Artif Intell Rev (2018) 49:105–130
https://doi.org/10.1007/s10462-016-9518-2

The robustness of majority voting compared to filtering misclassified instances in supervised classification tasks

Michael R. Smith[1] · **Tony Martinez**[1]

Published online: 22 September 2016

Abstract Removing or filtering outliers and mislabeled instances prior to training a learning algorithm has been shown to increase classification accuracy, especially in noisy data sets. A popular approach is to remove any instance that is misclassified by a learning algorithm. However, the use of ensemble methods has also been shown to generally increase classification accuracy. In this paper, we extensively examine filtering and ensembling. We examine 9 learning algorithms individually and ensembled together as filtering algorithms as well as the effects of filtering in the 9 chosen learning algorithms on a set of 54 data sets. We compare the filtering results with using a majority voting ensemble. We find that the majority voting ensemble significantly outperforms filtering unless there are high amounts of noise present in the data set. Additionally, for most cases, using an ensemble of learning algorithms for filtering produces a greater increase in classification accuracy than using a single learning algorithm for filtering.

Keywords Voting ensemble · Class noise · Filtering · Machine learning

1 Introduction

The goal of supervised machine learning is to induce an accurate generalizing function $\mathcal{F} : X \mapsto Y$ from a set of input feature vectors $X = \{x_1, x_2, \ldots, x_n\}$ and a corresponding set of of label vectors $Y = \{y_1, y_2, \ldots, y_n\}$. The quality of the induced function $\mathcal{F}$ by a learning algorithm is dependent on the quality of the data used for training. However, many real-world data sets are inherently noisy where the noise in a data set can be label noise and/or attribute noise. Label noise has been shown to be more detrimental than attribute noise (Zhu and Wu 2004) and is the focus of this paper. Noise can arise from various sources such as subjectivity, human errors, and sensor malfunctions. Most learning algorithms are designed

✉ Michael R. Smith
msmith4@sandia.gov

[1] Department of Computer Science, Brigham Young University, Provo, UT 84602, USA

to tolerate a certain degree of noise by avoiding overfitting the training data. There are two general approaches for handling class noise: (1) creating learning algorithms that are robust to noise such as the C4.5 algorithm for decision trees (Quinlan 1993) and (2) preprocessing the data prior to inducing a model of the data such as filtering (Wilson 1972; Brodley and Friedl 1999), weighting (Rebbapragada and Brodley 2007; Smith and Martinez 2014) or correcting (Teng 2003) noisy instances.

Previous works have generally examined filtering in a limited context using a single or very few learning algorithms and/or using a limited number of data sets. This may be in part due to the extra computational requirement to first filter a data set and then induce a model of the data using the filtered data set. As such, previous works were generally limited to investigating relatively fast learning algorithms such as decision trees (John 1995) and nearest-neighbor algorithms (Tomek 1976; Wilson and Martinez 2000). In addition, filtering prior to using instance-based learning algorithms was motivated in part to reduce the number of instances that have to be stored and because instance-based learning algorithms are more sensitive to noise than other learning algorithms. Most previous works also added artificial noise to the data set to show that filtering, weighting, or cleaning the data set is beneficial (5–50 % of the instances become noisy). In this work, we examine filtering misclassified instances and using a majority voting ensemble on a set of 54 data sets and 9 learning algorithms *without* adding artificial noise. The artificial noise was added in previous works to show that filtering/weighting/cleaning provided significant improvements with noisy data sets. Within the context of the benefits of filtering established by the previous work, we examine the extent to which filtering affects the performance of a learning algorithm without adding artificial noise to a data set. This also avoids making assumptions about the generation of the noise which may or may not be accurate. It also shows the effect of filtering on the inherent noise in real-world data sets that is not known before hand.

The results provide insights on the robustness of a majority voting ensemble and when to employ a misclassification filter. Using a larger number of data sets allows for more statistical confidence in the results than if only a small number of data sets are used. We find that, in general, a voting ensemble is robust to noise and achieves significantly higher classification accuracy trained on unfiltered data than a single learning algorithm trained on filtered data. For filtering, we find that using an ensemble filter achieves significantly higher classification accuracy than using a single learning algorithm filter. On data sets with higher percentages of inherent noisy instances, however, using the ensemble filter achieves higher classification accuracy than a voting ensemble for some learning algorithms. Training a voting ensemble on filtered training data significantly *decreases* classification accuracy compared to training a voting ensemble on unfiltered training data. This is likely due to a reduction of diversity in the induced models of the ensemble.

In the next section, we present previous works for handling noise in supervised classification problems. A mathematical motivation for filtering misclassified instances is presented in Sect. 3. We then present our experimental methodology in Sect. 4 followed by a presentation of the results in Sect. 5. In Sect. 6 we provide conclusions and directions for future work.

2 Related work

As many real-wold data sets are inherently noisy, most learning algorithms are designed to tolerate a certain degree of noise. Typically, learning algorithms are designed to be somewhat robust to noise by making a trade-off between the complexity of the induced model and

optimizing the induced function on the training data to prevent overfit. Some techniques to avoid overfit include early stopping using a validation set, pruning (such as in the C4.5 algorithm for decision trees Quinlan 1993), or regularization by adding a complexity penalty to the loss function Bishop and Nasrabadi (2006). Some previous works have examined how class noise and attribute noise affects the performance of various learning algorithms (Zhu and Wu 2004; Nettleton et al. 2010) and found that class noise is generally more harmful than attribute noise and that noise in the training set is more harmful than noise in the test set. Further, some learning algorithms have been adapted specifically to better handle label noise. For example, noisy instances are problematic for boosting algorithms (Schapire 1990; Freund 1990) where more weight is placed upon misclassified instances, which often include mislabeled and noisy instances. To address this, Servedio (2003) presented a boosting algorithm that does not place too much weight on any single training instance. For support vector machines, Collobert et al. (2006) use the ramp-loss function to place a bound on the maximum penalty for an instance that lies on the wrong side of the margin. Lawrence and Schölkopf Lawrence and Schölkopf (2001) explicitly model the possibility that an instance is mislabeled using a generative model and then use expectation maximization to update the probability that an instance is mislabeled.

Preprocessing the data set is another approach that explicitly handles label noise. This can be done by removing noisy instances, weighting the instances, or correcting incorrect labels. All three approaches first attempt to identify which instances are noisy by various criteria. Filtering noisy instances has received much attention and has generally resulted in an increase in classification accuracy (Gamberger et al. 2000; Smith and Martinez 2011). One frequently used filtering technique removes any instance that is misclassified by a learning algorithm (Wilson 1972) or set of learning algorithms (Brodley and Friedl 1999). Verbaeten and Van Assche (2003) further pursued the idea of using an ensemble for filtering using ideas from boosting and bagging. Other approaches use learning algorithm heuristics to remove noisy instances. Segata et al. (2009), for example, remove instances that are too close or on the wrong side of the decision surface generated by a support vector machine. Zeng and Martinez (2003) remove instances while training a neural network that have a low probability of being labeled correctly where the probability is calculated using the output from the neural network. Filtering has the potential downside of discarding useful instances. However, it is assumed that there are significantly more non-noisy instances and that throwing away a few correct instances with the noisy instances will not have a negative impact on a large data set.

Weighting the instances in a training set has the benefit of not discarding any instances. Rebbapragada and Brodley (2007) weight the instances using expectation maximization to cluster instances that belong to a pair of the classes. The probabilities between classes for each instances is compiled and used to weight the influence of each instance. Smith and Martinez (2014) examine weighting the instances based on their probability of being misclassified.

