise percentages (0, 10 and 50%), and given different types of noise for 'wbcd' and 'thy2c1'. In what follows, we analyze how the quality of the models decreases as the degree of noise increases in these two data sets.

We first analyze the results obtained with the 'wbcd' problem. Figure 6a represents the case when there is 0% noise in the *training* data set, and we progressively increment the noise in the *test* data set with 0, 10 and 50% noise. A first observation we can make is that NB goes from being the equally most precise method at 0% noise, to being the third most precise at 10%, and finally has equal worst precision at 50% noise. We could say that the Group 2 techniques (SMO and IBk) have suffered the smallest loss of precision of approx. 97–84 and 96–81% respectively, whereas the Group 1 techniques (NB and C4.5) have suffered a greater loss of approx. 97–75 and 93–75%, respectively. In general we observe that the precision loss is ranging between approximately 22 percentage points (97–75%).

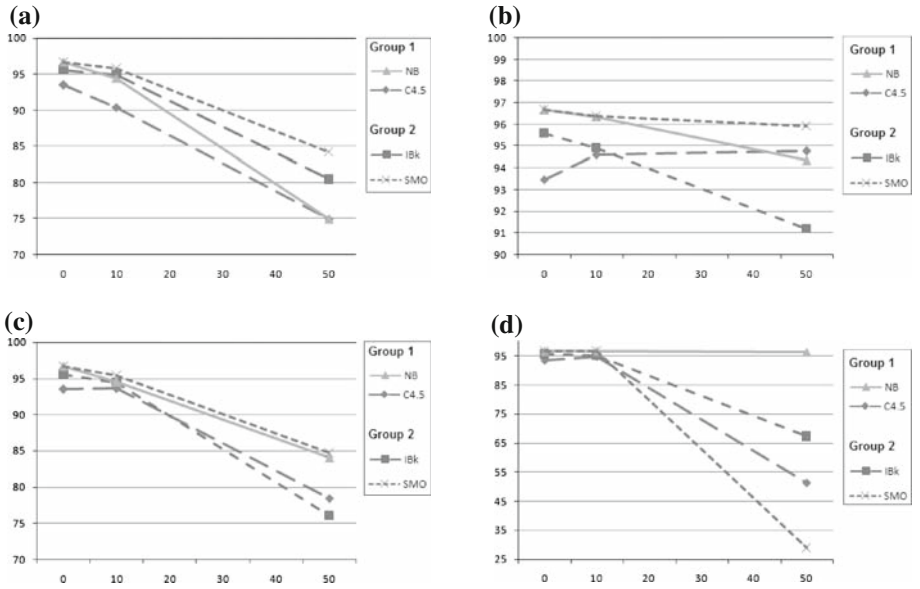

Fig. 6 Analysis of the evolution of the accuracy as the noise increases in the 'wbcd' data set. **a** '*wbcd' data set* with tra0 and incrementing test noise. **b** '*wbcd' data set* with ts0 and incrementing training noise. **c** '*wbcd' data set* with incrementing training and test noise. **d** '*wbcd' data set* with ts0 and incrementing class noise

