Data Min Knowl Disc
DOI 10.1007/s10618-012-0299-1

Ensemble-based noise detection: noise ranking and visual performance evaluation

Borut Sluban · Dragan Gamberger · Nada Lavrač

Received: 29 July 2011 / Accepted: 23 November 2012

Abstract Noise filtering is most frequently used in data preprocessing to improve the accuracy of induced classifiers. The focus of this work is different: we aim at detecting noisy instances for improved data understanding, data cleaning and outlier identification. The paper is composed of three parts. The first part presents an ensemble-based noise ranking methodology for explicit noise and outlier identification, named NoiseRank, which was successfully applied to a real-life medical problem as proven in domain expert evaluation. The second part is concerned with quantitative performance evaluation of noise detection algorithms on data with randomly injected noise. A methodology for visual performance evaluation of noise detection algorithms in the precision-recall space, named Viper, is presented and compared to standard evaluation practice. The third part presents the implementation of the NoiseRank and Viper methodologies in a web-based platform for composition and execution of data mining workflows. This implementation allows public accessibility of the developed approaches, repeatability and sharing of the presented experiments as well as the inclusion of web services enabling to incorporate new noise detection algorithms into the proposed noise detection and performance evaluation workflows.

Responsible editor: Eamonn Keogh

B. Sluban (✉)· N. Lavrač
Jožef Stefan Institute and Jožef Stefan International Postgraduate School,
Jamova 39, Ljubljana, Slovenia
e-mail: borut.sluban@ijs.si

N. Lavrač
e-mail: nada.lavrac@ijs.si

D. Gamberger
Rudjer Bošković Institute, Bijenička 54, Zagreb, Croatia
e-mail: dragan.gamberger@irb.hr

Keywords Noise detection · Ensembles · Noise ranking · Precision-recall evaluation

1 Introduction

The quality of real-world data is inevitably subject to errors and other irregularities which are usually referred to as noise. While data measurement errors represent noise which should best be repaired or discarded from the data, other irregularities such as outliers may actually lead to new insights, e.g., in fraud detection and network intrusion detection outlier mining has already proved to be very effective (Aggarwal and Yu 2001).

Both types of irregularities, errors and outliers, appear in the data as class noise (errors in class labels) and/or attribute noise (errors in one or more attribute values). Regardless of the type of noise, noise may have an adverse effect on the quality of information retrieved from the data, models created from the data, and decisions made based on the data. A qualitative study of the impacts of both types of noise on reduced classification accuracy, increased time complexity and size of the constructed classification or representation models was performed by Zhu and Wu (2004).

Standard noise handling approaches used in machine learning aim at improved accuracy of models induced from the data (achieved either by mechanisms trying to avoid overfitting noisy data or with mechanisms used for noise filtering in data preprocessing). On the other hand, this paper addresses explicit noise detection and outlier identification. While noise usually refers to errors in the data, outliers are defined as observations which are numerically distant from the rest of the data, or more formally, as observations that lie outside the overall pattern of a distribution. In this paper the terms noise detection and noise filtering will be used interchangeably, although our aim is not to filter out noise from the data. The paper actually focuses on noise detection and outlier identification, with the aim of improved data understanding.

This work extends our recent research on noise detection and performance evaluation (Sluban et al. 2010, 2011) by proposing an ensemble-based noise ranking methodology for explicit noise detection, an approach for visual performance evaluation of noise detection algorithms and publicly accessible and reusable implementations of the proposed approaches. The paper consists of three parts and the major advances are outlined below.

The first part presents a methodology, named NoiseRank, designed for noise detection by ranking instances according to their potential of being noisy or domain outliers. NoiseRank uses an ensemble of user selected noise detection algorithms to visually present the ranking of potentially noisy instances. NoiseRank was successfully applied to a medical coronary heart disease data analysis problem, where the identified top-ranked instances turned out to be indeed either the records of incorrectly diagnosed patients or medically interesting outlier cases, as proven in domain expert evaluation.

The second part presents a methodology for visual performance evaluation of noise detection algorithms, named Viper. After describing the experimental setting, we first present standard performance evaluation results, followed by the proposed approach which visualizes the performance of noise detection algorithms in the so-called *precision-recall space*. This visualization incorporates two new methods designed for visually presenting the results of noise detection algorithms on domains with artificially injected random noise. The first method highlights the best performing algorithms in the ε-proximity of the most precise noise detection algorithm. The second method presents F-measure values as isolines (contour lines) in the precision-recall space, enabling simple comparison of noise detection algorithm performance.

The third part presents the implementation of NoiseRank and Viper in the ClowdFlows web-based platform for construction and execution of data mining workflows (Kranjc et al. 2012). This implementation allows public accessibility of the developed approaches, repeatability and sharing of the presented experiments as well as the inclusion of web services enabling to incorporate new noise detection algorithms into the proposed noise detection and performance evaluation workflows.

The paper is structured as follows. Section 2 presents the related work. Section 3 presents the NoiseRank methodology for ensemble-based noise detection and ranking and the results of its application on a medical domain evaluated by the domain expert. The Viper methodology for visual performance evaluation on domains with randomly injected class noise is presented and compared to a standard performance evaluation approach in Sect. 4. The implementation of the NoiseRank and Viper methodologies into the ClowdFlows platform is described in Sect. 5. Summary and conclusions are presented in Sect. 6.

2 Related work

Noise handling is an active area of data mining and machine learning. First approaches aimed to develop inductive learning algorithms resistant to noise in the data. In order to avoid overfitting noisy data, the constructed models (decision trees and rule sets) were pruned to get a more general form, thereby reducing the chance of overfitting to noise (Gelfand et al. 1991; Quinlan 1987; Niblett and Bratko 1987; Mingers 1989; Fürnkranz 1997). However, pruning alone cannot avoid the damaging effects of noise and outliers on the structure of the constructed models.

Another common approach to noise handling is to eliminate noise by filtering of noisy instances before model construction; this has the advantage that noisy instances will not adversely affect the structure of the induced model. In the classification noise filtering approach by Brodley and Friedl (1999) multiple learning algorithms were applied to induce classifiers which were then used for noise identification. This was achieved in a cross-validation manner where a data instance was identified as noisy if it was incorrectly classified by one or more classifiers. In the case of multiple classifiers used, a majority or consensus scheme can be used, meaning that an instance is identified as noisy only if it is incorrectly classified by the majority or by all of the learned classifiers, respectively. In this paper the approach is referred to as the *Classification Filter*. It is commonly adopted in various application areas (Verbaeten 2002; Miranda et al. 2009) and/or used as a reference noise filtering approach (Gamberger et al. 2000; Khoshgoftaar et al. 2004).

A substantially different noise detection approach was proposed by Gamberger and Lavrač (1997). This approach, called *Saturation Filter*, is based on the observation that the elimination of noisy examples reduces the so-called *Complexity of the Least Complex correct Hypothesis* (CLCH) value of the training set. The proposed CLCH measure is used to find a saturated training set enabling the induction of a hypothesis which correctly captures the concept represented by the available data; noisy examples are those which are outside the saturated training set.

An approach to classification filtering using different levels of agreement among multiple classifiers, as explored by Khoshgoftaar et al. (2005, 2006) and Verbaeten and Van Assche (2003), is referred to as the *Ensemble Filter*. For large and distributed datasets, the *Partitioning Filter* was proposed by Zhu et al. (2003) which initially splits the dataset into *n* partitions and a classifier is induced for each of them. The *n* classifiers are evaluated on the whole dataset, and finally, voting is used to identify noisy examples. Two modifications of the partitioning scheme were introduced by Khoshgoftaar and Rebours (2004): first, the *Multiple-Partitioning Filter* which combines the predictions of multiple classifiers learned on each partition, and second, the *Iterative-Partitioning Filter* which builds only one model on each subset, but iterates the filtering process until a given stopping criterion is satisfied. Another approach which uses ensemble filters sequentially and passes weighted instances to the next iteration of ensemble filters is the *Boosted Noise Filter* (Zhong et al. 2005).

Numerous other approaches to noise handling can be found in the literature, among them are rule-based approaches (Khoshgoftaar et al. 2004), itemset based approaches (Van Hulse and Khoshgoftaar 2006), distribution-based approaches (Van Hulse et al. 2007) and clustering or distance based approaches (Wiswedel and Berthold 2005; Libralon et al. 2009; Yin et al. 2009). The later two types are common in the related but more specialized field of outlier detection. An extensive overview of outlier detection methodologies is presented by Hodge and Austin (2004).

All the above noise filtering algorithms are mainly used for detecting noise which should be removed from the data to improve its quality. Teng (1999) describes a different approach called *Polishing*. When the noisy instances are identified, instead of removing them, they are repaired by replacing the corrupted values with more appropriate ones, and corrected instances are reintroduced into the data set.

The performance of noise detection and noise filtering algorithms is evaluated very differently throughout the literature and it usually depends on the problem for which noise handling is used. Commonly the performance of noise detection is evaluated indirectly by measuring the change in accuracy of machine learning algorithms in classification or prediction tasks (Brodley and Friedl 1999; Gamberger et al. 2000; Zhu and Wu 2004; Khoshgoftaar et al. 2006; Libralon et al. 2009). In these cases noise detection is considered as a preprocessing step and its performance is evaluated only through its effect on the performance of the main classification/prediction task, e.g., the change in classification accuracy, error rates, model complexity and computational complexity (Verbaeten 2002; Zhu et al. 2003; Khoshgoftaar et al. 2005).

On the other hand, noise detection performance can be evaluated in a direct way when the main task is noise detection itself. Khoshgoftaar and Rebours (2004) compared the sets of instances detected by different noise filtering algorithms and observed their overlaps to determine the significance of the detected noisy instances. Qualitative performance evaluation of noise detection is commonly used in real-life problems, where the domain expert is asked to evaluate the instances identified by noise detection algorithms (Gamberger et al. 1999; Van Hulse et al. 2007), by qualitatively evaluating the significance of each of the detected noisy instances. This type of performance evaluation is usually limited to one or a small number of real-life domains and to the evaluation of one or a comparison of only a few noise detection algorithms.

In contrast, quantitative performance evaluation avoids these limitations, but requires data with known pre-annotated noisy instances. When the noisy instances are known in advance, the performance can be evaluated with standard performance measures used in information retrieval (Manning et al. 2008). *Precision* of noise detection and the amount of retrieved noise or the noise *recall* are the most widely used measures, used separately (Khoshgoftaar et al. 2004; Van Hulse and Khoshgoftaar 2006) or combined with other prediction measures (Verbaeten 2002; Verbaeten and Van Assche 2003; Zhu et al. 2003; Miranda et al. 2009). Some evaluation approaches aggregate the precision and recall results, like the F-measure which presents the trade-off between precision and recall of noise detection (Sluban et al. 2010, 2011), or the precision-recall curves, which are used for the evaluation of ranked retrieval results, i.e., ranked instances obtained by a noise filtering algorithm (Zhong et al. 2005).

3 NoiseRank: Ensemble-based noise detection and ranking

This section presents the motivation and the ensemble-based noise detection and ranking methodology named NoiseRank. After the motivation section, we first present the selected classification and saturation noise filtering approaches as examples of noise detection methods which can be used in NoiseRank. The section concludes by the application of NoiseRank on a real-life medical domain and the qualitative expert evaluation of the detected noisy instances.

3.1 Motivation for explicit noise detection

Standard noise handling approaches used in machine learning aim at improved classification accuracy of models induced from the data. In contrast, the focus of the work presented in this section is on providing a methodology enabling the detection of noisy instances to be inspected by human experts in the phase of data cleaning, data understanding and outlier identification. Therefore, the proposed approach should result in highest possible precision of noise detection. Moreover, it should detect a relatively small set of data instances that represent noise, outliers or data inconsistencies, worth the expert's inspection time and effort. To this end, our goal is not only to develop improved noise detection algorithms but also a new methodology which will support the expert in inspecting the data by detecting, ranking and explicitly presenting noisy instances and/or outliers. A scheme of the proposed methodology is shown in Fig. 1.

The proposed methodology enables expert-guided noise detection. The user can inspect the detected noisy instances and decide whether they are interesting outliers which lead to new insights in domain understanding, erroneous instances which should be removed from the data or false alarms (non-noisy regular instances) and/or instances with minor corrected errors to be reintroduced into the dataset.[1]

[1] The later is referred to as *polishing* by Teng (1999)

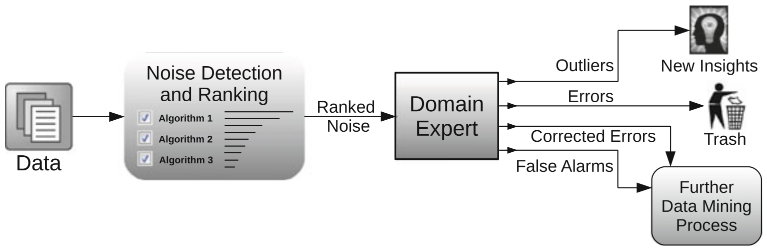


Fig. 1 Scheme of the proposed methodology for expert-guided noise detection

In contrast to other noise ranking approaches where the ranking is produced only by one noise detection algorithm (Van Hulse et al. 2007) or by aggregated predictions of only one machine learning algorithm trained on bootstrap samples of data (Zhong et al. 2005), our approach ranks noisy instances according to the predictions of different noise detection algorithms thus assuring more reliable results. Compared to ensemble filtering by Verbaeten and Van Assche (2003) and Khoshgoftaar et al. (2005, 2006) it enables the user to construct his own ensemble with his own noise detection algorithms. In addition it allows to select the desired filtering level (agreement among algorithms in the ensemble) for automated noise filtering, or to select only the most interesting instances by inspecting the ranked list of noisy instances provided by an interactive visualization interface.