Similar to weighting the training instances, data cleaning does not discard any instances, but rather strives to correct the noise in the instances. As in filtering, the output from a learning algorithm has been used to clean the data. Automatic data enhancement (Zeng and Martinez 2001) uses the output from a neural network to correct the label for training instances that have a low probability of being correctly labeled. Polishing (Teng 2000, 2003) trains a learning algorithm (in this case a decision tree) to predict the value for each attribute (including the class). The predicted (i.e. corrected) attribute values for the instances that increase generalization accuracy on a validation set are used instead of the uncleaned attribute values.

We differ from the related work in that we do not add artificial noise to the data sets when we examine filtering. Thus, we avoid making any assumptions about the noise source and

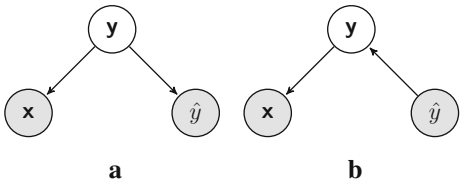

Fig. 1 Graphical model of the generative probabilistic model proposed by Lawrence and Schölkopf (2001)

focus on the noise inherent in the data sets. We also examine the effects of filtering on a larger set of learning algorithms and data sets providing more significance to the generality of the results.

3 Modeling class noise in a discriminative model

Lawrence and Schölkopf (2001) proposed to model a data set probabilistically using a generative model that models the noise process. They assume that the joint distribution $p(x, y, \hat{y})$ (where x is the set of input features, $\hat{y}$ is the observed, possibly noisy, class label given in the training set, and y is the actual unkown class label) is factorized as $p(\hat{y}|y)p(x|y)p(y)$ as shown in Fig. 1a. However, since modeling the prior distribution of the unobserved random variable y is not feasible, it is more practical to estimate the prior distribution of $p(\hat{y})$ with some assumptions about the class noise as shown in Fig. 1b.

Here, we follow the premise of Lawrence and Schölkopf by explicitly modeling the possibility that an instance is misclassified. Rather than using a generative model, though, we use a discriminative model since we are focusing on classification tasks and do not require the full joint distribution. Also, discriminative models have been shown to yield better performance on classification tasks (Ng and Jordan 2001). Using a discrimintative model that accounts for class noise motivates our investigation of filtering and using a majority voting ensemble.

Let T be a training set composed of instances $\langle x_i, \hat{y}_i \rangle$ drawn i.i.d. from the underlying data distribution $\mathcal{D}$. Each instance is composed of an input vector x_i with a corresponding possibly noisy label vector $\hat{y}_i$. Given the training data T, a learning algorithm generally seeks to find the most probable hypothesis h that maps each $x_i \mapsto \hat{y}_i$. For supervised classification problems, most learning algorithms maximize $p(\hat{y}_i|x_i, h)$ for all instances in T. This is shown graphically in Fig. 2a where the probabilities are estimated using a discriminative approach such as a neural network or a decision tree to induce a hypothesis of the data. Using Bayes' rule and decomposing T into its individual constituent instances, the maximum a posteriori hypothesis is:

$$\begin{aligned} \underset{h \in \mathcal{H}}{\operatorname{argmax}}\, p(h|T) &= \frac{p(T|h)p(h)}{p(T)} \\ &\propto \prod_i p(x_i, \hat{y}_i|h)p(h) \\ \underset{h \in \mathcal{H}}{\operatorname{argmax}}\, p(h|T) &= \prod_i p(\hat{y}_i|x_i, h)p(x_i|h)p(h). \end{aligned} \tag{1}$$

In Eq. 1, the MAP hypothesis h is found by finding a global optima where all instances are included in the optimization problem. However, noisy instances are often detrimental for finding the global optima since they are not representative of the true (and unknown) underlying data distribution $\mathcal{D}$. The possibility of label noise is not explicitly modeled in this

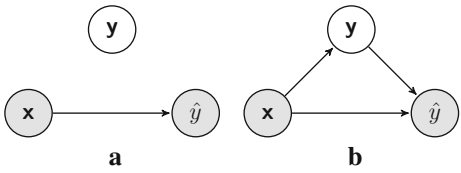

Fig. 2 Graphical representation of a discriminative probabilistic model for **a** $p(\hat{y}|x)p(x)$ and **b** $p(\hat{y}|x, y)p(y|x)p(x)$

form—completely ignoring y_i. Thus, label noise is generally handled by avoiding overfit such that more probable, simpler hypotheses are preferred ($p(h)$). The possibility of label noise can be modeled explicitly by including the latent random variable y_i with x_i and $\hat{y}_i$. Thus, an instance is the triplet $\langle x_i, \hat{y}_i, y_i \rangle$ and a supervised learning algorithm seeks to maximize $p(\hat{y}_i|x_i, y, h)$—modeled graphically in Fig. 2b. Using the model in Fig. 2b, the MAP hypothesis becomes:

$$\begin{aligned} \underset{h \in \mathcal{H}}{\operatorname{argmax}}\, p(h|T) &\propto \prod_i p(x_i, y_i, \hat{y}_i|h)p(h) \\ &= \prod_i p(\hat{y}_i|x_i, y_i, h)p(y_i|x_i, h)p(x_i|h)p(h). \end{aligned} \tag{2}$$

Equation 2 shows that for an instance x_i, the probability of an observed class label ($p(\hat{y}_i|x_i, y_i, h)$) should be weighted by the probability of the actual class ($p(y_i|x_i, h)$).

What we are really interested in is the probability that $y_i = \hat{y}_i$. Using a discriminative model h trained on T, we can calculate $p(y_i|\hat{y}_i, x_i, h)$ as

$$p(y_i|\hat{y}_i, x_i, h) = p(y_i|\hat{y}_i, h)p(\hat{y}_i|x_i, h)p(h).$$

Since the quantity $p(y|\hat{y}_i, h)$ is unknown, $p(y_i|\hat{y}_i, x_i, h)$ can be approximated as $p(\hat{y}_i|x_i, h)$ assuming that $p(y_i|\hat{y}_i, h)$ is represented in h. In other words, the induced discriminative model is able to model if one class label is more likely than another class label given an observed, possibly noisy, label. Otherwise, all class labels are assumed to be equally likely given an observed label. Thus, $p(y_i|\hat{y}_i, x_i, h)$ can be approximated by finding the class distributions for a given x_i from an induced discriminative model. That is, after training a learning algorithm on T, the class distribution for an instance x_i can be calculated based on the output from the learning algorithm. As shown in Eq. 1, $p(\hat{y}_i|x_i, h)$ is found naturally through a derivation of Bayes' law. The quantity $p(\hat{y}_i|x_i, h)$ is the maximum likelihood of an instance given a hypothesis h which a learning algorithm tries to maximize for each instance. Further, the dependence on a specific h can be removed by summing over all possible hypotheses h in $\mathcal{H}$ and multiplying each $p(\hat{y}_i|x_i, h)$ by $p(h)$:

$$p(y_i|\hat{y}_i, x_i) \approx p(\hat{y}_i|x_i) = \sum_{h \in \mathcal{H}} p(\hat{y}_i|x_i, h)p(h). \tag{3}$$

This formulation is infeasible though because (1) it is not practical (or possible) to sum over the set of all hypotheses, (2) calculating $p(h)$ is non-trivial, and 3) not all learning algorithms produce a probability distribution. These limitation make probabilistic generative models attractive, such as the kernel Fisher discriminant algorithm (Lawrence and Schölkopf 2001). However, for classification tasks, generative models generally have a higher asymptotic error than discriminative models (Ng and Jordan 2001). The following section shows how we estimate $p(y_i|\hat{y}_i, x_i, h)$.