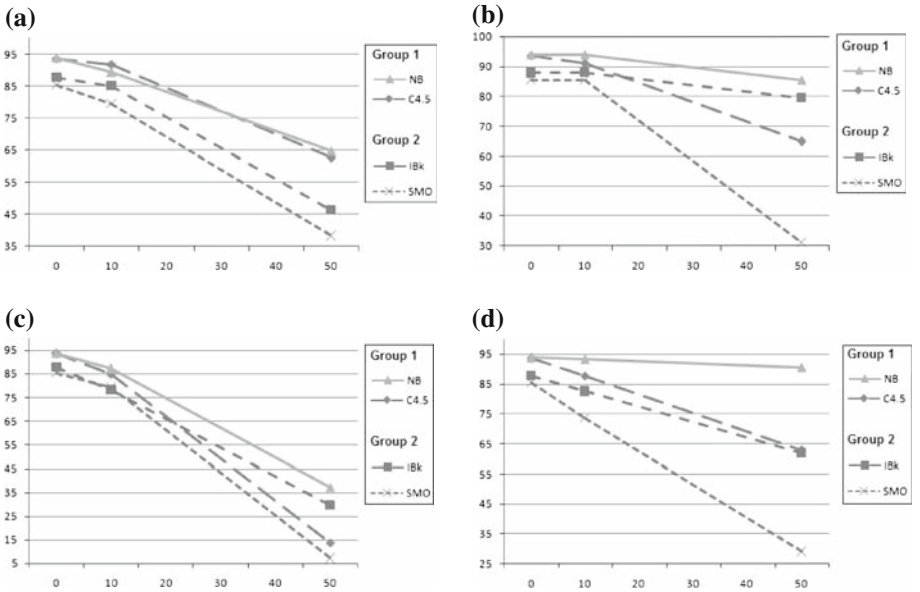

Fig. 7 Analysis of the evolution of the accuracy as the noise increases in the 'thy2c1' data set. **a** '*thy2c1' data set* with tra0 and incrementing test noise. **b** '*thy2c1' data set* with ts0 and incrementing training noise. **c** '*thy2c1' data set* with incrementing test noise. **d** '*thy2c1' data set* with ts0 and incrementing class noise

Figure 6b represents the case when there is 0% noise in the *test* data set, and we progressively increment the noise in the *training* data set with 0, 10 and 50% noise. We would expect noise in the training data to affect the learning process, and so the result for the distinct techniques and groupings of techniques would be more different due to their different approaches to learning. From Fig. 6b we can observe this is clearly the case for C4.5, which progressively improves its precision as the noise increases. With respect to the other techniques, we observe that IBk loses most precision, SMO the least and Naïve Bayes has an intermediate performance. In general we observe that the precision loss is between a range of approx. 6 percentage points (97–91%), which is a much smaller precision loss than that which we observed in Fig. 6a (when the noise is in the test data).

Figure 6c represents the case when there is a progressive increment (0, 10 and 50%) of noise both in the *test* data set and in the *training* data set. The first observation we can make with respect to Fig. 6c is that it has similar general characteristics to Fig. 6a (noise only in test). The range of precision loss is also similar, approx. 97–75%, which is significantly greater than the case when the noise is only in the test data (Fig. 6b). However, when we study which methods have suffered greater precision loss, we observe that NB is now equal best (instead of equal worst) at 50% noise, and C4.5 has got slightly better, and IBk now comes last at 50% noise, having lost 10 percentage points with respect to its performance for noise only in the test (Fig. 6a).

Figure 6d represents the case when there is a progressive increment of noise in the output class of the *training* data set (0, 10, 50%), and no noise in the *test* data set. The graph of Fig. 6d shows a different result to those of the previous Fig. 6a–c. Firstly, we can highlight that the precision loss is much greater for all methods, with the exception of NB. The precision loss range overall is approx. 95–27%, with SMO suffering the worst precision loss (27% precision at 50% noise). We also observe that C4.5 is the third worst performer with 48% precision at 50% noise, followed by IBk with 66% precision at 50% noise. The result of NB is relatively good, and this is confirmed by its performance with the other data sets for this type of noise (in the output class of the training data). We can also conclude that overall, SMO shows a relatively good performance for the 'wbcd' data set, with the exception of noise in the output class (Fig. 6d).

In terms of possible groupings of the techniques in terms of their observed behavior with the 'wbcd' data set, we can identify the following: in Fig. 6a (incrementing test noise), SMO and IBk (Group 2) follow a similar tendency; in Fig. 6c (incrementing training and test noise), SMO and NB follow a similar tendency, and IBk behaves similarly to C4.5; in Fig. 6d, IBk and C4.5 follow similar paths between the trajectories of NB (upper) and SMO (lower). Therefore there is not a clear behavior pattern in terms of our original technique groupings, and the behavior depends on the type of noise.