3.2 Selected methods for explicit noise detection

As the aim of NoiseRank is to enable the user to find noise and outliers in the phase of data exploration, methods for explicit noise detection need to be incorporated. As example methods which can be included in NoiseRank, this section presents only the selected classification and saturation-based noise filtering methods which were used when NoiseRank was applied to a medical domain.[2]

3.2.1 Classification-based noise detection methods

The idea behind the noise filtering approach presented by Brodley and Friedl (1999) is to use a classifier as a tool for detecting noisy instances in data. Their method, called the *classification filter*, is performed in a cross-validation manner, e.g., using repeatedly ninefolds for training of a classifier and one complementary fold for class prediction where the incorrectly classified instances are considered to be noisy. The classifiers used in their work were decision trees, nearest neighbor classifiers and linear machines.

In our implementation of classification filters we replaced these simple classifiers by other better performing classifiers. From a variety of available supervised learning algorithms we chose a small and diverse sample of algorithms, each of them

[2] Note that the publicly accessible implementation of the NoiseRank methodology described in Sect. 5.2, enables the developer to easily incorporate his own algorithms into the noise ranking ensemble.

employing a different approach to learning: statistics (Naïve Bayes); information gain, sampling and voting (Random Forest); maximum-margin hyperplane computation in high-dimensional feature spaces (Support Vector Machine) and optimization of computational models (networks) by cost function/error minimization (Neural Network). The reason for choosing diverse classifiers is to ensure good performance of ensemble-based noise detection by NoiseRank.

In summary, the following four different classifiers were used in our work:

Bayes:	Naïve Bayes (used as a baseline classifier),
RF:	Random forest with 500 decision trees (RF500),
SVM:	Support vector machine with a radial basis kernel function,
NN:	Multilayer Perceptron—feedforward artificial neural network.

In the implementation of NoiseRank, described in detail in Sect. 5.2, we used the existing implementations of Bayes, RF and SVM from Orange (Demšar et al. 2004) and the NN from Weka (Hall et al. 2009).

3.2.2 Saturation-based noise detection methods

In order to contribute to the diversity of noise detection algorithms in the ensemble used by NoiseRank, we selected also the algorithms for explicit noise detection exploiting the notion of a *saturated* training set, proposed by Gamberger and Lavrač (1997). A saturated training set is a subset of training examples which enables the induction of a simple hypothesis correctly capturing the regularity/concept represented by the data. A non-saturated training set can be transformed into a saturated one by the elimination of noisy examples. A saturation-based approach to noise filtering, called the *saturation filter*, recognizes noisy examples as those whose elimination enables the reduction of the complexity of the induced hypothesis measured by the Complexity of the Least Complex correct Hypothesis (*CLCH* value) of the reduced training set (Gamberger and Lavrač 1997).

A saturation filter is constructed from two basic methods. The first one is a *saturation test*. At first it computes the complexity of the classification model for the given training set, then it iteratively excludes one training example and computes the complexity of a classification model induced from the rest of the training examples. The examples which have the greatest effect on reducing the complexity of the classification model by their exclusion are labeled as the *most noisy* and are passed on to the second method. The second method, the *noise filter*, randomly chooses one among the most noisy examples and excludes it from the training set, while the other examples are returned to the example set. This is repeated as long as the saturation test finds noisy examples, meaning that a saturated subset has not yet been obtained.

A practical problem of the saturation filter implementation is to find a complexity measure for the induced classification model which corresponds to the *CLCH* value and is sensitive enough to enable the noise filtering algorithm to distinguish between the noisy and the non-noisy instances. While Gamberger and Lavrač (1997) use decision rules for modeling the data, in our reimplementation of saturation noise filtering algorithms used in NoiseRank, an unpruned decision tree learner is used to construct the models and the number of all decision tree nodes is used as the *CLCH* measure of model complexity. Hence, the saturation filtering algorithm has to construct as many decision trees as there are instances in the dataset and this is repeated as many times as there are potentially noisy instances. Consequently, this saturation filter implementation, named *SF*, is very time consuming. Therefore we also tested an implementation, named *PruneSF*, which prunes all the leaves of the decision tree supporting only one instance of the first constructed model, removes these instances, and only then starts the filtering phase. This results in a speed up to the existing saturation filter implementation.

In summary, the following two saturation filters were used in our work:

SF:	Excludes data instances that have the greatest effect in reducing the complexity of a classification model,
PruneSF:	Prunes all the leaves of the first constructed decision tree model supporting only one instance, removes these instances, and only then starts the *SF* phase.

Note that, unlike SF, PruneSF is unable to handle datasets with missing values, which reduces its applicability in real-life domain. Therefore, PruneSF was not used in the application of NoiseRank to a medical domain, described in Sect. 3.4.

3.3 NoiseRank methodology for explicit noise detection

This section presents the NoiseRank methodology for ensemble-based explicit noise detection. The proposed ranking approach, which allows the expert to inspect a ranked list of potentially noisy instances, proves to be a powerful way of presenting the results of noise detection to the expert, as our goal is to focus the expert's attention on the most relevant noisy instances that are worth his inspection time and effort. Reliable noise detection can be obtained by taking into account predictions of a number of different noise detection approaches: the higher the agreement, the higher is the chance that the detected instance is indeed erroneous or atypical and is therefore worth inspecting.

In order to show the expert a set of potentially noisy instances we decided for a visual approach. The proposed method of ranking the detected noisy instances according to the number of elementary noise detection approaches agreeing that an instance is noisy is shown in Fig. 2. The figure illustrates the output of NoiseRank in the real-life medical problem of coronary heart disease (CHD) patient analysis, described in detail in Sect. 3.4. Ranking of the detected instances should help the expert to focus on the instances that are potentially interesting for further analysis, either in terms of erroneousness (noisy instances) or in terms of their atypicality (outliers).

3.4 Application and evaluation of NoiseRank on a medical domain

Practical relevance and applicability of the presented explicit noise detection methodology was verified through expert evaluation of NoiseRank results in a real-life medical domain. For this purpose NoiseRank was applied to a medical dataset and then a domain expert was asked to analyze a small set of top-ranked instances detected

```
Rank |  Class  | ID | Detected by:
-------------------------------------------------------------------
  1.   non-CHD    51  __Bayes______RF______SVM_______NN_______SF___
-------------------------------------------------------------------
  2.   CHD       229  ____RF______SVM_______NN_______SF___
-------------------------------------------------------------------
  3.   CHD         0  ___SVM_______NN_______SF___
  4.   non-CHD    27  ____RF_______NN_______SF___
  5.   non-CHD    39  __Bayes_____SVM_______NN___
  6.   CHD       176  __Bayes_____SVM_______NN___
  7.   CHD       194  __Bayes_____SVM_______NN___
  8.   CHD       213  ____RF______SVM_______NN___
-------------------------------------------------------------------
  9.   CHD        42  ___SVM_______NN___
 10.   non-CHD   120  __Bayes_____SVM___
 11.   non-CHD   164  __Bayes______RF___
 12.   non-CHD   173  ____RF_______SF___
 13.   CHD       196  __Bayes_____SVM___
 14.   non-CHD   226  ____RF_______SF___
-------------------------------------------------------------------
 15.   non-CHD    30  ___SVM___
 16.   CHD        45  ____NN___
 17.   non-CHD    59  ____RF___
 18.   non-CHD    62  __Bayes__
 19.   non-CHD    63  ____SF___
 20.   CHD        67  ___SVM___
 21.   non-CHD    72  ___SVM___
 22.   CHD        97  ____SF___
 23.   non-CHD   135  ___SVM___
 24.   non-CHD   177  ____NN___
 25.   CHD       181  __Bayes__
 26.   non-CHD   189  ___SVM___
 27.   CHD       195  ___SVM___
 28.   CHD       200  ____NN___
 29.   CHD       205  ___SVM___
 30.   CHD       231  __Bayes__
```

Fig. 2 Visual representation of noise detection output. Instances are ranked by the number of noise detection algorithms which identified them as noise. Note that instance IDs are enumerated by zero-based indexing, i.e., starting the count with 0.

as potentially noisy. The coronary heart disease domain is presented in Sect. 3.4.1 and the results of expert evaluation are presented in Sect. 3.4.2.

3.4.1 Coronary heart disease domain

Coronary heart disease (CHD) is a disease of diminished blood supply caused by artherosclerotic plaque in coronary arteries. The diminished oxygen supply of the dependent region of the muscular tissue of the heart results in chest pain (angina pectoris). The extreme consequences are myocardial infarction and death. Coronary heart disease is one of the world's most frequent causes of mortality and an important medical problem (Maron et al. 1998).

In the available dataset, the instances are records of 238 patients who entered a specialized cardiovascular institution[3] in a few months period. The descriptor set includes

[3] Institute for Cardiovascular Prevention and Rehabilitation, Zagreb, Croatia.

10 anamnestic attributes, 7 attributes describing laboratory test results, ECG at rest (5 attributes), the exercise test data (5 attributes), echocardiography results (2 attributes), vectorcardiography results (2 attributes), and long term continuous ECG recording data (3 attributes). The patients were classified into two classes: CHD patients diagnosed as those with coronary heart disease and non-CHD patients. The included negative cases (patients who do not have CHD) are not randomly selected people but individuals with some subjective problems or those considered by the general practitioners as potential CHD patients. As the data was collected at a specialized medical clinic, the database is not an epidemiological CHD database reflecting actual CHD occurrence in a general population, since nearly one half of gathered patient records represent actual CHD patients. More specifically, the dataset consists of 238 patient records: 111 CHD patients (positive cases), and 127 people without CHD (negative cases). Among them there are 177 males (80 positive and 97 negative) and 61 females (31 positive and 30 negative).

3.4.2 *Expert evaluation of results obtained by* NoiseRank

The medical expert (cardiologist) received in total 8 top-ranked examples, shown in Fig. 2, for the analysis. In the qualitative evaluation he used all the available retrospective data for these patients to repeat the previous medical reasoning process with the intention to identify why they were detected by NoiseRank as special cases. Since exercise and Holter ST depression values are medically most relevant CHD markers for expert evaluation (Deanfield et al. 1984),[4] Table 1 shows the measured data for exercise and Holter ST depression values for the 8 top-ranked patients that were analyzed by the cardiologist. We refer to this table in the analysis of some of the top-ranked instances below.

Table 1 Comparison of the exercise (*exST*) and Holter (*holterST*) ST depression values (in millimeters) for the top 8 noisy instances and for the average *CHD* and *non-CHD* instances (patients) in the CHD domain

Rank	Diagnosis	Patient ID	exST	holterST
1	Non-CHD	51	0.80	1.00
2	CHD	229	0.30	0.30
3	CHD	0	1.10	0.50
4	Non-CHD	27	0.50	1.00
5	Non-CHD	39	0.40	0.50
6	CHD	176	0.80	1.00
7	CHD	194	0.10	1.00
8	CHD	213	0.20	0.20
		Average non-CHD	0.15	0.21
		Average CHD	1.11	1.40

[4] This fact has been also previously confirmed by machine learning approaches on the same CHD domain (Gamberger and Lavrač 2002).

As a result of the analysis the expert identified the following reasons why the top-ranked examples are indeed special cases (noise or outliers):

- **Classification mistake (patient ID 51):** After rechecking the patient data the cardiologist realized that a mistake in the classification has occurred. In the concrete situation the mistake was the result of a formal patient classification that did not take into account the actual conditions in which the measurements were performed. The situation is illustrated in the first row of Table 1. Patient 51 is classified as non-CHD because exercise ECG ST depression value 0.8 mm is below the cut-off value of 1 mm for myocardial ischemia and despite the fact that Holter ECG ST depression value of 1 mm is actually a borderline value. The mistake was done because it was not realized that exercise ECG ST value 0.8 was measured at only very moderate level of actual exercise stress. It can be assumed that with a normal level of exercise or effort the expected depression value would have been significantly over 1 mm. Given that this is an actual classification mistake, this type of noise was easily detected by all the noise detection algorithms.
- **Sportsman (patient ID 0):** The patient is correctly classified as CHD but the critical value 1.1 mm for exercise ST depression (second row of Table 1) was detected under the conditions of relatively high exercise stress. This is due to the excellent physical condition of the patient, having as the consequence that other patient data, including Holter ST depression, have almost normal values. The patient represents an actual outlier and its detection as a noisy example is fully justified.
- **False positive medical test (patient IDs 27, 194):** It is known that Holter ECG ST depression values (presented in the last column of Table 1) are sometimes prone to false positive results. It means that values $\geq$1 mm do not necessarily mean a positive diagnosis (CHD). The situation is relatively more common for female patients. In the cases when the Holter ST depression value is $\geq$1 mm and at the same time the more reliable test exercise ST depression is $<$1 mm, the medical expert may decide to classify the patient as negative (in the concrete data set for patient ID 27 presented in the fourth row of Table 1) or he may decide to classify the patient as positive (patient ID 194 presented in row 7 of Table 1). It is very interesting to notice that both situations were successfully recognized by most of the noise detection algorithms. This was possible because Holter ST depression is one of the two main decision attributes used for patient classification.
- **Other medical problems (patient ID 39):** The patient is correctly classified as non-CHD which is strongly suggested by her exercise and Holter ST depression values (row 5 of Table 1). However, this instance was detected as noisy because of significantly non-normal values of other diagnostic parameters like high cholesterol value, low high-density liphoprotein value, and the detected ST depression in the baseline ECG test at rest. Technically, this patient is an example of distributed attribute noise and medically it represents a patient with a higher level of cardiovascular risk factors.