This framework for modeling class noise in a discriminative model motivates the use of removing instances with low $p(y_i|\hat{y}_i, h)$ and the use of ensembles to lessen the dependence

on a given hypothesis h. Following Eq. 2, removing instances with low $p(y_i|x_i, h)$ will increase the global $p(h|T)$ since $p(\hat{y}_i|x_i, y_i, h)$ will be low. Further, following Eq. 3, an ensemble should theoretically be more robust to the bias of a particular hypothesis and noise by utilizing multiple overfit avoidance techniques. This motivates our examination of a majority voting ensemble composed of models induced by different learning algorithms[1] as a filtering technique and as a classifier.

4 Methodology

In this section, we present how we calculate $p(y_i|\hat{y}_i, x_i, h)$ and the learning algorithms and data set that we use in our analysis. We also provide an overview of our experimentss.

4.1 Calculating $p(y_i|\hat{y}_i, x_i, h)$

To calculate $p(y_i|\hat{y}_i, x_i, h)$ for each instance, we use an induced model from training a learning algorithm on the training set T. To lessen the dependence of $p(y_i|\hat{y}_i, x_i, h)$ on a particular h, we estimate marginalizing over the hypothesis space $\mathcal{H}$ by selecting a diverse set of learning algorithms to represent $\mathcal{H}$. The diversity of the learning algorithm refers to the learning algorithms not having the same classification for all of the instances and is determined using unsupervised meta-learning (UML) (Lee and Giraud-Carrier 2011). UML first uses Classifier Output Difference (COD) (Peterson and Martinez 2005) to measure the diversity between learning algorithms. COD measures the distance between two learning algorithms as the probability that the learning algorithms make different predictions. UML then clusters the learning algorithms based on their COD scores with hierarchical agglomerative clustering. We considered 20 learning algorithms from Weka with their default parameters (Hall et al. 2009). The resulting dendrogram is shown in Fig. 3, where the height of the line connecting two clusters corresponds to the distance (COD value) between them. A cut-point of 0.18 was chosen to create 9 clusters and a representative algorithm from each cluster was used to create a diverse set of learning algorithms. The learning algorithms that were used are listed in Table 1. UML provides a diverse set of learning algorithms intended to be representative of $\mathcal{H}$.

4.2 Experiments

Given a method for estimating $p(y_i|\hat{y}_i, x_i, h)$ and for lessening the dependence on a specific h, we examine several techniques for filtering instances with low $p(y_i|\hat{y}_i, x_i, h)$ and for constructing a voting ensemble.

4.2.1 Misclassification filters

In this paper, we examine misclassification filters which filter any instance that is misclassified by a given learning algorithm. Given that a number of different learning algorithms could be employed for filtering, we conduct an extensive evaluation of filtering misclassified instances using a diverse set of learning algorithms as described in the previous section. Each learning algorithm first filters instances from the training set that were misclassified and then induces

[1] As opposed to an ensemble composed of models linduced by the same learning algorithm such as bagging or boosting.

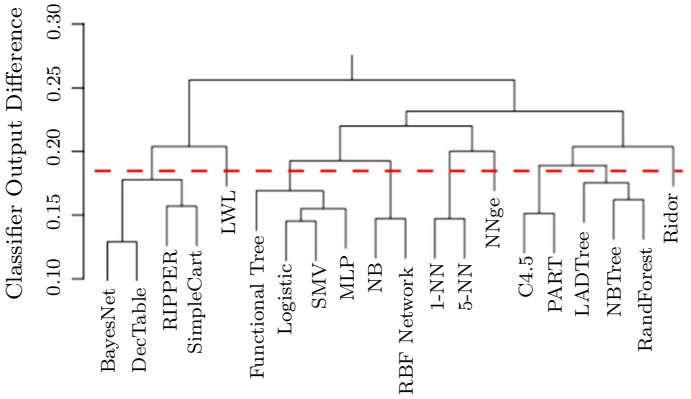

Fig. 3 Dendrogram of the considered learning algorithms clustered using unsupervised metalearning based on their classifier output difference

Table 1 Set of learning algorithms used for filtering

Learning algorithms
Multilayer Perceptron trained with Back Propagation (MLP)
Decision Tree (C4.5) (Quinlan 1993)
Locally Weighted Learning (LWL)
5-Nearest Neighbors (5-NN)
Nearest Neighbor with generalization (NNge)
Naïve Bayes (NB)
RIpple DOwn Rule learner (RIDOR)
Random Forest (RandForest)
Repeated Incremental Pruning to Produce Error Reduction (RIPPER)

a model of the data using the filtered training set. Misclassification filters using a single learning algorithm establish a good baseline to compare against.

4.2.2 Ensemble filter

We also examine using an ensemble filter–removing instances that are misclassified by different percentages of the 9 learning algorithms. The ensemble filter more closely approximates $p(y|\hat{y}_i, x_i)$ from Eq. 3 since it sums over a set of learning algorithms (which in this case were chosen to be diverse and represent a larger subset of the hypothesis space $\mathcal{H}$) lessening the dependence on a single hypothesis h. For the ensemble filter, $p(y|\hat{y}_i, x_i)$ is estimated using a subset of learning algorithms $\mathcal{L}$:

$$p(y|\hat{y}_i, x_i) \approx p(\hat{y}_i|x_i, \mathcal{L}) \approx \frac{1}{|\mathcal{L}|} \sum_{j=1}^{|\mathcal{L}|} p(\hat{y}_i|x_i, l_j(T)) \tag{4}$$

where $l_j(T)$ is the hypothesis from the jth learning algorithm trained on training set T. From Eq. 3, $p(h)$ is estimated as $\frac{1}{|\mathcal{L}|}$ for the jth hypothesis generated from training the learning algorithms in $\mathcal{L}$ on T and as zero for all of the other hypotheses in $\mathcal{H}$. Also, $p(\hat{y}_i|x_i, l_j(T))$

Algorithm 1 Adaptively constructing a filter set.

Let F be the filter set used for filtering and $\mathcal{L}$ be the set of candidate learning algorithms for F.
Initialize F to the empty set: $F \leftarrow \{\}$
Initialize the current accuracy to the accuracy from an empty filter set: $currAcc \leftarrow runLA(\{\})$. $runLA(F)$ returns the accuracy from a learning algorithm trained on a data set filtered with F.
while $\mathcal{L} \neq \{\}$ **do**
  $bestAcc \leftarrow currAcc$; $bestLA \leftarrow null$;
  for all $g \in \mathcal{L}$ **do**
    $tempF \leftarrow F + g$; $acc \leftarrow runLA(tempF)$;
    if $acc > bestAcc$ **then**
      $bestAcc \leftarrow acc$; $bestLA \leftarrow g$;
    end if
  end for
  if $bestAcc > currAcc$ **then**
    $\mathcal{L} \leftarrow \mathcal{L} - bestLA$; $F \leftarrow F + bestLA$; $currAcc \leftarrow bestAcc$;
  else
    break;
  end if
end while

is estimated using the indicator function $(h(x_i) = \hat{y}_i)$ since not all learning algorithms produce a probability distribution over the output classes. Set up as such, the ensemble filter filters an instance that is misclassified by x % of the learning algorithms in the ensemble. In this paper, we examine an ensemble filter, removing instances that are misclassified by 50, 70, and 90 percent of the learning algorithms in the ensemble. One of the problems of using an ensemble filter is having to choose the percentage of learning algorithms that misclassify an instance for filtering. For the results, we report the accuracy from the percentage that produces the highest accuracy using 5 by 10-fold cross-validation to choose the best percentage for each data set. This method highlights the impact of using an ensemble filter, however, in practice a validation set is often used to determine the percentage that would be used.