5.4.5 Thy2c1 data set

Now we consider the 'thy2c1' problem, which has 5 variables and a high class imbalance (86% class1 and 14% class2). Figure 7a represents the case when there is 0% noise in the *training* data set, and we progressively increment the noise in the *test* data set with 0%, 10% and 50% noise.

We observe that for the 'thy2c1' data set (Fig. 7a), the loss in precision for all methods is much greater than in the case of the 'wbcd' data set (Fig. 6a). The range of precision loss is 95–37%. We observe this time that SMO is the worst performer with a precision of only 37% at 50% noise, whereas the Group 1 techniques (Naïve Bayes and C4.5) are the best performers with approx. 64% precision at 50% noise.

Figure 7b represents the case when there is 0% noise in the *test* data set, and we progressively increment the noise in the *training* data set with 0, 10 and 50% noise. Again we observe a much greater precision loss (95–30%) for increasing noise in 'thy2c1', when compared to the 'wbcd' data set. The worst performer is SMO, and the best is again Naïve Bayes.

Figure 7c represents the case when there is a progressive increment of noise both in the *test* data set and in the *training* data set. Both data sets increment with the same percentage of noise, that is, 0, 10 and 50%. We observe again a much greater precision loss for the 'thy2c1' data sets for all methods. In this case (Fig. 7c) we see that none of the methods maintains a reasonable precision for 50% noise (in training and test). Relatively, Naïve Bayes has the best precision (35%) and SMO the worst (7%) at 50% noise.

Finally, Fig. 7d represents the case when there is a progressive increment of noise in the output class of the *training* data set (0, 10, 50%), and no noise in the *test* data set. In this case, we observe in Fig. 7d ('thy2c1' data set) a result which is very similar to that of Fig. 6d ('wcbd' data set): SMO is the worst performer and Naïve Bayes is the best with very little precision loss at 50% noise in the output class. C4.5 and IBk converge to the same precision (approx. 62%) at 50% noise.

The overall results provided in Figs. 6 and 7 show that SMO is in general relatively the worst performer and Naïve Bayes is relatively the best. NB shows good results for noise in the output class, with respect to the other methods (Figs. 6d and 7d). In general, it also appears that noise in the test set is more difficult for the methods to deal with, indicated by a more significant loss in precision (Figs. 6a and 7a).

In terms of possible groupings of the techniques in terms of their observed behavior with the 'thy2c1' data set, we can identify the following: in Fig. 7a (incrementing test noise), NB and C4.5 follow a similar tendency (Group 1), as do IBk and SMO (Group 2); in Fig. 7b, NB and IBk follow a similar tendency; in Fig. 7c (incrementing training and test noise), all methods follow a similar tendency, although NB more closely follows IBk and C4.5 more closely follows SMO; in Fig. 6d, IBk and C4.5 follow similar paths between the trajectories of NB (upper) and SMO (lower). Therefore, as with the 'wbcd' data set, 'thy2c1' shows that there is not a clear behavior pattern in terms of our original technique groupings, and the behavior depends on the type of noise, and the data set. The only noise configuration which has shown the same relative technique behavior between data sets is for the noise in the output class (compare Figs. 6d and 7d).

In conclusion, if we compare the results of the 'wbcd' data set with the results of the 'thy2c1' data set, we observe that the methods are sensitive to the specific data set characteristics. We recall that the 'thy2c1' data set has a class imbalance, whereas 'wbcd' does not.

6 Summary

In this section we summarize the results in terms of different criteria: (6.1) grouping of techniques in terms of behavior for increasing noise percentages and types; (6.2) grouping of techniques in terms of behavior for type of noise; (6.3) grouping of techniques in terms of behavior for specific data sets; (6.4) grouping of techniques in terms of geometric rankings for noise percentages and types (Table 9); (6.5) Grouping of techniques by difficulty of processing data sets with class imbalance; (6.6) Ranking of noise types by degree of difficulty; (6.7) Groupings of techniques as a consensus of groupings 6.1 to 6.5.