- **Borderline case (patient ID 176):** The patient has the main decision parameter exercise ECG ST depression value (0.8 mm) just below the cut-off point of 1.0 mm. Formally patient 176 (row 6 in Table 1) is correctly classified as CHD because of high Holter ECG ST depression value (1.0 mm). However, its detection as a noisy example is not surprising. It is interesting to notice that although the cardiologists will also in the future classify similar cases in the same way, the result demonstrates that from the technical point of view the classification of patients like the patient with ID 176 into the opposite class would be more appropriate. The result illustrates the complexity of the medical decision making process.
- **Complex medical case (patient IDs 229, 213):** Although in the majority of examples exercise and Holter ST depression values are used for the diagnosis of CHD patients, there exist situations when the disease is diagnosed from the generally very severe heart disease conditions (severe heart failure) detected by echocardiography examination. It means that patients with IDs 229 and 213, presented in rows number 2 and 8 of Table 1, respectively, are correctly classified but they are detected as noisy because their ST depression values correspond to the description of healthy subjects.

The results of medical evaluation of the detected noisy examples show that their case-by-case analysis was worth additional human time and effort. The most important is the example (patient with ID 51) for which the domain expert realized that the original patient classification was inappropriate. This example is important because it demonstrates how explicit noise detection can help to increase the quality and reliability of the medical decision making process. Expert analysis of other cases actually confirmed the existence of outliers in medical decision making. The identified reasons for outlier existence proved useful for increasing expert's awareness of such cases. In summary, the proposed analysis process proved relevant for data driven elicitation of implicit expert knowledge.

4 Viper: Visual performance evaluation

This section presents the methodology for visual performance evaluation of noise detection algorithms named Viper. We first present the motivation for developing a new approach to visual performance evaluation in Sect. 4.1, followed by the presentation of a simple experimental setting designed to show how the selected noise detection algorithms can be evaluated in Sect. 4.2. This setting is used to illustrate the standard performance evaluation approach in Sect. 4.3, contrasted with the Viper visual performance evaluation approach in Sect. 4.4.

4.1 Motivation for visual performance evaluation

Developing a new noise detection algorithm presents a substantial amount of developer's work and the noise detection performance it achieves is the main indicator of its successful future application. Therefore, easily understandable and comprehensive presentation of performance results can significantly support the developer in the evaluation and comparison of the results.

Evaluating the performance of noise detection algorithms requires either qualitative examination by domain experts, as presented in Sect. 3, or quantitative analysis of noise detection results on data with known or artificially injected noisy instances. The goal of this work is to improve and develop methods for quantitative performance evaluation, which is motivated by the following observations.

- First, in the literature the performance of noise handling approaches has been usually addressed in the context of classification problems and evaluated by its effect on classification performance (Brodley and Friedl 1999; Gamberger et al. 2000; Zhu and Wu 2004; Khoshgoftaar et al. 2006; Libralon et al. 2009). However, classification performance measures are not adequate measures for evaluating the success of explicit noise detection.
- Second, the actual performance of noise detection can be quantitatively evaluated on domains with annotated or injected noise by the *precision* of noise detection and the *recall* of noisy instances. These two measures known from information retrieval are widely used for noise detection evaluation in addition to other prediction measures (Verbaeten 2002; Verbaeten and Van Assche 2003; Zhu et al. 2003; Miranda et al. 2009) or just as two separate detection measures (Khoshgoftaar et al. 2004; Van Hulse and Khoshgoftaar 2006). Their joint or aggregated presentation, on the other hand, has not been reported in the noise handling literature, except for precision-recall curves used for evaluating ranked noise retrieval results by Zhong et al. (2005) and the F-measure used by Sluban et al. (2010, 2011).
- Third, performance results of noise detection are typically presented in tabular format, which proves to be difficult for presenting multiple performance measures at once as well as to compare the performance results of several algorithms. Graphically presenting the results by a bar chart can ease the comparison, but depicting more than one performance measure reduces the chart's understandability.

These observations motivated us to develop the VIPER performance evaluation methodology, which (1) addresses noise detection performance directly by measuring the precision and recall of noise detection on data with known or injected noisy instances, (2) integrates two new evaluation methods that trade off precision and recall: *F-isolines* and the *ε-proximity* evaluation methods, and (3) jointly visualizes these evaluation methods in the two-dimensional *precision-recall space*. A simplified example of the VIPER visual performance evaluation approach is presented in Fig. 3.

In contrast to Zhong et al. (2005) who use precision-recall curves for the evaluation of ranked instances retrieved by their noise filtering algorithm, the VIPER methodology aims at the evaluation and comparison of noise detection performance in terms of precision, recall and the F-measure results achieved by different noise detection algorithms. The VIPER methodology presents a novel kind of visual performance evaluation in the precision-recall space, which has—to the best of our knowledge—not been proposed before in the noise handling literature. By employing two new evaluation methods, VIPER enables intuitive evaluation, interpretation and comparison of noise detection performance in terms of the three performance measures at the same time. This visual performance evaluation methodology can also be used for evaluating information retrieval and entity recognition algorithms, as well as for any other algo-

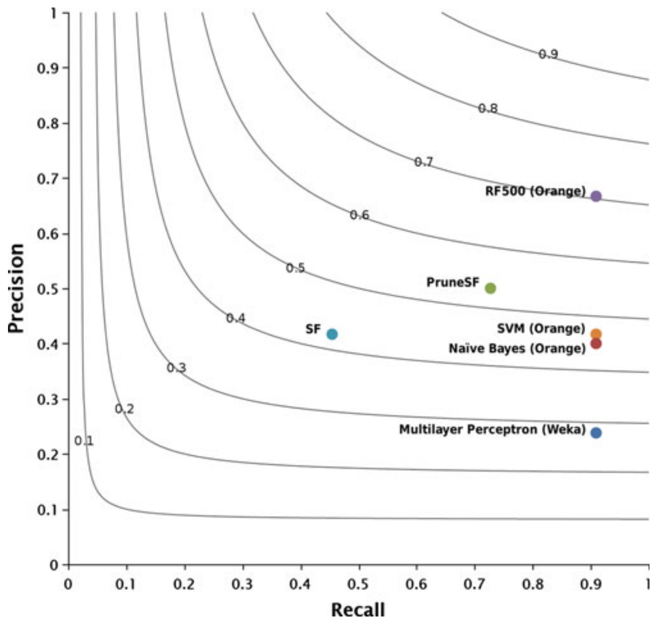


Fig. 3 Performance results of different noise detection algorithms in the precision-recall space on the CHD dataset with 5 % injected class noise, clearly showing that the RF500 algorithm outperforms other algorithms in terms of precision (Y-axis) and the F-measure (F-isolines are presented as curves in the precision-recall space)

rithms for which the evaluation in terms of precision, recall and the F-measure is most suitable.

4.2 Experimental setting

For evaluating the performance of different noise detection approaches we chose two noiseless and two real-world datasets, specified in Table 2. In all the datasets we injected three different levels of class noise: 2, 5 and 10 %, meaning that the particular percentage of instances of a dataset was randomly selected and the class values were replaced.[5]

We intentionally chose two artificial noiseless datasets Tic Tac Toe (TTT) and King-Rook versus King-Pawn (KRKP) from the UCI machine learning repository (Frank and Asuncion 2010), as a starting point to evaluate the noise filtering algorithms' ability to detect injected noise. The other two datasets are real-world datasets, containing their own noisy instances: a medical coronary heart disease (CHD) dataset,[6] described in Sect. 3.4.1 and used in our previous research (Gamberger et al. 2003), and a dataset of

[5] Alternatively, attribute noise could have been inserted, but we were motivated by medical domains where false diagnostics is the main concern.

[6] The original CHD dataset includes also 19 instances with missing values, which were removed in the experiments described in this section in order to be able to apply two different saturation filters.

Table 2 Datasets used in the quantitative evaluation of noise detection algorithms

Dataset	Instances	Attributes	Type
TTT	956	9	Noiseless
KRKP	3,196	36	Noiseless
CHD	219	37	Real-world
NAKE	464	500	Real-world

TTT Tic Tac Toe, *KRKP* King-Rook versus King-Pawn, *CHD* Coronary Heart Disease, *NAKE* News Articles on Kenyan Elections

News Articles on Kenyan Elections (NAKE) described in Pollak (2009) and Pollak et al. (2011). All the datasets are two class domains.

For each of the four datasets and for each noise level we made ten datasets with randomly injected noise. This was done to get more reliable noise filtering results, meaning that each result presented in this work for a specific domain and a specific noise level is the average value of the results obtained from ten separate experiments, each on a dataset with different randomly injected noise. In this way each noise filtering algorithm was tested not only on four different datasets but actually on 120 datasets, allowing for massive testing of noise detection algorithms despite the relatively small number of selected domains.

We evaluated the performance of elementary noise detection algorithms as well as of their ensembles. We selected the same set of elementary noise detection algorithms as in Sect. 3.2: the classification noise filters (Bayes, RF, SVM and NN) and the saturation noise filters (SF and PruneSF). A number of their ensembles was constructed and evaluated as well. To be more convinced that an instance is noisy, ensemble noise filtering approaches take into account predictions of several different noise filtering algorithms when deciding if an instance should actually be declared as noisy. To increase the diversity among the voters in the ensemble we constructed our ensemble filters from a combination of classification and saturation filters, forming different groups of filtering algorithms. First, we constructed three different ensembles consisting of classification filters:

Ens1: Bayes, RF, SVM,
Ens2: RF, SVM, NN,
Ens3: Bayes, RF, SVM, NN.

RF and SVM were used in all three ensembles, since they are generally accepted as good performing classification algorithms. Bayes and NN, on the other hand, were not used in Ens1 and Ens2, respectively, to see how less favorably performing members of the ensemble affect the ensemble's performance. To observe the influence of saturation filters in the ensembles we constructed ensembles **Ens4**, **Ens5** and **Ens6** by adding SF to Ens1, Ens2 and Ens3, respectively. Like SF, also PruneSF was used to construct ensembles **Ens7**, **Ens8** and **Ens9** from Ens1, Ens2 and Ens3, respectively. The last ensemble **Ens10** is formed from the group of all elementary noise filters explored in this work: Bayes, RF, SVM, NN, SF and PruneSF. All ten ensemble noise filters were used in the consensus voting scheme, denoting an instance as noisy only if all members in the ensemble identified it as noisy.

In addition, to illustrate how a novel noise detection algorithm can be evaluated both in the standard setting and in the VIPER visual performance evaluation setting, we included in the experimental evaluation also our High Agreement Random Forest noise detection algorithm—HARF (Sluban et al. 2010). To this end, two variants of HARF were used (HARF-70 and HARF-80) which proved to perform best in our evaluations. More details on HARF and its performance are presented in Appendix 1.

4.3 Standard performance evaluation

Quantitative performance evaluation can only be performed on the data for which the evaluator knows which instances are noisy and which are not. This can be performed on data with artificially inserted noise or on data with known noisy instances. In this section we elaborate this setting and illustrate how noise detection algorithms would be evaluated in a standard way. Unlike the common evaluation of noise detection performance through its influence on classification accuracy, a more adequate approach is the one which is standardly used in information retrieval: using precision and recall of correctly detected noisy instances. This section presents the basic evaluation measures, followed by the standard presentation and evaluation of noise detection results.

4.3.1 Basic performance measures

We adopted standard performance measures used in information retrieval (Manning et al. 2008). A basic measure to be used in the quantitative evaluation of noise detection methods is *precision*, defined as true positives divided by all the predicted positives (i.e., the percentage of correctly detected noisy instances among all the instances identified as noisy by the noise detection algorithm).

$$\text{Precision} = \frac{\text{number of true noisy instances detected}}{\text{number of all instances identified as noisy}}$$

Another useful measure is *recall*, which is defined as true positives divided by all the positives (i.e., the percentage of correctly detected noisy instances among all the noisy instances inserted into the dataset as random noise).

$$\text{Recall} = \frac{\text{number of true noisy instances detected}}{\text{number of all noisy instances in the dataset}}$$

To model a desired precision-recall trade-off, the so-called F-measure combining precision and recall is used. The general formula for computing the F-measure is the following:

$$F_\beta = (1 + \beta^2)\frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}} \qquad (1)$$

where for $\beta = 1$ we get the standard F-measure, also referred to as the F_1 score. By setting the β parameter the user can assign more importance to either precision or recall in the computation of the F-score.

4.3.2 Experimental results

According to the experimental settings described in Sect. 4.2 we evaluated elementary noise detection algorithms, their ensembles and our recently developed HARF noise detection algorithm, using the performance measures described in the previous section. Since precise noise detection is desired by the experts who get to evaluate and identify the output of noise detection algorithms, we chose to attach more importance to precision than to recall in the computation of the F-measure. Therefore we used the $F_{0.5}$-score performance measure for our quantitative evaluation of noise detection algorithms (obtained by using $\beta = 0.5$ in Eq. 1). The experiments were performed on four datasets with three different levels of randomly injected class noise, each repeated ten times.

We present the average precision (Pr) of noise detection and $F_{0.5}$-score experimental results along with their standard deviations in Tables 3 and 4. Due to their size, these large tables are presented in Appendix 2. Table 3 presents performance results of the elementary and the HARF noise detection algorithms, whereas in Table 4 the performance results of the ensemble filters can be found. The best precision and the best $F_{0.5}$-score results in each table are printed in bold.

Once the tabular results have been presented, standard evaluation practice requires elaborate interpretation of these results. Such an extensive discussion, needed to evaluate and compare the results presented in Tables 3 and 4, is too lengthy and is therefore above the scope of this section. For completeness, however, it is made available in Appendix 2, in order to illustrate the need for such a lengthy discussion of tabular results.