4.2.3 Adaptive filter

We also examine an adaptive filtering approach that iteratively adds a learning algorithm to a set of filtering learning algorithms by selecting the learning algorithm from a set of candidate learning algorithms $\mathcal{L}$ that produces the highest classification accuracy on a validation set when added to the set of learning algorithms used for filtering, as shown in Algorithm 1. The function $runLA(F)$ trains a learning algorithm on a data set using the filter set F to filter the instances and returns the accuracy of the learning algorithm on a validation set. As with the ensemble filter, instances are removed that are misclassified by a given percentage of the filtering learning algorithms. The idea is to choose an optimal subset of learning algorithms through a greedy search of the candidate filtering algorithms. For the results, we report the accuracy from the percentage that produces the highest accuracy using n-fold cross-validation to choose the best percentage for each data set.

4.2.4 Majority voting esemble

In addition, we examine the use of a majority voting ensemble compared to filtering misclassified instances. The majority voting ensemble is composed of the diverse set learning algorithms as described in Sect. 4.1. The classification of an instance is the class that receives the most votes from the trained ensembled models.

4.3 Evaluation

Each method is evaluated using 5 by 10-fold cross-validation (running 10-fold cross validation 5 times, each time with a different seed to partition the data). We examine filtering using the 9 chosen learning algorithms on a set of 47 data sets from the UCI data repository and 7 non-UCI data sets (Thomson and McQueen 1996; Salojärvi et al. 2005; Sayyad Shirabad and Menzies 2005; Stiglic and Kokol 2009). For filtering, we examine two methods for training the filtering algorithms: (1) removing the misclassified instances when trained on the entire training set and (2) using cross-validation on the training set that removes instances that are misclassified in the validation set. The number of folds for using cross-validation for the training set was set to 2, 3, 4, and 5. Table 2 shows the data sets used in this study organized according to the number of instances, number of attributes, and attribute type. The non-UCI data sets are in bold.

Statistical significance between pairs of algorithms is determined using the Wilcoxon signed-ranks test as suggested by Demšar (2006). We emphasize the extensive nature of this evaluation:

1. Filtering is examined on 9 diverse learning algorithms.
2. 9 diverse learning algorithms are examined as misclassification filtering techniques.
3. In addition to the single algorithm misclassification filters, an ensemble filter and an adaptive filter are examined.
4. Each filtering method is examined on a set of 54 data sets using 5 by 10-fold cross-validation.
5. Each filtering method is examined on the entire training set as well as using 2-, 3-, 4-, and 5-fold cross-validation.
6. A majority voting ensemble is examined on a set of 54 data sets using 5 by 10-fold cross-validation.

5 Results

In this section, we present the results from filtering the 54 data sets using (1) a biased misclassification filter (the same learning algorithm to filter misclassified instances is used to induce a model of the data), (2) the ensemble filter, and (3) the adaptive filter as well as a voting ensemble. Our results can be summarized as follows: (1) using a voting ensemble is generally preferable to filtering, (2) when filtering, our results suggest that using the ensemble filter in all cases produces the best results, and (3) filtering is preferable to a voting ensemble in *some* cases with high amounts of noise. Except for the adaptive filter, we find that using cross-validation on the training set for filtering results in a lower accuracy (and often significantly lower) than using the entire training set and, as such, the following results for the biased filter and the ensemble filter are from using the entire training set for filtering rather than using cross-validation. We first show how filtering affects each learning algorithm in Sect. 5.1. Next, we examine using a set of data set measures to determine when filtering is the most effective in Sect. 5.2. In Sect. 5.3, we compare filtering with a voting ensemble.

Table 2 Datasets used organized by number of instances (# Ins), number of attributes, and attribute type

# Ins	# Attributes	Attribute type Categorical	Numerical	Mixed
$M < 100$	$k < 10$	Contact lenses		Post-operative **cm1_req**
	$10 < k < 100$	Lung cancer	**desharnais**	Labor **Pasture**
$100 < M < 1000$	$k < 10$	Breast-w	Iris	Badges 2
		Breast cancer	Ecoli	Teaching-assistant
			Pima Indians	
			Glass	
			Bupa	
			Balance Scale	
	$10 < k < 100$	Audiology	Ionosphere	Annealing
		Soybean(large)	Wine	Dermatology
		Lymphography	Sonar	Credit-A
		Congressional-voting records	Heart-Statlog	Credit-G
			ar1	Horse Colic
		Vowel		Heart-c
		Primary-Tumor		Hepatitis
		Zoo		Autos
				Heart-h
				eucalyptus
	$k > 100$		**AP_Breast-Uterus**	Arrhythmia
$1000 < M < 10{,}000$	$k < 10$	Car Evaluation	Yeast	
		Titanic		
	$k < 100$		Waveform-5000	Thyroid-(sick & hypothyroid)
			Segment	
			Spambase	
			Ozone level-detection	
$M > 10{,}000$	$k < 10$	Nursery	MAGIC	
			Telescope	
	$k < 100$	Chess-(King-Rook vs. King-Pawn)		**Eye-movements**

5.1 Filtering results

The results of the biased, ensemble, and adaptive filters are summarized in Table 3—showing the average classification accuracy for each learning algorithm and filtering algorithm pair.[2]

[2] The NNge learning algorithm did not finish running two data sets: eye-movements and Magic telescope. RIPPER did not finish on the lung cancer data set. In these cases, the data sets are omitted from the presented

Table 3 Summary of filtering using the same learning algorithm to filter misclassified instances and to induce a model of the data, the ensemble filter, and the adaptive filter

	MLP	C4.5	IB5	LWL	NB	NNge	RF	Rid	RIP
Orig	81.74	80.80	79.91	72.80	76.94	80.14	82.28	79.90	79.76
Biased	81.72	80.75	79.53	70.91	75.88	80.34	82.14	79.02	79.87
Ensemble	**83.40**	**81.61**	**80.85**	**73.48**	**78.92**	**82.21**	**82.93**	**80.57**	**81.26**
Adaptive	**82.38**	80.63	80.01	**73.44**	**78.48**	81.33	81.87	80.00	80.43

For all learning algorithms, the ensemble filter significantly increases the classification accuracy

The values in bold represent those that are a statistically significant improvement over not filtering. The results of the statistical significance tests for each of the learning algorithms is provided in Tables 10, 11, 12, 13, 14, 15, 16, 17 and 18 in "Appendix 1".

We find that using a biased filter does not significantly increase the classification for any of the learning algorithms and that using a biased filter significantly decreases the classification accuracy for the LWL, naïve Bayes, Ridor and RIPPER learning algorithms (Tables 13, 14, 17, 18). These results suggest that simply removing the misclassified instances by a single learning algorithm is not sufficient. Bear in mind that these results reflect not adding any artificial noise to the training set. In the case where artificial noise is added to the training set (as was commonly done in previous works), using a biased filter may result in an improvement in accuracy. However, most real-world scenarios do not artificially add noise to their data set but are concerned with the inherent noise found within it.