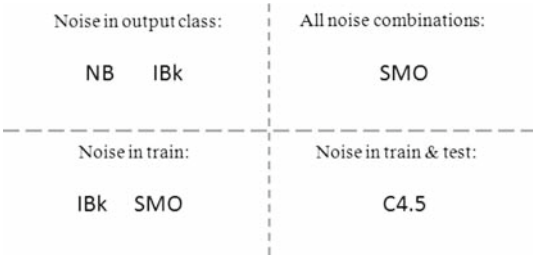

Fig. 8 Grouping of techniques in terms of behavior for increasing noise percentages for different noise types

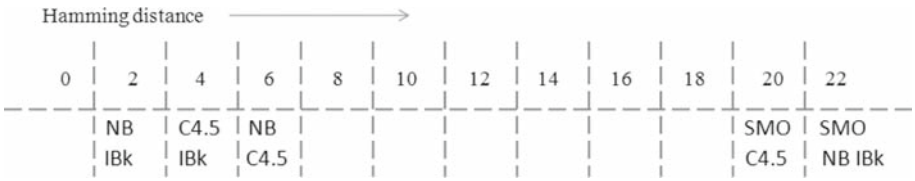

Fig. 9 Grouping of techniques based on hamming distance of similarity of ranking of noise types by difficulty

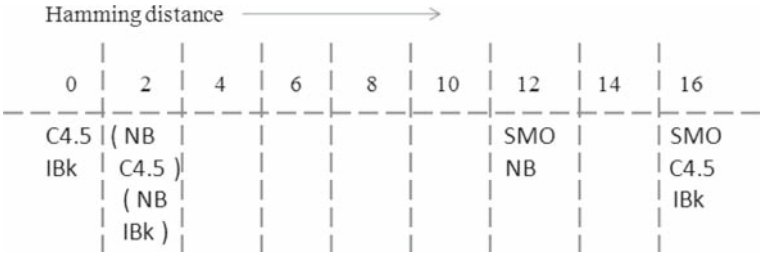

Fig. 10 Grouping of techniques based on hamming distance of similarity of ranking of specific data sets by difficulty

6.1 Grouping of techniques in terms of behavior for increasing noise percentages and types (Table 9, Sect. 5.3)

With reference to Fig. 8, in terms of the behavior for increasing noise percentages, we could define four groups: NB and IBk; IBk and SMO; SMO in its own group, and C4.5 in its own group. Each of these groupings is dependent on a particular combination of noise.

6.2 Grouping of techniques in terms of behavior for type of noise (Ref. Table 11, Sect. 5.3)

With reference to Fig. 9, in terms of the results for type of noise (where the noise is found and its percentage), we could define four groups: NB and IBk; C4.5 and IBk; NB and C4.5 and SMO in its own group. For an explanation of the Hamming distance calculation, refer to Sect. 5.3 and Table 11.

6.3 Grouping of techniques in terms of behavior for specific data sets (Ref. Table 12, Sect. 5.4)

With reference to Fig. 10, in terms of the results for specific data sets, we could define four groups: C4.5 and IBk; NB and C4.5; NB and IBk and SMO in its own group. For an explanation of the Hamming distance calculation, refer to Sect. 5.4 and Table 12.

6.4 Grouping of techniques in terms of geometric rankings for noise percentages and types (Table 9)

In terms of geometric rankings (Table 9), we could say that C4.5 and IBk form a first group for similar behaviors, NB forms a second group on its own and SMO a third group on its own.

6.5 Grouping of techniques by difficulty of processing datasets with class imbalance

For the data sets with the class imbalance problem (thy2c1, thy2c2, thy2c3, bal2c1 and INI), NB was the technique with the best mean geometric precision (66.33), followed by IBk with 59.01, C4.5 with 55.18 and SMO with 26.40. The specific details of the results with data sets for class imbalance were given in Sect. 5.4.1. In terms of technique groupings, NB and SMO had lower results for the 'INI' data set, whereas IBk and C4.5 had better results for 'INI'. SMO also had significantly lower precisions for data sets 'thy2c1', 'thy2c2' and 'thy2c3', with respect to the other techniques. Thus in terms of behavior with the class imbalance data sets, we could group the techniques as: NB with SMO, NB in its own group, SMO in its own group, and C4.5 with IBk.