In addition to the tabular representation, the usual approach to compare the performance results of different noise detection algorithms is the results visualization by bar charts. We compare the precision of noise detection results, noise recall results and the $F_{0.5}$-scores of different noise detection algorithms on four domains with 5 % artificially injected noise in Figs. 4, 5 and 6, respectively. The results obtained with 2 and 10 % injected noise level are very similar and therefore their presentation is skipped.

Figure 4 shows that ensemble filters and our HARF noise detection algorithm outperform the elementary filters in terms of high precision, where Ens10 and HARF-80 are the most precise algorithms on the TTT and KRKP domains and HARF-80 is the precisest noise detection algorithm on the CHD and NAKE domains. The comparison of noise recall illustrated in Fig. 5 shows that classification filters undoubtedly perform best, followed by the ensembles Ens1, Ens2 and Ens3, and HARF-70. In terms of $F_{0.5}$-scores shown in Fig. 6 the ensembles and HARF algorithms are also significantly better compared to elementary noise detection algorithms, best results achieved by HARF-80 on all the domains, except on the TTT domain where Ens7 performs best. However, all these results are shown in separate figures for precision, recall and the $F_{0.5}$-score, which makes them hard to interpret as a whole.

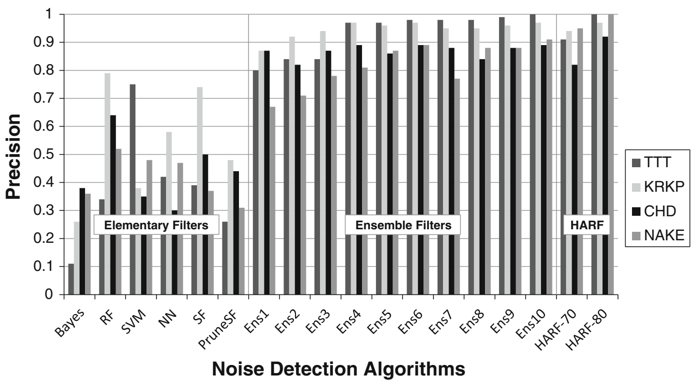


Fig. 4 Comparison of precision of noise detection results of different noise detection algorithms on four domains with 5 % injected noise

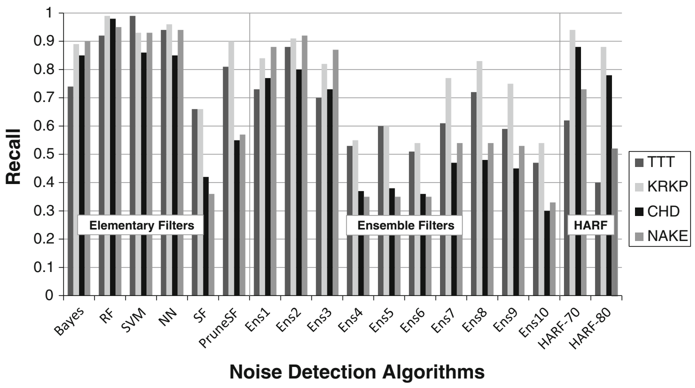


Fig. 5 Comparison of noise recall results achieved by different noise detection algorithms on four domains with 5 % injected noise

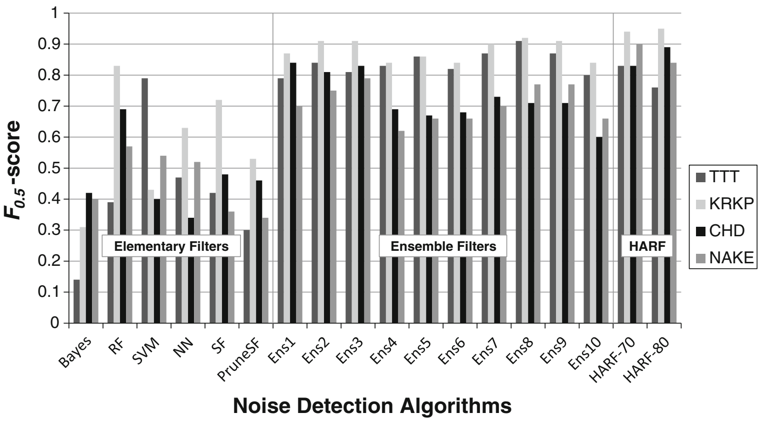


Fig. 6 Comparison of $F_{0.5}$-scores achieved by different noise detection algorithms on four domains with 5 % injected noise

4.4 Improved performance evaluation with VIPER

After the standard performance evaluation of different noise detection approaches described in the previous section, we now present the evaluation of experimental results by our VIPER methodology for visual performance evaluation. In contrast to tabular and simple graphical presentation of results in standard evaluation practice, our methodology visualizes performance results in the precision-recall space enabling intuitive interpretation and simultaneous comparison of three performance measures. In this section we describe two new evaluation methods, followed by the VIPER visualization methodology and finally we present the evaluation of noise detection algorithms with the proposed VIPER methodology for visual performance evaluation.

4.4.1 New evaluation methods

In this section we present two evaluation methods which enable the comparison of noise detection performance of different algorithms by modeling their precision-recall trade-off.

- **ε-proximity evaluation method:** The designed measure for best noise detection algorithm selection does not insist on maximal precision (Pr) of noise detection at the cost of low recall (Re). Instead, it trades off high precision by maximizing the recall of noise detection in the ε-neighborhood of the most precise noise detection algorithms, where ε is a (small) tolerated proximity of the maximal possible precision achieved by different noise detection algorithms $A_j \in \mathcal{A}$, where $\mathcal{A}$ is the set of all used noise detection algorithms.

$$\begin{aligned} BestNoiseFilter(\mathcal{A}) &= \arg\left(\max_{A_i \in \mathcal{A}\ ;\ Cond} [Re(A_i)]\right) \\ Cond\ :\ Pr(A_i) &> \max_{A_j \in \mathcal{A}} [Pr(A_j)] - \varepsilon\ . \end{aligned} \tag{2}$$

- ***F*-isoline evaluation method:** To compare the interaction of different performance measures of noise detection all at once, this method proposes to depict the F-measure values as isolines (contour lines) in the precision-recall space. This allows for intuitive visual interpretation of algorithm performance and its comparison to other noise detection algorithms not only in terms of precision and recall but at the same time also in terms of the F-score. Additionally, in this way equivalence classes of noise detection algorithms that achieve a similar F-score can be made and thus groups of algorithms with similar noise detection capabilities can be identified.

4.4.2 VIPER *visualization methodology*

These two evaluation methods are jointly used in a novel VIsual PERformance (VIPER) evaluation approach. In this approach, noise detection performance results are presented in the precision-recall space, a two-dimensional space where one dimension depicts recall values and the other precision values, in the following way:

- The noise detection performance result of a noise detection algorithm on a specific domain is presented as a point in the precision-recall space.
- According to the *F-isoline evaluation method*, equal F-scores are depicted in the form of isolines, which enables the comparison of F-measure results of noise detection algorithms in the precision-recall space.
- The ε-proximity of the maximal achieved precision of noise detection is emphasized to enable easy identification of the algorithm with maximal recall as the best performing algorithm in the ε-proximity of the algorithm achieving maximal precision of noise detection.

The Viper visualization enables an intuitive interpretation and comparison of performance results of different noise detection algorithms on a selected domain. By integrating the *F-isoline evaluation method* into the precision-recall space three basic performance evaluation measures (presented in Sect. 4.3.1) can be observed at the same time in a two dimensional space. Our *ε-proximity evaluation method* formally described with Eq. 2 becomes also easily understandable with its visualization in the precision-recall space. An example of the proposed visualization was already presented in the motivation Sect. 4.1 in Fig. 3, and the Viper evaluation of all noise detection algorithms examined in this work follows in the next section.

4.4.3 *Experimental results visualized with* Viper

In the standard evaluation and comparison of noise detection performance results, as presented in the Sect. 4.3, inspection of tabular results is needed, which can require quite some effort to compare the results, especially in the case of several performance measures and algorithms used. Furthermore, graphical presentation with bar charts requires a separate figure for each performance measure, although theoretically more than one measure could be depicted in a bar chart, but it would reduce its understandability. On the other hand, the Viper methodology for visual evaluation and comparison of noise detection performance results, enables the visualization of several performance measures and results for several noise detection algorithms in a single figure.

Viper presents the performance results of noise detection algorithms on a specific domain as points in the precision-recall space. In the experimental evaluation of our noise detection algorithms, we used the $F_{0.5}$-score which weights precision more than recall. This score is used in the F-isoline evaluation method, where equal $F_{0.5}$-scores in the form of isolines enable to compare noise detection performance in the precision-recall space. When selecting the algorithm with maximal recall in the ε-proximity of maximal achieved precision, as defined in Eq. 2, we chose the proximity value $\varepsilon = 0.05$. Using these settings we compared the performance of elementary and ensemble noise detection algorithms with the HARF noise detection algorithm. The performance results of noise detection on four domains with 5 % injected class noise and visualized with the proposed Viper methodology can be found in Figs. 7, 8, 9 and 10.[7]

[7] Similar figures are available upon request for 2 and 10 % noise levels.

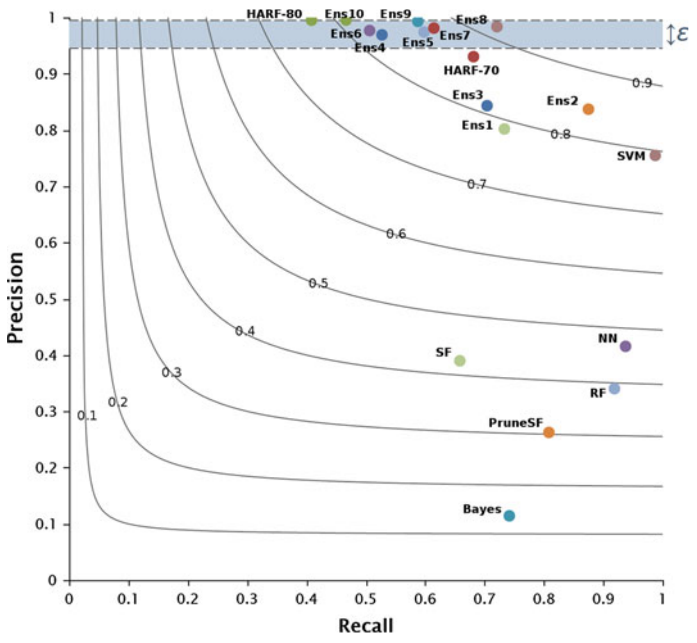


Fig. 7 Performance results of different noise detection algorithms in the precision-recall space on the Tic Tac Toe dataset with 5 % injected noise

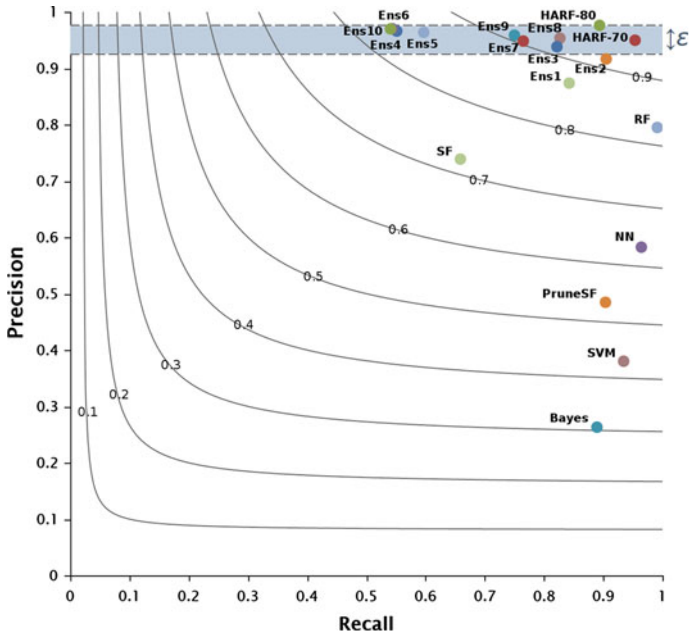


Fig. 8 Performance results of different noise detection algorithms in the precision-recall space on the King-Rook versus King-Pawn dataset with 5 % injected noise

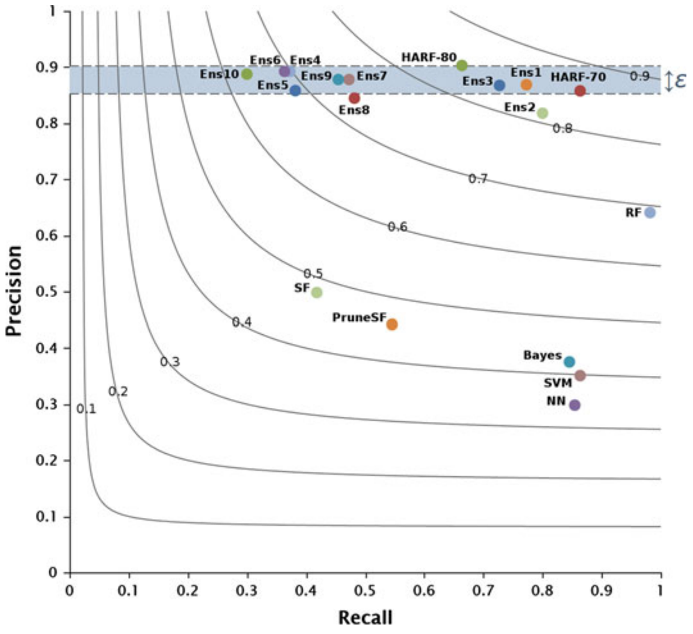


Fig. 9 Performance results of different noise detection algorithms in the precision-recall space on the Coronary Heart Disease dataset with 5 % injected noise