For all of the learning algorithms, the ensemble filter significantly increases the classification accuracy over not filtering and over the other filtering techniques. An ensemble generally provides better predictive performance than any of the constituent learning algorithms (Polikar 2006) and generally yields better results when the underlying ensembled models are diverse (Kuncheva and Whitaker 2003). Thus, by using a more powerful model, only the noisiest instances are removed. This provides empirical evidence supporting the notion that filtering instances with low $p(\hat{y}_i|x_i)$ that are not dependent on a single hypothesis is preferred to filtering instances where the probability of the class is dependent on a particular hypothesis $p(\hat{y}_i|x_i, h)$ as outlined in Eq. 3.

Surprisingly, the adaptive filter does not outperform the ensemble filter and in, one case, it does not even outperform training on unfiltered data. Perhaps this is because it overfits the training data since the best accuracy is chosen on the training set. Adaptive filtering has significantly better results when cross-validation is used to filter misclassified instances as opposed to removing misclassified instances that were also used to train the filtering algorithm. Even with using the results with cross-validation, the results are not significantly better than using the ensemble filter.

Examining each learning algorithm individually, we find that some learning algorithms are more robust to noise than others. To determine which learning algorithms are more robust to noise, we compare the accuracy of the learning algorithms without filtering to the accuracy obtained using the ensemble filter. The p values from the Wilcoxon signed-ranks statistical significance test are shown in Table 4 ordered from greatest (least significant impact) to

Footnote 2 continued
results. As such, NNge was evaluated on a set of 52 data sets and RIPPER was evaluated on a set of 53 data sets.

Table 4 The p values from the Wilcoxon signed-ranks statistical significance test comparing not filtering with the ensemble filter

	RF	C4.5	Rid	IB5	NNge	MLP	LWL	RIP	NB
p val	0.045	0.035	0.019	0.018	0.006	0.004	0.004	<0.001	<0.001

The learning algorithms are ordered in descending order of p value from left to right

least reading from left to right. We see that random forests and decision trees are the most robust to noise as filtering has the least significant impact on their accuracy. This is not too surprising given that the C4.5 algorithm was designed to take noise into account and random forests are built using decision trees. Ridor and 5-nearest neighbor (IB5) are more robust to noise, but still greatly improve with filtering. IB5 is more robust to noise since it compares with the 5 nearest neighbors of an instance. If K were set to 1, then filtering would have a greater effect on the accuracy. Filtering has the most significant effect on the accuracy of the last five learning algorithms: MLP, NNge, LWL, RIPPER, and naïve Bayes.

5.2 Analysis of when to filter

Using only the inherent noise in a data set, the efficacy of filtering is limited and can be detrimental in some data sets. Thus, we examine the cases in which filtering significantly improves the classification accuracy. This investigation is similar to the recent work by Sáez et al. (2013) who investigate creating a set of rules to understand when to filter using a 1-nearest neighbor learning algorithm. They use a set of data complexity measures from Ho and Basu (2002). The complexity measures are designed for binary classification problems, yet we do not limit ourselves to binary classification problems. As such, we use a subset of the data complexity measures shown in Table 5 that have been extended to handle multi-class problems (Orriols-Puig et al. 2009). In addition, we also examine a set of hardness measures (Smith et al. 2014) shown in Table 6. The hardness measures are designed to determine and characterize instances that have a high likelihood of being misclassified and are taken with respect to a specific instance. For the hardness measures, the "disjunct" refers to the class leaf in a decision tree that classifies an investigated instance. We examine using the set of data complexity measures and the hardness measures to create rules and/or a classifier to determine when to use filtering. We set up the classification problem similar to Sáez et al. where the features are the complexity measures and the hardness measures. The class label is set to "TRUE" if filtering significantly improves the classification accuracy for a data set using the Wilcoxon signed-ranks test otherwise it is set to "FALSE". We also examine predicting the difference in accuracy between using and not using a filter. Unlike Sáez et al., we find that the data complexity measures and the hardness measures do *not* create a satisfactory classifier to determine when to filter. Granted, we examine more learning algorithms and do not artificially add noise to the data sets which provides for few data sets where filtering significantly improves the classification accuracy. In the study by Sáez et al., 75 % of the data sets had at least 5 % noise added providing more positive examples. More future work is required to determine when to use filtering on unmodified data sets. Based on our results, we would recommend always using the ensemble filter for all of the learning algorithms as it significantly outperforms the other filtering techniques.

Table 5 List of complexity measures from Ho and Basu (2002)

F2:	*Volume of overlap region*: The overlap of the per-class bounding boxes calculated for each attribute by normalizing the difference of the maximum and minimum values from each class
F3:	*Max individual feature efficiency*: For all of the features, the maximum ratio of the number of instances not in the overlapping region to the total number of instances
F4:	*Collective feature efficiency*: F3 only return the ratio for the attribute that maximizes the ratio. F4 is a measure for all of the attributes
N1:	*Fraction of points on class boundary*: The fraction of instances in a data set that are connected to their nearest neighbors that have a different class in a spanning tree
N2:	*Ratio of ave intra/inter class NN dist*: The average distance to the nearest intra-class neighbors divided by the average distance to the nearest inter-class neighbors
N3:	*Error rate of 1NN classifier*: Leave-one-out error estimate of 1NN
T1:	*Fraction of maximum covering spheres*: The normalized count of the number of clusters of instances containing a single class
T2:	*Ave number of points per dimension*: Compares the number of instances to the number of features

Table 6 List of hardness measures from Smith et al. (2014)

*k*DN	*k-Disagreeing neighbors*: The percentage of the k nearest neighbors (using Euclidean distance) for an instance that do not share its target class value
DS	*Disjunct size*: The number of instances covered by a disjunct that the investigated instance belongs to divided by the number of instances covered by the largest disjunct in an unpruned decision tree induced using C4.5 (Quinlan 1993)
DCP	*Disjunct class percentage*: The number of instances in a disjunct that have the same class label as the investigated instance divided by the total number of instances in the disjunct in a pruned decision tree
TD	*Tree depth*: The depth of the leaf node that classifies an instance in an induced decision tree
CL	*Class likelihood*: The probability that an instance belongs to its class given the input features
CLD	*Class likelihood difference*: The difference between the class likelihood of an instance and the maximum class likelihood for all of the other classes
MV	*Minority value*: The ratio of the number of instances sharing its target class value to the number of instances in the majority class
CB	*Class balance*: The difference of the ratio of the number of instances belonging to a class and the ratio of the classes if they were distributed equally

5.3 Voting ensemble versus filtering

In this section, we compare the results of filtering using the ensemble filter with a voting ensemble. The voting ensemble uses the same learning algorithms as the ensemble filter (Table 1) and the vote from each learning algorithm is equally weighted. Table 7 compares the voting ensemble with using the ensemble filter on each of the investigated learning algorithms giving the average accuracy, the p-value, and the number of times that the accuracy of a voting ensemble is greater than, equal to, or less than using the ensemble filter. The results for each data set are provided in Table 19 in "Appendix 2". With no artificially generated noise, a voting ensemble achieves significantly higher classification accuracy than the ensemble filter

Table 7 Summary of the comparison between a voting ensemble and filtering using the ensemble filter on each learning algorithm

	Ensemble	MLP	C4.5	IB5	LWL	NB
Acc	84.37	83.40	81.61	80.85	73.48	78.92
p value		**0.008**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
>, =, <		33, 1, 20	43, 1, 10	42, 2, 10	47, 1, 6	41, 0, 13
	Ensemble	NNge	RF	Rid	RIP	
Acc	84.37	81.59	82.93	80.57	80.76	
p value		**<0.001**	**<0.001**	**<0.001**	**<0.001**	
>, =, <		44, 2, 8	39, 0, 15	47, 1, 6	44, 1, 9	

The average accuracy for each method over all tested datasets is used to illustrate the differences. Using the ensemble significantly improves the classification accuracy over using the ensemble filter for all of the examined learning algorithms

for each of the examined learning algorithms. This is not too surprising considering that previous research has shown that ensemble methods address issues that are common to all non-ensemble learning algorithms (Dietterich 2000) and that ensemble methods generally obtain a greater accuracy than that from a single learning algorithm that makes up part of the ensemble (Opitz and Maclin 1999). Considering the computational requirements for training, using a voting ensemble for classification rather than filtering appears to be more beneficial.