6.6 Ranking of noise types by degree of difficulty

In this section we rank the noise combinations in decreasing order of difficulty. With reference to Table 14, we observe that the most difficult combinations (considering the first six) were those which contained 50% noise in one of the data sets (especially in the train). A high noise percentage in the output class (50%) was the second most difficult, although the lower noise percentage in the output class (10%) came penultimate.

6.7 Grouping of groupings of techniques (consensus of groupings 6.1 to 6.5)

In order to obtain a final consensus on the groupings of the techniques, we have noted the number of times a given combination occurs using the different criteria of Sects. 6.1 to 6.5.

Table 14 Ranking of noise combinations in decreasing order of difficulty

Rank (1=most difficult)	Noise combination / percentage		Mean accuracy (based on geometric mean) for all techniques
	Train	Test	
1	50	50	46.51
2	50c*	0	48.95
3	0	50	54.4
4	10	50	55.54
5	50	10	60.33
6	50	0	63.22
7	10	10	71.70
8	0	10	73.02
9	10	0	74.51
10	0	0	75.18
11	10c*	0	75.35

[a] c = Noise in output class

Table 15 Grouping of groupings of techniques (derived from Sects. 6.1 to 6.5)

	NB	*C4.5*	*IBk*	*SMO*
NB	1*	2	3	1
C4.5		1	4	0
IBk			0	1
SMO				4

* Number of times this technique was considered as a group on its own

This summary is shown in Table 15. Therefore, from Table 15, based on the highest frequencies, we can see that SMO forms its own group, C4.5 and IBk form a second group, and NB with IBk form a third group (which overlaps with group 2).

7 Conclusions

In this paper, we proposed a systematic study to analyze how attribute noise and class noise affect the quality of the models created by different learning techniques that represent different learning paradigms. Initially, we have defined two groups of techniques, Group 1, comprised of the Naïve Bayes probabilistic classifier and C4.5 tree induction, and Group 2 comprised of the IBk instance-based classifier and the SMO support vector machine. However, the detailed analysis using different evaluation criteria shows a more complex situation, in which the behavior of each technique depends on the type and percentage of noise, class imbalance and the characteristics of the data sets themselves.

The evaluation has a clear utility for situations in which we have to process real-world data which may contain intrinsic or introduced errors. The simple observation is that NB is the best general performer relative to the other three techniques, and SMO gives the poorest relative results. However, in terms of technique groupings based on behavior, it has been seen that this depends on the noise configuration and data set. Different overlapping similarities can be found: NB in a group of its own, SMO in a group of its own, IBk and C4.5 as a group, NB and IBk as a group and NB with C4.5 as a group. The general summary is shown in Table 15.

Returning to the initial hypothesis of groupings which was proposed in the introduction to the article, the superior performance of Group 1 (NB/C4.5), would be mainly due to NB, and the poorer performance of Group 2 (IBk and SMO) would be mainly due to SMO. IBk and C4.5 are on a general parity in terms of performance, and IBk has given superior performance in some cases. With reference to the (theoretical) noise tolerance characteristics of each technique that we described in Sect. 3.4, NB has confirmed that its two strong points (conditional independence assumption and use of conditional probabilities) make it relatively more robust to noisy data. C4.5 has also demonstrated some relatively good results, given its strong points of a heuristic limit for pruning and a confidence measure for creating decision nodes. IBk has shown relatively good results, possibly as a consequence of its using more than one training instance to make a prediction. Finally, SMO has confirmed that its two weak points (reliance on support vectors, and therefore on single instances, and interdependence assumption of attributes) have given it a poorer performance in the presence of noise (Tables 16, 17, 18, 19).