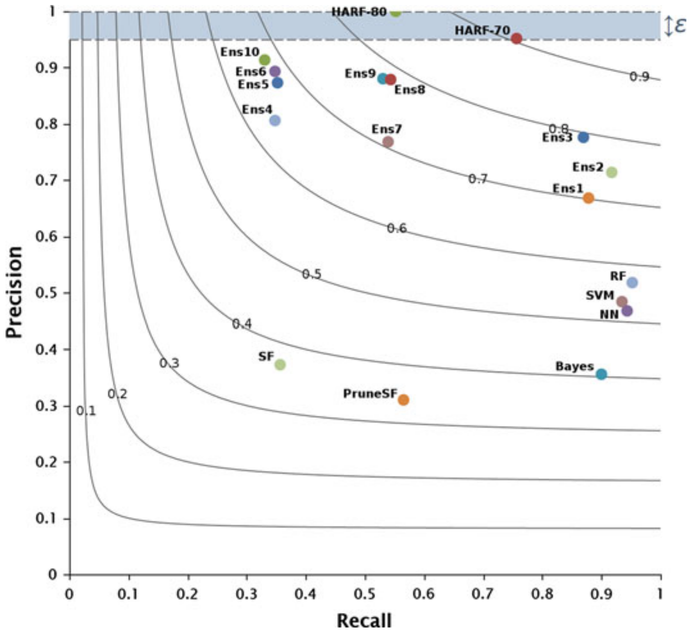


Fig. 10 Performance results of different noise detection algorithms in the precision-recall space on the News Articles on Kenyan Elections dataset with 5 % injected noise

The comparison and interpretation of results by the VIPER performance visualization is very intuitive and almost straightforward. The advantage of noise detection algorithm performance evaluation by VIPER visualization in the precision-recall space provides the expert with the means to compare and see the interaction of three performance measures: precision, recall and the F-measure. Furthermore, also our ε-proximity evaluation method for selecting the best noise detection algorithm becomes very clear through its visualization of the ε neighborhood of maximal achieved precision.

By inspecting the figures we can observe the following facts. Elementary classification filters are fairly good at finding most of the injected noise based on their high recall results, however they identified as noisy also a lot of regular (non-noisy) instances and therefore their precision is not as satisfactory nor are their $F_{0.5}$-scores. In comparison, the saturation filters have lower recall, while the precision and the $F_{0.5}$-score results are between the best and the worst classification filter performance.

Ensemble filters formed from groups of elementary filters, on the other hand, prove to be better suited for precise noise detection. Especially the Ens10 filter which has the highest variety of different filtering algorithms achieves the highest precision on three out of four domains among the ensemble filters, but of course at the expense of lower recall and hence also of lower $F_{0.5}$-score. Generally ensemble filters show great increase in precision of noise detection, however their $F_{0.5}$-scores are (because of their lower recall) in the majority of cases comparable to the performance of best classification filters.

Furthermore, since classification filters achieve good noise recall, also Ens1, Ens2 and Ens3, which are constructed from different groups of classification filters, achieve best recall results among ensembles filters. It is interesting to notice that including a poorly performing noise detection algorithms in the ensemble[8] does reduce the ensemble's recall, but it does not negatively affect its precision, on the contrary it may even increase it, which may lead also to higher F-scores. This can be observed in all three ensemble groups: Ens1–3, Ens4–6 (with SF) and Ens7–9 (with PruneSF), which are all constructed from the same three basic groups of classification filters, i.e., Ens1, Ens2 and Ens3. Another interesting observation, very easy to see with the VIPER visualization, is that the three ensemble groups form cluster-like formations in the precision-recall space. By adding a saturation filter to the Ens1–3 group and obtaining the Ens4–6 or Ens7–9 group, the performance change seems to be almost a translation of the group's cluster formation in the precision-recall space.

The addition of a new noise detection algorithm, like the saturation filter, to an ensemble with a consensus voting scheme translates the ensemble's performance in the direction of the lowest recall of a member in the ensemble and elevates its precision. These observations may suggest that the best approach to constructing an ensemble with a good precision-recall performance could be by using a diverse group of noise detection algorithms, where each member is achieving highest possible recall.

After showing the VIPER evaluation of few basic elementary filters and some of their ensembles, we depict the VIPER evaluation of a new noise detection algorithm

[8] Bayes added to Ens2 on domains TTT, KRKP and NAKE to get Ens3, or NN added to Ens1 on the CHD domain to get Ens3.

and its comparison to other (existing) approaches by including our High Agreement Random Forest noise detection algorithms (HARF) with 70 and 80 % agreement level (HARF-70 and HARF-80, respectively) into the evaluation. Without in-depth analysis like with tabular results presentation, the performance of the two HARF algorithms is immediately apparent from the visualization. They achieve comparable or higher precision and recall of noise detection compared to ensemble filters and clearly outperform all elementary noise detection algorithms. While HARF-80 is the most precise noise detection algorithms in all four domains, in terms of the $F_{0.5}$-score results HARF-70 performs best on all domains except the TTT, where the ensembles Ens7–9 perform better.

Finally, the selection of the best noise detection algorithm according to our ε-proximity evaluation method, defined by Eq. 2, depends on the tolerated ε drop of precision. In the case of $\varepsilon = 0.05$ on the TTT domain ensemble filter Ens8 performs best, but on the other three domains our HARF-70 noise detection algorithms perform best.[9]

5 Implementation and public availability

This section presents our implementations of NoiseRank and Viper, enabling their public accessibility, repeatability of the experiments and the obtained results, as well as the integration of user specific noise detection algorithms available as web services. First, we describe a web-based platform for composition and execution of data mining workflows in which we have integrated all the components required for ensemble-based noise ranking and visual performance evaluation of noise detection. Second, we present the implementation of the proposed NoiseRank methodology, and finally, the implementation of the Viper performance evaluation methodology.

5.1 Embedding environment

We implemented the NoiseRank and Viper approaches in the ClowdFlows web-based platform for composition and execution of data mining workflows (Kranjc et al. 2012). In contrast to other data mining environments (Weka, Orange, etc.) the ClowdFlows environment does not require any installation, since all the processing load is taken from the users machine to remote servers, and therefore it may entirely be run in a modern web browser. This enables public accessibility of our approaches, repeatability and sharing of the presented experiments and the obtained results, as well as the inclusion of web services allowing the application of new noise detection algorithms.

We extended the functionality of the ClowdFlows platform by implementing the building blocks, called widgets,[10] required for the construction of workflows for

[9] An increase or decrease of the ε value for few percents, does not dramatically change the selection of the best noise detection algorithm.

[10] Widgets are processing units used as components of a workflow, which—given some input data and/or parameters—perform a certain computation or visualization task.

explicit noise detection with NoiseRank and the construction of workflows for visual performance evaluation of noise detection with Viper. Our implementation provides compatibility with algorithms from two known data mining platforms, Weka (Hall et al. 2009) and Orange (Demšar et al. 2004). We implemented widgets and web services covering the following functionalities:

– Loading datasets (File to Orange Data Table (ODT), UCI dataset to ODT, 26 standard UCI datasets included),
– Data format conversion (ODT to String or ARFF Weka Instances, ODT to TAB file, CSV file or ARFF file)
– Adding class noise (widget),
– Noise filtering (Classification filter widget and Saturation filter widget),
– Classification algorithms (12 classifiers (widgets) from Orange and 11 classifiers from Weka, implemented as web services),
– Performance evaluation (Evaluate Detection Algorithms, Aggregate Detection Results, Evaluate Repeated Detection, Average and Standard Deviation),
– Visualization (Data Table Information, Data Table Viewer, Evaluation Results Table, Evaluation Results Chart, Viper Visual Performance Evaluation and NoiseRank interactive widgets).

The ClowdFlows platform offers subprocess widgets and the *'for loop'* subprocess widget, which allow repeated experimental evaluation with different initialization parameters. We included into the platform also 26 standard datasets from the UCI repository (Frank and Asuncion 2010), which can be used for extensive experimental evaluation of data mining algorithms. Experiments can be repeated and shared as public workflows that are accessible through unique URL addresses. The ClowdFlows platform is available at http://www.clowdflows.org/.

5.2 NoiseRank implementation

The aim of the NoiseRank (ensemble-based noise detection and ranking) methodology is to support domain experts in identifying noisy, outlier and/or erroneous data instances. The user should be able to select the noise detection algorithms to be used in the ensemble-based noise detection process. Therefore, we have implemented the NoiseRank methodology as a workflow in the ClowdFlows platform, which now offers widgets implementing classification and saturation noise filters, and enables the inclusion of external user specific noise detection algorithms available as web services. Figure 11 presents the NoiseRank workflow applied to a medical dataset using the noise detection algorithms presented in Sect. 3.2.

The NoiseRank methodology workflow returns a visual representation of a list of potentially noisy or outlier instances ranked according to the decreasing number of elementary noise detection algorithms[11] which identified an instance as noisy. Figure 12 shows the actual output of the NoiseRank workflow/widget, when applied to the coronary heart disease (CHD) data presented in Sect. 3.4.1. By selecting instances

[11] Note that all the noise detection algorithms described in Sect. 3.2 were used, except for *PruneSF* which is unable to cope with missing values.

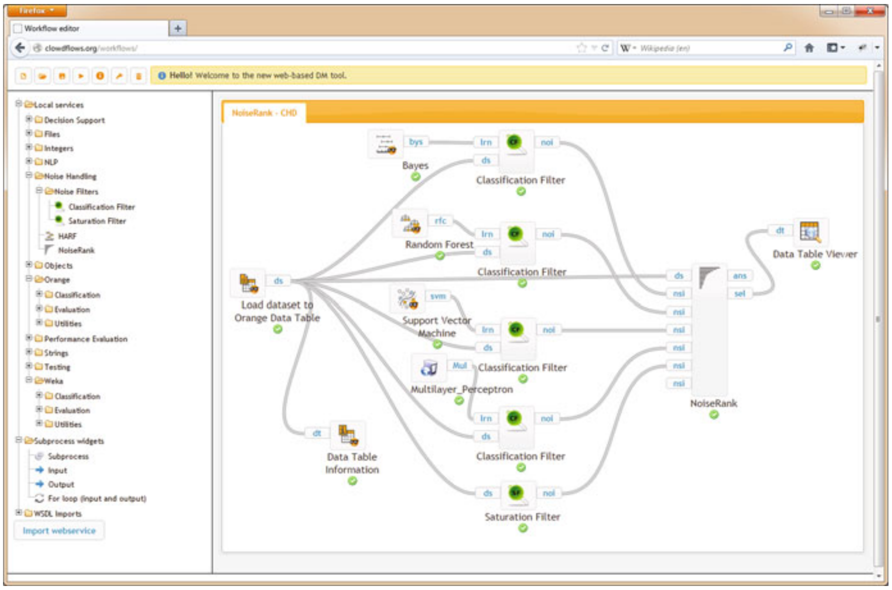


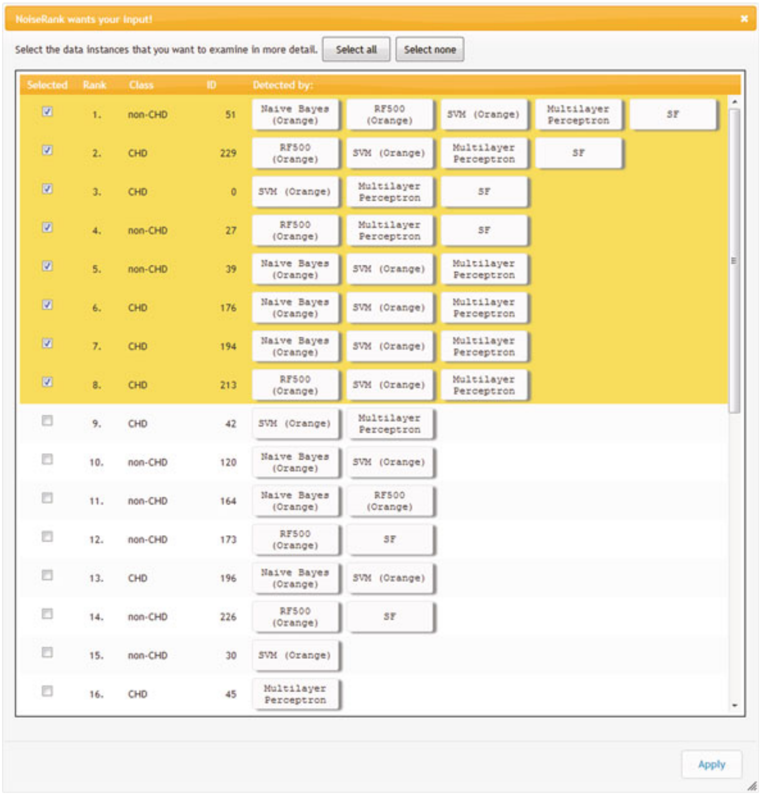
Fig. 11 Example workflow of the NOISERANK methodology

from the ranked list the user can inspect them and decide if they are noise due to errors in the data (attribute or class noise), outliers or special cases of the concept represented in the domain, or just false alarms. The NOISERANK workflow presented in Fig. 11 is publicly accessible at: http://www.clowdflows.org/workflow/115.