Many previous studies (Zhu and Wu 2004; Lawrence and Schölkopf 2001; Brodley and Friedl 1999; Verbaeten and Van Assche 2003) have shown that when a large amount of artificial noise is added to a data set (i.e. $\geq$10 %), then filtering outperforms a voting ensemble. We examine which of the 54 data sets have a high percentage of noise using instance hardness (Smith et al. 2014) to identify suspected noisy instances. Instance hardness approximates the likelihood that an instance will be misclassified by evaluating the classification of an instance from a set of learning algorithms $\mathcal{L}$: $p(\hat{y}_i | x_i, \mathcal{L})$. The set of learning algorithms $\mathcal{L}$ is composed of the learning algorithms shown in Table 1. The instances that have a probability greater than 0.9 of being misclassified we consider to be noisy instances. Table 8 shows the accuracies from a voting ensemble and the considered learning algorithms using the ensemble filter for the subset of data sets with more than 10 % noisy instances. Examining the more noisy data sets shows that the gains from using the ensemble filter are more noticeable. However, only 9 out of the 54 investigated data sets were identified as having more than 10 % noisy instances. We ran a Wilcoxon signed-ranks test, but with the small sample size it is difficult to determine the statistical significance of using the ensemble filter over using a voting ensemble. Based on the small sample provided here, training a learning algorithm on a filtered data set is statistically equivalent to training a voting ensemble classifier. The computational complexity required to train an ensemble is less than that to train an ensemble for filtering followed by training another learning algorithm from the filtered data set. A single learning algorithm trained on the filtered data set has the benefit that only one learning algorithm is queried for a novel instance. Future work will include discovering if a smaller subset of learning algorithms for filtering approximates using the ensemble filter in order to reduce the computational complexity.

Examining the more noisy data sets shows that filtering has a more significant effect on classification accuracy, however, the amount of noise is not the only factor that needs to be considered. For example, 32.2 % of the instances in the primary-tumor data set are noisy, yet

Table 8 Comparison of a voting ensemble with the ensemble filter on a subset of data sets where more than 10 % of the constituent instances are noisy

Data set	Ens	MLP	C4.5	IB5	LWL	NB	NNge	RF	Rid	RIP	Per
breastc	73.99	73.43	**75.17**	**74.13**	73.31	73.43	**74.13**	73.19	**74.71**	**74.59**	10.1
arrhyth	**71.11**	70.13	70.65	59.14	57.67	65.63	65.71	66.59	70.65	71.09	12.0
contact	76.67	**83.33**	**83.33**	76.39	76.39	76.39	**80.56**	**80.56**	**79.17**	**77.78**	12.5
lungCan	53.75	52.08	**56.25**	47.92	**55.21**	**55.21**	**56.25**	52.08	51.04	**54.17**	12.5
yeast	**61.08**	59.43	60.13	59.32	40.7	58.15	59.4	**61.08**	59.74	60.01	13.7
cm1_req	75.73	**76.40**	**77.53**	**77.53**	**76.78**	**77.53**	**77.15**	**76.78**	**77.53**	**76.78**	16.9
titanic	**78.72**	78.66	78.68	78.59	77.9	77.77	78.68	78.68	78.28	78.68	16.9
post-op	69.78	**71.11**	**71.11**	**71.11**	**71.11**	**71.11**	**71.11**	**71.11**	**71.11**	**71.11**	26.7
pri-tum	48.08	47.79	41.2	45.82	34.42	**48.57**	44.44	45.23	39.82	40.41	32.2
Acc	67.66	**68.04**	**68.23**	65.55	62.61	67.09	67.49	67.26	66.89	67.18	
p value		0.367	0.820	0.125	0.213	0.410	0.545	0.312	0.410	0.715	
>, =<		6, 0, 3	4, 0, 5	6, 0, 3	6, 0, 3	5, 0, 4	4, 0, 5	5, 1, 3	5, 0, 4	4, 0, 5	

The accuracy of the voting ensemble ("Ens") is in bold if it is greater than the accuracies from using the ensemble filter for the investigated learning algorithms. The accuracy from using the ensemble filter is in bold if it is higher than the accuracy from a voting ensemble. The column "Per" refers to the percentage of instances in the data set that are considered noisy

Table 9 Comparison of a majority voting ensemble trained on unfiltered (Ens) and filtered data (FEns)

	Ens	FEns 50	FEns 70	FEns 90	FEns Max
All					
Accuracy	84.37	83.40	82.21	73.96	83.62
p value		**<0.001**	**<0.001**	**<0.001**	**<0.001**
Greater-equal-less		42, 3, 9	44, 2, 8	48, 1, 5	39, 2, 13
Noisy					
Accuracy	67.66	67.00	67.47	60.52	67.93
p value		0.102	0.455	**0.049**	0.633
Greater-equal-less		7, 0, 2	5, 0, 4	6, 0, 3	5, 0, 4
<90 %					
Accuracy	78.49	77.19	75.74	66.02	77.44
p values		**<0.001**	**<0.001**	**<0.001**	**<0.001**
Greater-equal-less		31, 0, 6	30, 1, 6	34, 0, 3	28, 0, 9
<80 %					
Accuracy	74.70	73.08	71.41	61.49	73.41
p values		**<0.001**	**<0.001**	**<0.001**	**0.002**
Greater-equal-less		24, 0, 3	22, 0, 5	24, 0, 3	21, 0, 6
<70 %					
Accuracy	64.65	61.99	60.41	51.04	62.25
p values		**0.001**	**0.002**	**<0.001**	**0.009**
Greater-equal-less		10, 0, 1	10, 0, 1	10, 0, 1	10, 0, 1
<60 %					
Accuracy	58.44	55.81	53.49	42.56	56.12
p values		**0.016**	**0.016**	**0.016**	**0.016**
Greater-equal-less		6, 0, 0	6, 0, 0	6, 0, 0	6, 0, 0
<50 %					
Accuracy	50.92	48.22	48.07	38.61	49.16
p values		0.250	0.250	0.250	0.250
Greater-equal-less		2, 0, 0	2, 0, 0	2, 0, 0	2, 0, 0

The value after "FEns" represents the percentage of learning algorithms that have to misclassify an instance for it to be filtered from the training set and "Max" uses the accuracy from the percentage that results in the greatest accuracy. Training with unfiltered data is significantly better than training with filtered data for a voting ensemble

only one learning algorithm achieves a greater classification accuracy than the voting ensemble. On the other hand, the classification accuracy on the ar1 and ozone data sets for all of the considered learning algorithms trained on filtered data is greater than using a voting ensemble despite only having 3.3 and 0.5 % noisy instances respectively. Thus, there are other unknown data set features affecting when filtering is appropriate. Future work also includes discovering and examining data set features that are indicative of when filtering should be used.