Table 16 Naïve Bayes: geometric test accuracy (in percentage) for all data sets and noise configurations

	Naïve Bayes geometric test accuracy										
	tr0 ts0	*tr0 ts10*	*tr0 ts50*	*tr10 ts0*	*tr10 ts10*	*tr10 ts50*	*tr50 ts0*	*tr50 ts10*	*tr50 ts50*	*tr10c ts0*	*tr50c ts0*
bal2c1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	29.57
bal2c2	94.66	88.22	67.92	95.54	89.10	68.55	92.30	87.54	66.81	90.21	44.08
bal2c3	95.20	85.91	68.08	95.47	85.42	67.85	91.27	83.75	67.25	91.40	49.84
bpa	58.33	58.09	50.17	57.94	59.11	55.29	51.63	50.32	43.78	53.59	29.57
INI	14.24	14.08	4.85	9.39	9.22	9.70	0.00	0.00	0.00	22.34	47.26
iris2c1	92.42	91.85	54.97	88.30	89.52	58.29	88.63	89.01	63.12	82.46	74.54
iris2c2	100.00	95.78	44.21	100.00	97.89	68.69	100.00	97.92	80.70	98.46	90.43
iris2c3	90.70	88.19	36.48	87.88	86.11	42.78	83.57	83.41	52.33	86.42	58.69
pim	71.04	71.43	57.67	70.59	70.36	57.89	65.31	64.24	51.86	72.34	65.75
thy2c1	93.86	89.32	64.90	93.86	87.24	58.24	85.36	79.22	37.21	93.29	90.65
thy2c2	98.50	97.29	68.66	96.97	94.97	71.63	81.26	80.32	56.85	94.47	82.04
thy2c3	88.00	83.66	67.64	83.33	76.58	71.79	75.45	69.28	56.44	89.93	90.69
wbcd	96.67	94.42	74.93	96.36	94.48	81.29	94.35	93.47	84.06	96.67	96.29

Table 17 C4.5: geometric test accuracy (in percentage) for all data sets and noise configurations

	C4.5 geometric test accuracy										
	tr0 ts0	*tr0 ts10*	*tr0 ts50*	*tr10 ts0*	*tr10 ts10*	*tr10 ts50*	*tr50 ts0*	*tr50 ts10*	*tr50 ts50*	*tr10c ts0*	*tr50c ts0*
bal2c1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	27.01
bal2c2	83.42	80.16	67.73	83.89	81.27	68.21	75.91	74.72	63.66	83.09	47.39
bal2c3	83.28	77.96	65.96	82.77	77.68	66.69	77.60	76.56	62.92	84.02	56.66
bpa	57.32	57.68	48.14	60.32	57.26	49.24	11.89	11.58	11.00	55.80	22.61
INI	94.34	90.65	71.38	94.78	92.13	72.48	19.14	10.00	4.70	94.48	52.15
iris2c1	92.87	88.75	70.61	93.18	91.39	67.27	95.55	90.77	69.30	92.87	34.81
iris2c2	98.94	95.29	61.73	98.94	96.32	61.94	96.92	92.04	72.75	98.94	41.89
iris2c3	90.71	90.74	72.37	90.43	87.83	62.49	88.48	87.56	59.16	93.80	56.15
pim	66.83	67.69	57.10	66.99	69.05	59.89	53.49	53.96	52.77	69.31	38.35
thy2c1	93.58	91.95	62.59	90.93	84.87	43.68	64.87	54.67	13.74	87.71	62.71
thy2c2	86.43	85.71	61.73	93.36	86.57	66.51	59.97	59.80	41.51	84.16	42.53
thy2c3	88.58	80.85	65.91	88.16	77.16	57.45	72.75	63.88	50.99	88.94	57.30
wbcd	93.47	90.37	74.99	94.60	93.61	76.81	94.77	93.67	78.48	94.69	51.29

In terms of the noise combinations, we observe that noise in the training data set gave the most difficulty in general, for all the learners.

The observations of the performance of the techniques for specific types of noise, noise combinations, and percentages may serve as a reference for those who wish to select an appropriate technique for their data environment.