5.3 VIPER implementation

In the CLOWDFLOWS platform we have implemented the widgets required for constructing the workflow of the visual performance evaluation methodology VIPER. Thereby, the user can evaluate his own noise detection algorithm (or an ensemble of algorithms) and compare its noise detection performance to the performance of other (existing) algorithms in the precision-recall space. User specific noise detection algorithms that are available as web services can be easily included into the workflow for the VIPER visual performance evaluation. Widgets supporting different input formats are available: string, ORANGE data table and WEKA instances, while the output of noise detection web services should be a simple list of zero-based indices of the detected noisy instances in the initial dataset. Figure 13 presents an example workflow of our VIPER visualization approach for performance comparison and evaluation of a sample noise detection algorithm (HARF) against other existing algorithms on a dataset with artificially injected random noise, which is publicly accessible at http://www.clowdflows.org/workflow/43.

At the end of such workflows, which model noise detection performance evaluation experiments, the VIPER *Visual Performance Evaluation* widget is placed. This is the main widget that visualizes the results and enables performance comparison and


NoiseRank wants your input!

Select the data instances that you want to examine in more detail. Select all Select none

Selected	Rank	Class	ID	Detected by:				
✓	1.	non-CHD	51	Naive Bayes (Orange)	RF500 (Orange)	SVM (Orange)	Multilayer Perceptron	SF
✓	2.	CHD	229	RF500 (Orange)	SVM (Orange)	Multilayer Perceptron	SF	
✓	3.	CHD	0	SVM (Orange)	Multilayer Perceptron	SF		
✓	4.	non-CHD	27	RF500 (Orange)	Multilayer Perceptron	SF		
✓	5.	non-CHD	39	Naive Bayes (Orange)	SVM (Orange)	Multilayer Perceptron		
✓	6.	CHD	176	Naive Bayes (Orange)	SVM (Orange)	Multilayer Perceptron		
✓	7.	CHD	194	Naive Bayes (Orange)	SVM (Orange)	Multilayer Perceptron		
✓	8.	CHD	213	RF500 (Orange)	SVM (Orange)	Multilayer Perceptron		
☐	9.	CHD	42	SVM (Orange)	Multilayer Perceptron			
☐	10.	non-CHD	120	Naive Bayes (Orange)	SVM (Orange)			
☐	11.	non-CHD	164	Naive Bayes (Orange)	RF500 (Orange)			
☐	12.	non-CHD	173	RF500 (Orange)	SF			
☐	13.	CHD	196	Naive Bayes (Orange)	SVM (Orange)			
☐	14.	non-CHD	226	RF500 (Orange)	SF			
☐	15.	non-CHD	30	SVM (Orange)				
☐	16.	CHD	45	Multilayer Perceptron				

Apply

Fig. 12 Visualization output of the NoiseRank widget. The displayed instances, ranked according to the number of noise detection algorithms identifying them as noise, can be selected for further inspection

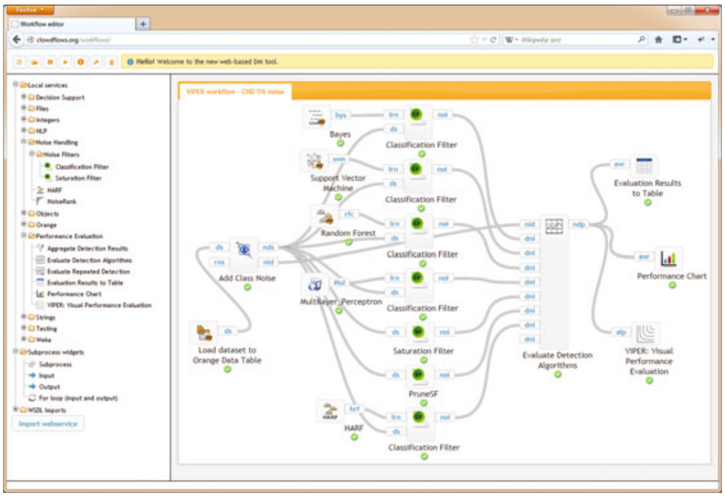


Fig. 13 An example workflow for VIPER evaluation and comparison of new noise detection algorithms against other algorithms

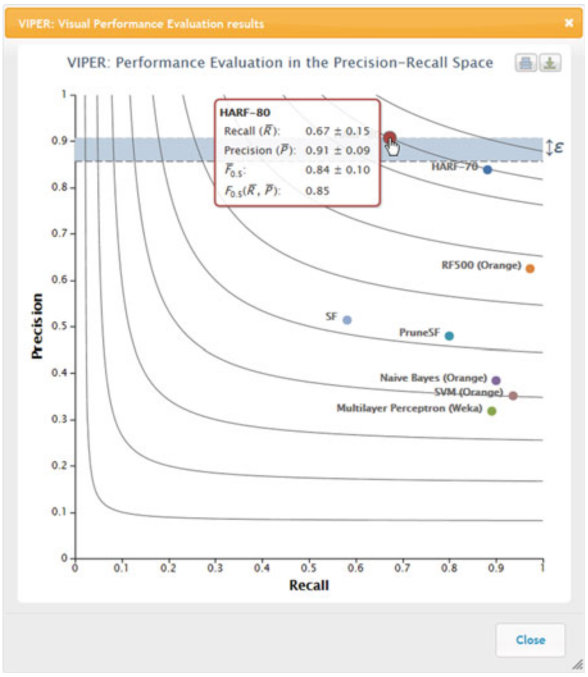

VIPER: Visual Performance Evaluation results
VIPER: Performance Evaluation in the Precision-Recall Space
HARF-80
Recall ($\bar{R}$): 0.67 ± 0.15
Precision ($\bar{P}$): 0.91 ± 0.09
$\bar{F}_{0.5}$: 0.84 ± 0.10
$F_{0.5}(\bar{R}, \bar{P})$: 0.85
↕ε
HARF-70
RF500 (Orange)
SF
PruneSF
Naive Bayes (Orange)
SVM (Orange)
Multilayer Perceptron (Weka)
Precision
Recall
Close

Fig. 14 Additional functionality of the VIPER interactive visualization. Visualization of selected F-isolines for easier comparison and tooltips with additional information on specific algorithm performance

evaluation in the precision-recall space. The output of the VIPER *Visual Performance Evaluation* widget is an interactive chart[12] visualizing performance results of noise detection algorithms according to the VIPER visual evaluation approach. F-scores and F-isolines are computed and drawn based on the provided β parameter in the *Evaluate* widget (the default value is set to 1). The ε-proximity of maximal achieved precision of noise detection is set as a parameter of the VIPER widget (by default set to 0.05). Additionally, the user can obtain more information (algorithm name, precision, recall and F-score results, including standard deviations in case of repeated evaluation) by hovering over algorithm performance results or points on the F-isolines; by clicking on the algorithm performance results which draws their corresponding F-isolines enabling easier comparison; and by selecting an area which is zoomed in, thereby enabling the inspection of performance results in a densely 'populated' results area. An example of the interactive VIPER visualization showing some of its functionality is presented in Fig. 14.

6 Summary and conclusions

This paper presents several contributions, the most important being a methodology for ensemble-based noise ranking, a visualization methodology for algorithm performance evaluation, and their implementation in a publicly accessible web platform.

[12] The implementation uses the HIGHCHARTS charting library, available at http://www.highcharts.com.

Our first contribution is the NoiseRank ensemble-based methodology and a tool for explicit noise detection and ranking, enabling expert inspection of potentially noisy instances in the phase of data understanding, data cleaning and outlier identification. In contrast to other noise ranking approaches where the ranking is produced only by one noise detection algorithm (Van Hulse et al. 2007) or by aggregated predictions of only one machine learning algorithm trained on bootstrap samples of data (Zhong et al. 2005), our approach ranks noisy instances according to the predictions of several different noise detection algorithms thus assuring more reliable results.

We tested this methodology on a real-life medical coronary heart disease dataset where a group of top-ranked noisy instances identified by NoiseRank was given to the domain expert for evaluation. The expert analysis proved that the highest ranked noisy instance, detected by the consensus of all elementary noise detection algorithms, was previously indeed incorrectly classified. Another interesting result of the expert analysis was the confirmation of the existence of outliers, which provide new insights and imply the need of increased human awareness of these special medical cases.

The results obtained by NoiseRank depend on the selection of elementary noise detection algorithms in the ensemble as well as on their ability to detect noisy instances in data. Evaluating the performance of noise detection algorithms is therefore of crucial importance for the successful composition of efficient noise detection ensembles.

Our second contribution is the Viper evaluation methodology which supports effortless interpretation and comparison of noise detection performance by offering comprehensive and intuitively understandable visualization of results. Unlike common indirect evaluation approaches, which measure noise detection only by its influence on the classification accuracy (Brodley and Friedl 1999; Gamberger et al. 2000; Zhu and Wu 2004; Khoshgoftaar et al. 2006; Libralon et al. 2009), the Viper methodology addresses noise detection performance directly be measuring precision and recall of noise detection on data with known (or artificially injected) noisy instances. Compared to other quantitative evaluation approaches working with known or injected noisy instances (Verbaeten 2002; Verbaeten and Van Assche 2003; Zhu et al. 2003; Miranda et al. 2009; Khoshgoftaar et al. 2004; Van Hulse and Khoshgoftaar 2006; Sluban et al. 2010, 2011), our methodology integrates two new evaluation methods which trade off precision and recall: the F-isoline evaluation method and the ε-proximity evaluation method.

The Viper methodology presents a novel kind of visual performance evaluation in the precision-recall space, which has—to the best of our knowledge—not been proposed before in the noise handling literature. In this paper it was compared to standard performance evaluation of noise detection and it has shown to be a valuable approach that enables intuitive evaluation, interpretation and comparison of performance results. This visual performance evaluation methodology could, however, be used also for the evaluation of information retrieval and entity recognition algorithms, as well as for any algorithm for which the evaluation in terms of precision, recall and the F-measure is most suitable.

Finally, we implemented the developed approaches for ensemble-based noise ranking and visual performance evaluation of noise detection algorithms in the ClowdFlows (Kranjc et al. 2012) web platform for construction, execution and sharing of data mining workflows. The implementation of NoiseRank extends the use of ensemble filtering by Verbaeten and Van Assche (2003) and Khoshgoftaar et al. (2005, 2006) by enabling the user to construct ensembles with his own noise detection algorithms. In addition it also allows to select the desired filtering level (agreement among algorithms in the ensemble) or only specific interesting instances by inspecting an interactive ranking visualization. Similarly, the VIPER implementation enables the user to evaluate his own noise detection algorithms through interactive results visualization in the precision-recall space, which enables easy and intuitive comparison of performance results. The CLOWDFLOWS platform implements all the widgets needed for the construction of workflows/experiments with the NOISERANK tool as well as for the VIPER visual performance evaluation methodology. The provided links (http://www.clowdflows.org/workflow/115 and http://www.clowdflows.org/workflow/43) enable public accessibility of the workflows and the repeatability of the experiments described in our work.

Acknowledgments We are grateful to Goran Krstačić from the Institute for Cardiovascular Diseases and Rehabilitation, Zagreb, Croatia, for his evaluation of the given list of potentially noisy patient records in the Coronary Heart Disease medical domain. We also thank two colleagues from the Knowledge Technologies Department at Jožef Stefan Institute, Ljubljana, Slovenia: Janez Kranjc for advice concerning the implementation of the developed approaches in the CLOWDFLOWS platform and Vid Podpečan for the implementation of WEKA algorithms as web services. This work was partially funded by the Slovenian Research Agency and by the European Commission in the context of the FP7 project FIRST, Large scale information extraction and integration infrastructure for supporting financial decision making, under the grant agreement n. 257928.

Appendix 1: Evaluation of HARF with different agreement levels

We here present in more detail our High Agreement Random Forest noise detection algorithm (HARF), introduced in our paper (Sluban et al. 2010). The HARF noise detection algorithm is used as an example of a new algorithm whose noise detection performance is evaluated and compared to the performance of other existing approaches by our VIPER visual performance evaluation methodology. This appendix briefly presents HARF and experimentally shows that HARF-70 and HARF-80 are the best performing variants of the HARF noise detection algorithm.

Classification filters using Random Forest (RF) classifiers (actually) act like an ensemble noise filter, since randomly generated decision trees vote for the class to be assigned to an instance. However, if the class which got the maximal number of votes won only by a single vote (over the 50–50 % voting outcome), this result is not reliable, since the decision trees are generated randomly. Therefore, we decided to demand high agreement of classifiers (decision trees), i.e., a significantly increased majority of votes, namely, for two class domains used in our experiments this means that over 50 % (e.g., 60, 70, 80 or 90 %) of the randomly generated decision trees should classify an instance into the opposite class in order to accept it as noisy.

We have tested and implemented this high agreement based approach in the classification filtering manner with the RF classifier with 500 decision trees and with four different agreement levels: 60, 70, 80 and 90 %, referred to as HARF-60, HARF-70, HARF-80 and HARF-90, respectively. The obtained noise detection results are presented in part C of Table 3.