We further investigate the robustness of the majority voting ensemble to noise by applying the ensemble filter to the training data for the voting ensemble. We find that a majority voting ensemble is significantly better *without* filtering. The summary results are shown in Table 9 and the full results for each data set can be found in Table 20 in "Appendix 2". Table 9 divides the data sets into subsets that have more than 10 % noisy instances ("Noisy"), and

those that have an original accuracy less than 90, 80, 70, 60, and 50 % averaged across the investigated learning algorithms ($<N$ %). Even with harder data sets and more noisy instances, using unfiltered training data produces significantly higher classification accuracy for the voting ensemble. Thus, we find that a majority voting ensemble is more robust to noise than filtering in most cases. The strength of a voting ensemble comes from the diversity of the ensembled learning algorithms. However, the induced models from the learning algorithms trained on the filtered training data are less diverse since the diversity often comes from how a learning algorithm treats a noisy instance, lessening the power of the voting ensemble. This is evidenced as we examined a voting ensemble consisting of C4.5, random forest, and Ridor which are three of the more similar learning algorithms using unsupervised meta-learning (see Sect. 4). When trained on the filtered training data, the less diverse voting ensemble achieves a significantly lower classification average accuracy of 82.09 % compared to 83.62 % from the voting ensemble composed of the 9 examined learning algorithms. Thus, some noise in the training set is beneficial to create diversity in the ensemble.

6 Conclusions and Discussion

In this paper, we presented an extensive empirical evaluation of misclassification filters on a set of 54 multi-class data sets and 9 diverse learning algorithms. As opposed to other work on filtering, we used a large set of data sets and learning algorithms and we did not artificially add noise to the data set. In previous works, noise was added to a data set to verify that the noise filtering method was effective and that filtering was more effective when more noise was present. However, the artificial noise may not be representative of the actual noise and the impact of filtering on an unmodified data set is not always clear.

Using a set of multi-class data sets, we focused our analysis on accuracy. However, for many 2-class problems other metrics could be more indicative of good performance such as precision or recall. We also did not examine the case of class imbalance that may affect the probability of a class or cases where one class may be more important than another such as a false negative in diagnosing a terminal disease. These are important issues that arise in many real-world machine learning applications. In cases of extreme data imbalance, all of the instances of a majority class could be removed because they have low $p(y_i|\hat{y}_i, x_i, h)$. Thus, other techniques to account for class imbalance should also be used. This risk also highlights a benefit of using a voting ensemble as instances will not be discarded. However, the voting ensemble requires a higher computational budget to induce the ensembled models. For dealing with class imbalance or class weighted differently by importance, we suggest using a voting ensemble with another technique that address the class imbalance or weighted classes.

Through our experiments we found that, without artificially adding label noise, using the same learning algorithm for filtering and for inducing a model of the data can be significantly detrimental and does not significantly increase the classification accuracy even when examining harder data sets. Using the ensemble filter significantly improved the accuracy over not filtering and outperformed both the adaptive filtering method and using each learning algorithm individually as a filter for all of the investigated learning algorithms. We compared filtering with a voting ensemble and found that a voting ensemble achieves significantly higher classification accuracy than any of the other considered learning algorithms trained on filtered data. A majority voting ensemble trained on unfiltered data significantly outperforms a voting ensemble trained on filtered data. Thus, a voting ensemble exhibits robustness to noise in the training set and is preferable to filtering.

Appendix 1: Statistical Significance Tables

This section provides the results from the statistical significance tests comparing not filtering with filtering with a biased filter, the ensemble filter, and the adaptive filter for the investigated learning algorithms. The results are in Tables 10, 11, 12, 13, 14, 15, 16, 17 and 18. The p values with a value <0.05 are in bold and "greater-equal-less" refers to the number of times that the algorithm listed in the row is greater than, equal to, or less than the algorithm listed in the column.

Table 10 Pair-wise comparison of filtering for multilayer perceptrons trained with backpropagation

	Orig	Biased	Ensemble	Greedy
Accuracy	81.74	81.87	83.33	82.33
Orig				
p values	1	0.771	1.000	0.953
Greater-equal-less	0, 54, 0	25, 2, 27	16, 2, 36	18, 3, 33
Biased				
p values	0.232	1	1	0.957
Greater-equal-less	27, 2, 25	0, 54, 0	9, 4, 41	23, 2, 29
Ensemble				
p values	**<0.001**	**<0.001**	1	**<0.001**
Greater-equal-less	36, 2, 16	41, 4, 9	0, 54, 0	40, 1, 13
Greedy				
p values	**0.048**	**0.044**	1	1
Greater-equal-less	33, 3, 18	29, 2, 23	13, 1, 40	0, 54, 0

Table 11 Pair-wise comparison of filtering for decision trees

Accuracy	80.80	80.83	81.59	80.56
Orig				
p values	1	0.460	1.000	0.221
Greater-equal-less	0, 54, 0	26, 5, 23	17, 3, 34	29, 2, 23
Biased				
p values	0.544	1	0.999	0.271
Greater-equal-less	23, 5, 26	0, 54, 0	17, 5, 32	29, 1, 24
Ensemble				
p values	**<0.001**	**0.001**	1	**<0.001**
Greater-equal-less	34, 3, 17	32, 5, 17	0, 54, 0	44, 2, 8
Greedy				
p values	0.782	0.732	1	1
Greater-equal-less	23, 2, 29	24, 1, 29	8, 2, 44	0, 54, 0

Table 12 Pair-wise comparison of filtering for 5-nearest neighbors

	Orig	Biased	Ensemble	Greedy
Accuracy	79.91	79.40	80.83	79.91
Orig				
p values	1	0.693	0.985	0.877
Greater-equal-less	0, 54, 0	25, 1, 28	17, 2, 35	20, 2, 32
Biased				
p values	0.310	1	1	0.999
Greater-equal-less	28, 1, 25	0, 54, 0	5, 4, 45	17, 1, 36
Ensemble				
p values	**0.015**	<**0.001**	1	<**0.001**
Greater-equal-less	35, 2, 17	45, 4, 5	0, 54, 0	44, 1, 9
Greedy				
p values	0.125	**0.001**	1	1
Greater-equal-less	32, 2, 20	36, 1, 17	9, 1, 44	0, 54, 0

Table 13 Pair-wise comparison of filtering for locally weighted learning (LWL)

	Orig	Biased	Ensemble	Greedy
Accuracy	72.80	70.91	73.48	73.44
Orig				
p values	1	<**0.001**	0.992	0.988
Greater-equal-less	0, 54, 0	34, 11, 9	14, 9, 31	16, 8, 30
Biased				
p values	0.999	1	1	1
Greater-equal-less	9, 11, 34	0, 54, 0	3, 12, 39	9, 10, 35
Ensemble				
p values	**0.009**	<**0.001**	1	0.595
Greater-equal-less	31, 9, 14	39, 12, 3	0, 54, 0	19, 8, 27
Greedy				
p values	**0.013**	<**0.001**	0.409	1
Greater-equal-less	30, 8, 16	35, 10, 9	27, 8, 19	0, 54, 0

Table 14 Pair-wise comparison of filtering for naïve Bayes

	Orig	Biased	Ensemble	Greedy
Accuracy	76.94	75.84	78.82	78.45
Orig				
p values	1	**0.001**	1.000	0.985
Greater-equal-less	0, 54, 0	38, 0, 16	17, 4, 33	24, 1, 29
Biased				
p values	0.999	1	1	1
Greater-equal-less	16, 0, 38	0, 54, 0	4, 2, 48	10, 2, 42

Table 14 continued

	Orig	Biased	Ensemble	Greedy
Ensemble				
p values	<**0.001**	<**0.001**	1	**0.012**
Greater-equal-less	33, 4, 17	48, 2, 4	0, 54, 0	32, 4, 18
Greedy				
p values	**0.016**	<**0.001**	0.988	1
Greater-equal-less	29, 1, 24	42, 2, 10	18, 4, 32	0, 54, 0