Table 18 IBk: geometric test accuracy (in percentage) for all data sets and noise configurations

	IBk geometric test accuracy										
	tr0 ts0	*tr0 ts10*	*tr0 ts50*	*tr10 ts0*	*tr10 ts10*	*tr10 ts50*	*tr50 ts0*	*tr50 ts10*	*tr50 ts50*	*tr10c ts0*	*tr50c ts0*
bal2c1	0.00	0.00	4.87	0.00	0.00	0.00	14.55	4.82	9.36	0.00	43.45
bal2c2	88.54	85.14	65.35	82.48	79.89	64.70	67.22	65.41	58.78	87.94	64.90
bal2c3	89.96	83.10	66.95	82.99	76.41	65.06	66.65	65.60	60.35	89.34	67.13
bpa	58.24	53.84	50.67	56.89	54.20	55.26	48.02	45.01	52.08	57.00	50.15
INI	84.59	84.29	55.54	78.57	75.98	65.01	42.98	35.73	34.32	82.14	65.34
iris2c1	93.81	91.22	65.46	91.02	86.59	63.52	81.67	80.59	58.53	84.68	70.96
iris2c2	100.00	98.43	68.76	100.00	99.49	72.18	96.18	94.86	69.47	94.61	68.23
iris2c3	95.83	92.68	66.36	94.78	92.00	73.18	87.98	82.65	58.31	93.32	68.46
pim	65.87	65.21	52.85	64.86	61.00	56.09	58.07	57.34	52.93	64.33	53.02
thy2c1	87.80	85.16	46.34	88.00	78.28	53.81	79.57	73.99	29.63	82.52	61.97
thy2c2	97.57	96.43	59.78	97.57	95.43	64.93	67.00	69.32	49.54	92.11	68.76
thy2c3	95.40	86.92	57.68	90.10	82.64	54.14	79.36	71.40	58.33	91.19	71.45
wbcd	95.58	94.90	80.48	94.91	94.39	80.47	91.21	90.58	76.09	95.13	67.21

Table 19 SMO: geometric test accuracy (in percentage) for all data sets and noise configurations

	SMO geometric test accuracy										
	tr0 ts0	*tr0 ts10*	*tr0 ts50*	*tr10 ts0*	*tr10 ts10*	*tr10 ts50*	*tr50 ts0*	*tr50 ts10*	*tr50 ts50*	*tr10c ts0*	*tr50c ts0*
bal2c1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	18.83
bal2c2	93.51	88.54	66.66	94.70	88.26	67.61	91.90	87.89	65.43	90.39	24.82
bal2c3	91.61	84.38	68.11	94.91	85.31	67.67	90.74	83.27	66.24	91.74	30.81
bpa	0.00	13.15	6.10	0.00	2.37	11.20	0.00	0.00	0.00	51.62	0.00
INI	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.43
iris2c1	94.95	90.60	56.70	93.90	88.49	59.02	91.78	91.14	62.02	91.11	27.33
iris2c2	100.00	98.43	68.27	100.00	98.94	69.98	100.00	97.92	68.63	100.00	27.64
iris2c3	4.47	4.47	7.75	4.47	4.47	7.75	0.00	0.00	0.00	40.63	3.16
pim	68.64	67.13	52.98	67.61	67.53	51.17	19.59	18.71	12.89	70.20	0.00
thy2c1	85.36	79.50	38.28	85.36	79.50	38.28	31.21	21.21	7.07	73.64	29.06
thy2c2	61.82	59.98	50.03	61.82	57.59	42.28	25.49	17.32	5.77	61.82	13.05
thy2c3	54.86	60.43	46.51	53.17	55.96	49.20	15.63	8.16	4.08	56.16	0.00
wbcd	96.70	95.81	84.22	96.37	95.37	85.59	95.93	95.03	84.65	96.48	28.86

Acknowledgments The authors are grateful to the following persons for their comments and suggestions with respect to the generation of noise and its incorporation into data sets: Dr. Vicenç Torra of The Institute for Investigation in Artificial Intelligence, Bellaterra, Catalunya, Spain, and Dr. Xingquan Zhu of the Department of Computer Science & Engineering, Florida Atlantic University, United States. Thank you to Dr. Mark Hall, previously of the University of Waikato (New Zealand), for his information about IBk. The authors also wish to acknowledge the *Ministerio de Educación y Ciencia* for its support under project TIN2008-06681-C06-05, and *Generalitat de Catalunya* for its support under grant 2009SGR-00183.