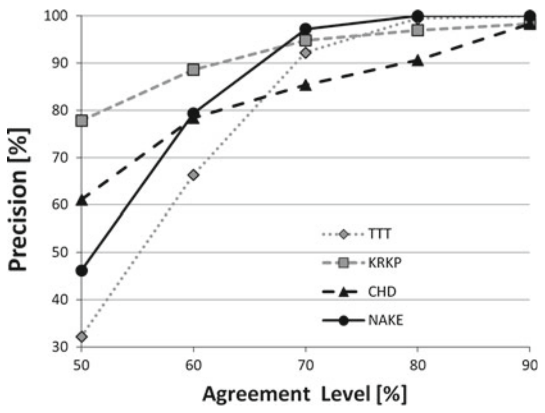


Fig. 15 Precision of HARF noise detection based on the agreement level for RF on all domains with 5 % injected noise

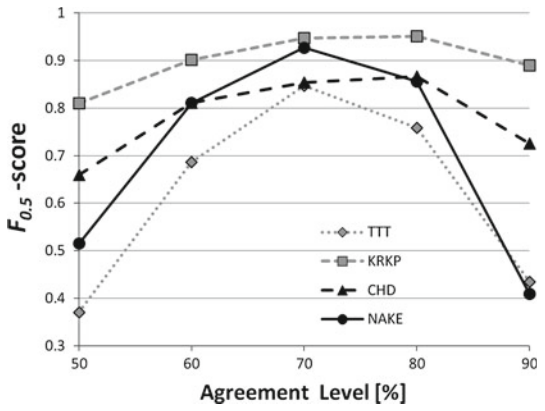


Fig. 16 $F_{0.5}$-scores of HARF noise detection based on the agreement level for RF on all domains with 5 % injected noise

The results in part C of Table 3 prove that our request for high agreement of decision trees in the Random Forest leads to excellent performance results of the HARF noise detection algorithm. Figure 15 clearly indicates the increase in precision of noise detection based on the increased agreement level, for all experimental domains with 5 % injected noise.[13] However, the best results of the proposed HARF noise detection algorithm in terms of $F_{0.5}$-scores are achieved at agreement levels 70 and 80 % as shown in Figure 16 and presented in bold in part C of Table 3. Therefore, only the HARF noise detection algorithms using RF500 with 70 and 80 % agreement level—referred to as HARF-70 and HARF-80, respectively—are used in further evaluation and comparisons of noise detection algorithms in this paper.

[13] Similar results as in Figs. 15 and 16 are observed on domains with 2 and 10 % noise level.

Appendix 2: Discussion of tabular noise detection results

This appendix presents and discusses the performance results of different noise detection algorithms and their ensembles evaluated on four domains with three different levels of randomly injected class noise. The precision and $F_{0.5}$-score results in Tables 3 and 4 present averages and standard deviations over ten repetitions of noise detection on a given domain with a given level of randomly injected class noise. The best precision and the best $F_{0.5}$-score results in each table are printed in bold.

Observing the results in part (A) of Table 3 we can see that among the classification filters the RF and SVM filters generally perform best, RF on the KRKP and CHD datasets, SVM on the TTT dataset and both RF and SVM on the NAKE dataset. The NN filter is quite good on NAKE at lower noise levels, but otherwise achieves only average performance. Noise detection with the Bayes classification filter shows overall the worst performance results, except for the CHD domain where it outperforms NN and is comparable to SVM.

The performance of the saturation filters presented in part (B) of Table 3 is between the worst and the best classification filters. Except for lower noise levels on the KRKP dataset, where SF even outperforms the best classification filter (RF), the saturation filters are not really comparable to the best classification filtering results. On the other hand, among the saturation filters, SF achieves higher or comparable (on the NAKE dataset) precision results like PruneSF, whereas the $F_{0.5}$-scores of PruneSF are not much lower, expect for the KRKP dataset, which indicates that PruneSF has higher recall of injected class noise than SF.

Ensemble filtering using a majority voting scheme (a simple majority of votes of elementary noise filters in the given ensemble) did not outperform the precision of injected noise detection achieved by the best elementary filtering algorithms alone.[14] On the other hand, with consensus voting we significantly improved the precision of noise detection as well as the $F_{0.5}$-scores. Therefore, from now on, when we talk about ensemble noise filters in this paper we actually refer to ensemble noise filters using the consensus voting scheme.

Performance results of the ten ensemble noise filters constructed from different elementary noise detection algorithms can be found in Table 4. The results show that in terms of precision using many diverse elementary noise detection algorithms in the ensemble proves to be better. Even including a different type of worst performing noise filter like Bayes, this does not reduce the ensemble's precision but it actually improves it, as can be seen by comparing precision results of Ens2 and Ens3, Ens5 and Ens6, as well as Ens8 and Ens9. The highest precision among noise detection ensembles was hence not surprisingly achieved by Ens10 which uses all elementary noise filters examined in this work. However, considering the $F_{0.5}$-scores it is interesting to notice that ensembles without SF perform best, namely Ens1, Ens2 and Ens3 perform better on all datasets at higher noise levels (5–10 %), whereas Ens7, Ens8 and Ens9 are better at lower noise levels (2–5 %).

[14] Tabular results are available by request from the authors.

Table 3 Precision and $F_{0.5}$-score results of elementary and HARF noise detection algorithms on 4 datasets with various levels of injected class noise

		(A) Classification filters								(B) Saturation filters	
	Noise level	Bayes		RF500		SVM		NN		SF	
Dataset	(%)	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$
TTT	2	0.04 ± 0.01	0.05 ± 0.01	0.15 ± 0.01	0.18 ± 0.02	0.54 ± 0.00	0.60 ± 0.00	0.34 ± 0.04	0.39 ± 0.04	0.19 ± 0.03	0.22 ± 0.03
	5	0.11 ± 0.01	0.14 ± 0.01	0.34 ± 0.03	0.39 ± 0.03	0.75 ± 0.01	0.79 ± 0.00	0.42 ± 0.03	0.47 ± 0.03	0.39 ± 0.05	0.42 ± 0.05
	10	0.20 ± 0.01	0.23 ± 0.01	0.46 ± 0.03	0.51 ± 0.03	0.87 ± 0.01	**0.89** ± 0.00	0.44 ± 0.02	0.49 ± 0.02	0.50 ± 0.05	0.51 ± 0.04
KRKP	2	0.13 ± 0.00	0.15 ± 0.00	0.60 ± 0.01	0.65 ± 0.01	0.19 ± 0.01	0.22 ± 0.01	0.54 ± 0.03	0.59 ± 0.03	0.81 ± 0.06	0.81 ± 0.06
	5	0.26 ± 0.01	0.31 ± 0.01	0.79 ± 0.02	0.83 ± 0.01	0.38 ± 0.00	0.43 ± 0.00	0.58 ± 0.02	0.63 ± 0.02	0.74 ± 0.05	0.72 ± 0.05
	10	0.42 ± 0.01	0.46 ± 0.01	0.87 ± 0.01	0.89 ± 0.01	0.57 ± 0.01	0.62 ± 0.01	0.56 ± 0.01	0.61 ± 0.01	0.61 ± 0.06	0.57 ± 0.06
CHD	2	0.22 ± 0.03	0.27 ± 0.03	0.37 ± 0.04	0.43 ± 0.04	0.19 ± 0.02	0.22 ± 0.02	0.17 ± 0.02	0.21 ± 0.03	0.32 ± 0.08	0.35 ± 0.09
	5	0.38 ± 0.04	0.42 ± 0.04	0.64 ± 0.04	0.69 ± 0.04	0.35 ± 0.05	0.40 ± 0.05	0.30 ± 0.06	0.34 ± 0.06	0.50 ± 0.06	0.48 ± 0.06
	10	0.51 ± 0.05	0.55 ± 0.06	0.73 ± 0.05	0.77 ± 0.04	0.53 ± 0.07	0.57 ± 0.07	0.39 ± 0.04	0.43 ± 0.04	0.60 ± 0.09	0.54 ± 0.08
NAKE	2	0.18 ± 0.02	0.22 ± 0.02	0.24 ± 0.02	0.29 ± 0.03	0.28 ± 0.03	0.32 ± 0.03	0.33 ± 0.04	0.38 ± 0.04	0.22 ± 0.10	0.25 ± 0.11
	5	0.36 ± 0.02	0.40 ± 0.02	0.52 ± 0.02	0.57 ± 0.02	0.48 ± 0.02	0.54 ± 0.02	0.47 ± 0.05	0.52 ± 0.05	0.37 ± 0.09	0.36 ± 0.08
	10	0.55 ± 0.04	0.60 ± 0.03	0.67 ± 0.03	0.71 ± 0.02	0.70 ± 0.03	0.73 ± 0.03	0.52 ± 0.03	0.57 ± 0.03	0.42 ± 0.12	0.36 ± 0.10

Table 3 continued

		(C) HARF with agreement level								(B) Saturation filters	
	Noise level	60 %		70 %		80 %		90 %		PruneSF	
Dataset	(%)	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$
TTT	2	0.45 ± 0.09	0.49 ± 0.10	0.87 ± 0.12	**0.80** ± 0.09	**1.00** ± 0.00	0.73 ± 0.10	**1.00** ± 0.00	0.44 ± 0.11	0.14 ± 0.02	0.17 ± 0.02
	5	0.63 ± 0.09	0.66 ± 0.08	0.91 ± 0.05	**0.83** ± 0.05	**1.00** ± 0.01	0.76 ± 0.06	**1.00** ± 0.00	0.45 ± 0.09	0.26 ± 0.01	0.30 ± 0.02
	10	0.70 ± 0.07	0.71 ± 0.06	0.90 ± 0.05	0.81 ± 0.04	0.99 ± 0.01	0.71 ± 0.05	**1.00** ± 0.00	0.33 ± 0.05	0.35 ± 0.03	0.38 ± 0.03
KRKP	2	0.78 ± 0.03	0.82 ± 0.03	0.88 ± 0.02	0.90 ± 0.02	0.93 ± 0.01	**0.92** ± 0.01	**0.95** ± 0.02	0.89 ± 0.02	0.46 ± 0.02	0.51 ± 0.02
	5	0.88 ± 0.01	0.90 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.97 ± 0.01	**0.95** ± 0.01	**0.98** ± 0.00	0.88 ± 0.01	0.48 ± 0.02	0.53 ± 0.02
	10	0.94 ± 0.01	0.94 ± 0.01	0.97 ± 0.01	**0.96** ± 0.01	**0.99** ± 0.00	0.95 ± 0.01	**0.99** ± 0.00	0.83 ± 0.01	0.50 ± 0.01	0.54 ± 0.01
CHD	2	0.54 ± 0.03	0.59 ± 0.04	0.65 ± 0.07	0.69 ± 0.08	0.80 ± 0.07	**0.81** ± 0.08	**0.98** ± 0.08	**0.81** ± 0.11	0.24 ± 0.04	0.27 ± 0.05
	5	0.80 ± 0.04	0.82 ± 0.04	0.82 ± 0.05	0.83 ± 0.05	0.92 ± 0.06	**0.89** ± 0.05	**1.00** ± 0.00	0.63 ± 0.13	0.44 ± 0.10	0.46 ± 0.10
	10	0.88 ± 0.06	0.88 ± 0.05	0.93 ± 0.04	**0.90** ± 0.04	0.96 ± 0.04	0.84 ± 0.05	**1.00** ± 0.00	0.47 ± 0.13	0.63 ± 0.06	0.63 ± 0.05
NAKE	2	0.63 ± 0.06	0.67 ± 0.06	0.90 ± 0.05	**0.88** ± 0.06	**1.00** ± 0.00	**0.88** ± 0.07	**1.00** ± 0.00	0.59 ± 0.17	0.15 ± 0.03	0.18 ± 0.04
	5	0.76 ± 0.05	0.78 ± 0.04	0.95 ± 0.05	**0.90** ± 0.05	**1.00** ± 0.00	0.84 ± 0.05	0.90 ± 0.30	0.39 ± 0.17	0.31 ± 0.05	0.34 ± 0.06
	10	0.89 ± 0.05	0.88 ± 0.04	0.99 ± 0.02	**0.91** ± 0.03	**1.00** ± 0.00	0.74 ± 0.07	0.90 ± 0.30	0.16 ± 0.08	0.45 ± 0.03	0.46 ± 0.03

Table 4 Precision and $F_{0.5}$-score results of different noise detection ensembles on four datasets with various levels of injected class noise

Dataset	Noise level (%)	Ens1 (B, RF, SVM)		Ens2 (RF, SVM, NN)		Ens3 (B, RF, SVM, NN)		Ens4 (B, RF, SVM, SF)		Ens5 (RF, SVM, NN, SF)	
		Pr	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$
TTT	2	0.56 ± 0.05	0.57 ± 0.05	0.64 ± 0.08	0.67 ± 0.07	0.60 ± 0.09	0.61 ± 0.09	0.90 ± 0.09	0.72 ± 0.11	0.92 ± 0.10	0.78 ± 0.09
	5	0.80 ± 0.02	0.79 ± 0.03	0.84 ± 0.01	0.84 ± 0.01	0.84 ± 0.02	0.81 ± 0.02	0.97 ± 0.04	0.83 ± 0.05	0.97 ± 0.03	0.86 ± 0.05
	10	0.89 ± 0.02	0.83 ± 0.02	0.92 ± 0.01	**0.89** ± 0.01	0.91 ± 0.02	0.81 ± 0.03	0.97 ± 0.02	0.75 ± 0.05	0.97 ± 0.02	0.78 ± 0.03
KRKP	2	0.70 ± 0.02	0.72 ± 0.02	0.84 ± 0.02	0.85 ± 0.02	0.87 ± 0.03	0.86 ± 0.03	0.90 ± 0.03	0.85 ± 0.03	0.93 ± 0.02	0.88 ± 0.02
	5	0.87 ± 0.02	0.87 ± 0.02	0.92 ± 0.02	0.91 ± 0.02	0.94 ± 0.01	0.91 ± 0.01	**0.97** ± 0.02	0.84 ± 0.03	0.96 ± 0.02	0.86 ± 0.02
	10	0.93 ± 0.01	0.91 ± 0.01	0.95 ± 0.01	**0.93** ± 0.01	0.96 ± 0.01	0.92 ± 0.01	0.96 ± 0.01	0.71 ± 0.05	0.96 ± 0.02	0.73 ± 0.04
CHD	2	0.66 ± 0.10	0.70 ± 0.09	0.59 ± 0.12	0.63 ± 0.12	0.71 ± 0.09	0.73 ± 0.09	0.76 ± 0.18	0.69 ± 0.14	0.81 ± 0.16	0.72 ± 0.11
	5	0.87 ± 0.06	**0.84** ± 0.06	0.82 ± 0.04	0.81 ± 0.04	0.87 ± 0.06	0.83 ± 0.08	0.89 ± 0.12	0.69 ± 0.11	0.86 ± 0.11	0.67 ± 0.06
	10	0.87 ± 0.06	**0.85** ± 0.05	0.85 ± 0.06	0.83 ± 0.06	0.90 ± 0.06	**0.85** ± 0.06	0.90 ± 0.10	0.67 ± 0.08	0.92 ± 0.08	0.65 ± 0.08
NAKE	2	0.41 ± 0.03	0.45 ± 0.03	0.48 ± 0.04	0.53 ± 0.04	0.58 ± 0.06	0.62 ± 0.06	0.73 ± 0.15	0.63 ± 0.15	0.90 ± 0.17	0.72 ± 0.15
	5	0.67 ± 0.05	0.70 ± 0.05	0.71 ± 0.06	0.75 ± 0.05	0.78 ± 0.07	**0.79** ± 0.06	0.81 ± 0.15	0.62 ± 0.12	0.87 ± 0.09	0.66 ± 0.11
	10	0.81 ± 0.03	0.83 ± 0.02	0.81 ± 0.03	0.82 ± 0.03	0.87 ± 0.04	**0.86** ± 0.03	0.86 ± 0.07	0.52 ± 0.11	0.89 ± 0.07	0.53 ± 0.10