References

Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6:37–66

Aha DW (1992) Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. Int J Man–Mach Stud 36:267–287

Angluin D, Laird P (1988) Learning from noisy examples. Mach Learn 2(4):343–370

Asuncion A, Newman DJ (2007) UCI repository of machine learning databases. Available by anonymous ftp to **ics.uci.edu** in the **pub/machine-learning-databases** directory. University of California

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32:675–701

Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11:86–92

Fürnkranz J (1997) Noise-tolerant windowing. In: Proceedings of the 15th international joint conference on artificial intelligence (IJCAI-97), Nagoya, Japan. Morgan Kaufmann, pp 852–857

Goldman SA, Sloan RH (1995) Can PAC learning algorithms tolerate random attribute noise. Algorithmica 14(1):70–84 (Springer, New York)

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten, IH (2009) The WEKA data mining software: an update; SIGKDD Explor 10–18

Hunt EB, Martin J, Stone P (1966) Experiments in induction. Academic Press, New York

John GH, Langley P (1995) Estimating continuous distributions in Bayesian classifiers. In: Eleventh conference on uncertainty in artificial intelligence, San Mateo, pp 338–345

Kearns M (1998) Efficient noise-tolerant learning from statistical queries. J ACM 45(6):983–1006

Meeson S, Blott BH, Killingback ALT (1996) EIT data noise evaluation in the clinical environment. Physiol Meas 17:A33–A38

Nelson R (2005) Overcoming noise in data-acquisition systems (WEBCAST). Test Meas World. http://www.tmworld.com/article/319648-Overcomming_noise_in_data_acquisition_systems.php

Nettleton D, Torra V (2001) A comparison of active set method and genetic algorithm approaches for learning weighting vectors in some aggregation operators. Int J Intel Syst 16(9):1069–1083

Nettleton D, Muñiz J (2001) Processing and representation of meta-data for sleep apnea diagnosis with an artificial intelligence approach. Int J Med Inform 63(1–2):77–89

Platt J (1998) Fast training of support vector Machines using sequential minimal optimization. In: Schölkopf B, Burges CJC, Smola AJ (eds) Advances in kernel methods—support vector learning, Chap 12. MIT Press, pp 169–185

Quinlan JR(1986) Induction of decision trees. Mach Learn 1:81–106 (Kluwer Academic Publishers)

Quinlan JR (1993) C4.5 programs for machine learning. Morgan Kaufmann, San Mateo

Sloan R (1988) Types of noise in data for concept learning. In: Annual workshop on computational learning theory. Proceedings of the first annual workshop on Computational learning theory: 91–96. SIGART: ACM special interest group on artificial intelligence

Sloan RH (1995) Four types of noise in data for PAC learning. Inform Process Lett 54(3):157–162

Torra V (1997) The weighted owa operator. Int J Intell Syst 12(2):153–166

Vapnik VN (1995) The nature of statistical learning theory. Springer Verlag, New York

Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco

Yu S, Zhou ZH, Steinbac M, Hand DJ, Steinberg D (2007) Top 10 algorithms in data mining. Knowl Inform Syst 14(1):1–37

Zhu X, Wu X, Chen Q (2003) Eliminating class noise in large datasets. In: Proceedings of the 20th ICML international conference on machine learning, Washington, DC, pp 920–927

Zhu X, Wu X (2004) Class noise vs. attribute noise: a quantitative study of their impacts. Artif Intel Rev 22:177–210 (Kluwer Academic Publishers)