Table 4 continued

Dataset	Noise level (%)	Ens6 (B, RF, SVM, NN, SF)		Ens7 (B, RF, SVM, PSF)		Ens8 (RF, SVM, NN, PSF)		Ens9 (B, RF, SVM, NN, PSF)		Ens10 (B,RF,SVM,NN,SF,PSF)	
		Pr	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$	*Pr*	$F_{0.5}$
TTT	2	0.93 ± 0.09	0.74 ± 0.11	0.86 ± 0.09	0.75 ± 0.07	0.91 ± 0.07	**0.84** ± 0.07	0.89 ± 0.07	0.76 ± 0.08	**0.95** ± 0.09	0.73 ± 0.10
	5	0.98 ± 0.03	0.82 ± 0.05	0.98 ± 0.02	0.87 ± 0.04	0.98 ± 0.02	**0.91** ± 0.03	0.99 ± 0.01	0.87 ± 0.04	**1.00** ± 0.01	0.80 ± 0.06
	10	0.97 ± 0.03	0.70 ± 0.05	0.97 ± 0.03	0.81 ± 0.03	0.98 ± 0.02	0.85 ± 0.03	0.98 ± 0.02	0.78 ± 0.05	**0.99** ± 0.02	0.69 ± 0.05
KRKP	2	0.93 ± 0.02	0.86 ± 0.02	0.90 ± 0.02	0.87 ± 0.02	0.92 ± 0.01	**0.91** ± 0.01	0.93 ± 0.01	0.90 ± 0.02	**0.94** ± 0.02	0.86 ± 0.02
	5	**0.97** ± 0.02	0.84 ± 0.03	0.95 ± 0.01	0.90 ± 0.01	0.95 ± 0.02	**0.92** ± 0.01	0.96 ± 0.01	0.91 ± 0.01	**0.97** ± 0.02	0.84 ± 0.03
	10	**0.97** ± 0.02	0.70 ± 0.04	0.96 ± 0.01	0.88 ± 0.01	**0.97** ± 0.01	0.90 ± 0.01	**0.97** ± 0.01	0.88 ± 0.01	**0.97** ± 0.01	0.68 ± 0.04
CHD	2	0.83 ± 0.15	0.73 ± 0.09	0.72 ± 0.17	0.71 ± 0.15	0.81 ± 0.15	**0.76** ± 0.12	0.81 ± 0.14	**0.76** ± 0.10	**0.86** ± 0.14	0.75 ± 0.09
	5	0.89 ± 0.12	0.68 ± 0.11	0.88 ± 0.10	0.73 ± 0.12	0.84 ± 0.10	0.71 ± 0.10	0.88 ± 0.10	0.71 ± 0.13	**0.89** ± 0.12	0.60 ± 0.14
	10	0.95 ± 0.07	0.65 ± 0.10	0.91 ± 0.06	0.80 ± 0.09	0.91 ± 0.06	0.80 ± 0.08	0.93 ± 0.07	0.78 ± 0.12	**0.96** ± 0.07	0.65 ± 0.10
NAKE	2	0.94 ± 0.12	0.75 ± 0.14	0.54 ± 0.06	0.54 ± 0.07	0.84 ± 0.11	0.77 ± 0.11	0.87 ± 0.11	**0.79** ± 0.10	**0.98** ± 0.08	0.74 ± 0.12
	5	0.89 ± 0.11	0.66 ± 0.11	0.77 ± 0.10	0.70 ± 0.11	0.88 ± 0.06	0.77 ± 0.07	0.88 ± 0.08	0.77 ± 0.09	**0.91** ± 0.09	0.66 ± 0.12
	10	0.92 ± 0.07	0.51 ± 0.11	0.87 ± 0.04	0.75 ± 0.05	0.90 ± 0.05	0.76 ± 0.04	0.91 ± 0.04	0.76 ± 0.05	**0.93** ± 0.06	0.48 ± 0.11

Last we discuss the results presented in part (C) of Table 3 for our HARF noise detection algorithm (described in more detail in Appendix 1) with four different agreement levels: 60, 70, 80 and 90 %, also referred to as HARF-60, HARF-70, HARF-80 and HARF-90, respectively. The results in part (C) of Table 3 as well as Figs. 15 and 16 in Appendix 1 show that HARF-80 and HARF-90 are the most precise noise detection algorithms presented in Table 3 and achieve comparable or higher precision than the precisest ensemble filters in Table 4. Furthermore, in terms of the $F_{0.5}$-scores the best results are achieved by HARF-70 and HARF-80, which shows to be also significantly better than the results achieved by elementary noise detection algorithms, expect in one case where SVM outperforms the HARF algorithms on the TTT dataset with 10 % noise level. Compared to ensemble noise filters the HARF-70 and HARF-80 achieve lower $F_{0.5}$-scores only on the TTT dataset, but are slightly better on the KRKP and CHD datasets and better on the NAKE dataset.

At this point we can observe another interesting phenomenon, indicating that classification noise filters are robust noise detection algorithms. By analyzing the results in Tables 3 and 4 we see that the proportion of detected noise (i.e., the precision of noise detection) increases with the increased abundance of noise (increased levels of artificially inserted noise 2, 5, 10 %). Clearly, with increased noise levels the classifiers are less accurate (their accuracy drops), therefore leading to the increased number of misclassified instances, i.e., the number of instances which are considered potentially noisy increases. As an observed side effect, the proportion of detected artificially corrupted instances increases as well.

References

Aggarwal CC, Yu PS (2001) Outlier detection for high dimensional data. In: Proceedings of the 2001 ACM SIGMOD international conference on management of data, pp 37–46

Brodley CE, Friedl MA (1999) Identifying mislabeled training data. J Artif Intell Res 11:131–167

Deanfield J, Shea M, Ribiero P, de Landsheere C, Wilson R, Horlock P, Selwyn A (1984) Transient st-segment depression as a marker of myocardial ischemia during daily life. Am J Cardiol 54(10):1195–1200

Demšar J, Zupan B, Leban G, Curk T (2004) Orange: From experimental machine learning to interactive data mining. In: Boulicaut JF, Esposito F, Giannotti F, Pedreschi D (eds) Knowledge discovery in databases: PKDD 2004, lecture notes in computer science. Springer, vol 3202, pp 537–539

Frank A, Asuncion A (2010) UCI machine learning repository. URL http://archive.ics.uci.edu/ml

Fürnkranz J (1997) Pruning algorithms for rule learning. Mach Learn 27:139–171

Gamberger D, Lavrač N (1997) Conditions for Occam's razor applicability and noise elimination. In: Lecture notes in artificial intelligence: machine learning: ECML-97, vol 1224, pp 108–123

Gamberger D, Lavrač N (2002) Expert-guided subgroup discovery: methodology and application. J Artif Intell Res 17:501–527

Gamberger D, Lavrač N, Grošelj C (1999) Experiments with noise filtering in a medical domain. In: Proceedings of 16th international conference on machine learning—ICML, Morgan Kaufmann, pp 143–151

Gamberger D, Lavrač N, Džeroski S (2000) Noise detection and elimination in data preprocessing: experiments in medical domains. Appl Artif Intell 14(2):205–223

Gamberger D, Lavrač N, Krstačić G (2003) Active subgroup mining: a case study in a coronary heart disease risk group detection. Artif Intell Med 28:27–57

Gelfand S, Ravishankar C, Delp E (1991) An iterative growing and pruning algorithm for classification tree design. IEEE Trans Pattern Anal Mach Intell 13:163–174

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. SIGKDD Explor Newsl 11(1):10–18

Hodge V, Austin J (2004) A survey of outlier detection methodologies. Artif Intell Rev 22(2):85–126

Khoshgoftaar TM, Rebours P (2004) Generating multiple noise elimination filters with the ensemble-partitioning filter. In: Proceedings of the 2004 IEEE International Conference on information reuse and integration. IEEE Systems, Man, and Cybernetics Society, pp 369–375.

Khoshgoftaar T, Seliya N, Gao K (2004) Rule-based noise detection for software measurement data. In: Proceedings of the 2004 IEEE international conference on information reuse and integration, 2004 (IRI 2004), pp 302–307

Khoshgoftaar TM, Zhong S, Joshi V (2005) Enhancing software quality estimation using ensemble-classifier based noise filtering. Intell Data Anal 9(1):3–27

Khoshgoftaar TM, Joshi VH, Seliya N (2006) Detecting noisy instances with the ensemble filter: a study in software quality estimation. Int J Softw Eng Knowl Eng 16(1):53–76

Kranjc J, Podpečan V, Lavrač N (2012) Clowdflows: a cloud based scientific workflow platform. In: Flach P, Bie T, Cristianini N (eds) Machine learning and knowledge discovery in databases, lecture notes in computer science. Springer, Berlin, vol 7524, pp 816–819

Libralon GL, Carvalho ACPLF, Lorena AC (2009) Ensembles of pre-processing techniques for noise detection in gene expression data. In: Proceedings of the 15th international conference on advances in neuro-information processing—volume part I, ICONIP'08. Springer, Berlin, pp 486–493

Manning CD, Raghavan P, Schtze H (2008) Introduction to Information Retrieval. Cambridge University Press, New York

Maron D, Ridker P, Pearson A (1998) Risk factors and the prevention of coronary heart disease. In: Wayne A, Schlant R, Fuster V (eds) HURST'S: the Heart, pp 1175–1195

Mingers J (1989) An empirical comparison of pruning methods for decision tree induction. Mach Learn 4:227–243

Miranda A, Garcia L, Carvalho A, Lorena A (2009) Use of classification algorithms in noise detection and elimination. In: Hybrid artificial intelligence systems, lecture notes in computer science. Springer, Berlin, vol 5572, pp 417–424

Niblett T, Bratko I (1987) Learning decision rules in noisy domains. In: Bramer M (ed) Research and development in expert systems. Cambridge University Press, Cambridge

Pollak S (2009) Text classification of articles on kenyan elections. In: Proceedings of the 4th language & technology conference: human language technologies as a challenge for computer science and linguistics, pp 229–233.

Pollak S, Coesemans R, Daelemans W, Lavrač N (2011) Detecting contrast patterns in newspaper articles by combining discourse analysis and text mining. Pragmatics 21(4):674–683

Quinlan JR (1987) Simplifying decision trees. Int J Man-Mach Stud 27:221–234

Sluban B, Gamberger D, Lavrač N (2010) Advances in class noise detection. In: Coelho H, Studer R, Wooldridge M (eds) Proceedings of the 19th European conference on artificial intelligence (ECAI 2010), pp 1105–1106

Sluban B, Gamberger D, Lavrač N (2011) Performance analysis of class noise detection algorithms. In: Ågotnes T (ed) STAIRS 2010—proceedings of the fifth starting AI researchers' symposium, pp 303–314

Teng CM (1999) Correcting noisy data. In: Proceedings of the sixteenth international conference on machine learning, pp 239–248

Van Hulse JD, Khoshgoftaar TM (2006) Class noise detection using frequent itemsets. Intell Data Anal 10(6):487–507

Van Hulse JD, Khoshgoftaar TM, Huang H (2007) The pairwise attribute noise detection algorithm. Knowl Inf Syst 11(2):171–190

Verbaeten S (2002) Identifying mislabeled training examples in ilp classification problems. In: Proceedings of twelfth Belgian-Dutch conference on machine learning, pp 1–8

Verbaeten S, Van Assche A (2003) Ensemble methods for noise elimination in classification problems. In: Windeatt T, Roli F (eds) Multiple classifier systems, lecture notes in computer science. Springer, Berlin, vol 2709, pp 317–325

Wiswedel B, Berthold MR (2005) Fuzzy clustering in parallel universes with noise detection. In: Proceedings of the ICDM 2005 workshop on computational intelligence in data mining, pp 29–37

Yin H, Dong H, Li Y (2009) A cluster-based noise detection algorithm. International workshop on database technology and applications, pp 386–389

Zhong S, Tang W, Khoshgoftaar TM (2005) Boosted noise filters for identifying mislabeled data. Technical report, Department of computer science and engineering, Florida Atlantic University

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

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