Table 15 Pair-wise comparison of filtering for NNge

	Orig	Biased	Ensemble	Greedy
Accuracy	80.62	80.30	82.18	81.32
Orig				
p values	1	0.080	1.000	0.888
Greater-equal-less	0, 52, 0	25, 5, 22	12, 4, 36	24, 1, 27
Biased				
p values	0.921	1	1	0.992
Greater-equal-less	22, 5, 25	0, 52, 0	12, 2, 38	18, 2, 32
Ensemble				
p values	<**0.001**	<**0.001**	1	<**0.001**
Greater-equal-less	36, 4, 12	38, 2, 12	0, 52, 0	41, 2, 9
Greedy				
p values	0.114	**0.008**	1	1
Greater-equal-less	27, 1, 24	32, 2, 18	9, 2, 41	0, 52, 0

Table 16 Pair-wise comparison of filtering for random forests

	Orig	Biased	Ensemble	Greedy
Accuracy	82.28	82.21	82.92	81.85
Orig				
p values	1	0.408	0.981	**0.022**
Greater-equal-less	0, 54, 0	28, 2, 24	23, 1, 30	35, 2, 17
Biased				
p values	0.595	1	0.992	0.084
Greater-equal-less	24, 2, 28	0, 54, 0	22, 4, 28	31, 2, 21
Ensemble				
p values	**0.020**	**0.009**	1	<**0.001**
Greater-equal-less	30, 1, 23	28, 4, 22	0, 54, 0	46, 1, 7
Greedy				
p values	0.979	0.918	1	1
Greater-equal-less	17, 2, 35	21, 2, 31	7, 1, 46	0, 54, 0

Table 17 Pair-wise comparison of filtering for Ridor

	Orig	Biased	Ensemble	Greedy
Accuracy	79.90	79.16	80.56	79.96
Orig				
p values	1	**0.016**	1.000	0.895
Greater-equal-less	0, 54, 0	33, 2, 19	15, 1, 38	20, 3, 31
Biased				
p values	0.985	1	1	0.998
Greater-equal-less	19, 2, 33	0, 54, 0	7, 3, 44	17, 2, 35
Ensemble				
p values	<**0.001**	<**0.001**	1	<**0.001**
Greater-equal-less	38, 1, 15	44, 3, 7	0, 54, 0	36, 3, 15
Greedy				
p values	0.107	**0.002**	1.000	1
Greater-equal-less	31, 3, 20	35, 2, 17	15, 3, 36	0, 54, 0

Table 18 Pair-wise comparison of filtering for RIPPER

	Orig	Biased	Ensemble	Greedy
Accuracy	80.34	79.98	81.25	80.41
Orig				
p values	1	**0.040**	1	0.704
Greater-equal-less	0, 53, 0	30, 2, 21	11, 2, 40	21, 6, 26
Biased				
p values	0.961	1	1	0.989
Greater-equal-less	21, 2, 30	0, 53, 0	8, 1, 44	19, 4, 30
Ensemble				
p values	<**0.001**	<**0.001**	1	<**0.001**
Greater-equal-less	40, 2, 11	44, 1, 8	0, 53, 0	38, 4, 11
Greedy				
p values	0.300	**0.011**	1	1
Greater-equal-less	26, 6, 21	30, 4, 19	11, 4, 38	0, 53, 0

7 Appendix 2: Ensemble results for each data set

This section provides the results for each data set comparing a voting ensemble with filtering using the ensemble filter for each investigated learning algorithm as well as filtering using the ensemble filter for a voting ensemble. The results comparing a voting ensemble with filtering for each investigated non-ensembled learning algorithm are shown in Table 19. The bold values represent the highest classification accuracy and the rows highlighted in gray are the data sets where filtering with the ensemble filter increased the accuracy over the voting ensemble for all learning algorithms. The results comparing a voting ensemble with a filtered voting ensemble are shown in Table 20. The bold values for the "Ens" column represent if the

Table 19 Comparison of the accuracy for each data set using a voting ensemble (Ens) with using the ensemble filter for the investigated learning algorithms

	Ens	MLP	C4.5	IB5	LWL	NB	NNge	RF	Rid	RIP	Per
anneal	98.08	**98.29**	91.72	92.91	92.72	83.93	92.87	94.8	96.59	94.84	0.33
AP-BU	97.61	96.87	94.87	96.87	93.3	96.72	96.72	**98.01**	93.59	94.73	0.85
ar1	*90.08*	*92.29*	***92.56***	***92.56***	*92.29*	*92.29*	*92.29*	*92.29*	***92.56***	***92.56***	*3.31*
arrhyth	**71.11**	70.13	70.65	59.14	57.67	65.63	65.71	66.59	70.65	71.09	**11.95**
audiolo	**78.94**	78.61	76.99	62.54	47.05	73.01	72.42	73.6	71.24	73.89	7.08
autos	**83.51**	78.54	79.84	64.72	51.71	56.1	74.8	82.6	69.59	76.1	4.88
badges2	**100**	**100**	**100**	**100**	**100**	99.66	**100**	99.89	**100**	**100**	0.00
balance	88.45	**90.35**	78.67	89.65	60.59	89.97	82.56	82.77	79.68	79.09	4.16
breastc	73.99	73.43	**75.17**	74.13	73.31	73.43	74.13	73.19	74.71	74.59	**10.14**
breastw	96.88	**97.00**	95.14	96.76	92.61	95.95	95.99	96.57	95.61	95.8	1.72
bupa	71.3	**71.5**	66.47	62.71	60.29	59.03	65.31	69.28	67.44	68.02	2.61
carEval	96.7	**98.82**	92.09	92.77	70.02	85.22	94.21	92.46	95.72	87.15	0.00
chess	**99.53**	99.41	99.44	96.17	72.15	87.85	98.56	98.77	98.72	99.21	0.03
cm1_req	*75.73*	*76.4*	***77.53***	***77.53***	*76.78*	***77.53***	*77.15*	*76.78*	***77.53***	*76.78*	***16.85***
colic	85.33	**86.41**	85.78	82.97	81.52	83.33	84.42	85.69	84.42	85.96	4.35
contact	76.67	**83.33**	**83.33**	76.39	76.39	76.39	80.56	80.56	79.17	77.78	**12.50**
credita	**86.64**	85.7	85.99	86.62	85.51	81.64	85.6	86.04	85.85	86.09	4.35
creditg	**75.64**	75.07	73.17	73.37	70.03	74.8	73.33	74.2	71.97	72.6	5.20
derma	**97.43**	97.09	93.99	96.08	87.61	97.36	95.36	95.99	94.35	88.8	0.00
desh	74.32	70.78	69.55	65.84	71.6	62.14	67.49	**74.49**	71.6	71.6	7.41
ecoli	**87.44**	86.21	84.72	87.2	65.87	86.9	85.22	85.71	83.73	82.74	4.76
eucalyp	**65.11**	63.32	62.86	55.8	51.04	57.52	56.84	56.88	61.73	63.32	6.66
eye-mov	**64.76**	54.34	63.72	54.93	42.88	44.11	48.13	62.8	54.11	56.04	1.77
glass	74.02	66.04	68.07	66.51	52.18	53.58	71.5	**74.92**	68.85	68.85	5.14
heart-c	83.83	83.39	77.56	83.17	75.69	**83.94**	79.98	81.63	79.65	80.97	3.30
heart-h	82.93	83.22	81.63	**84.69**	80.16	84.47	81.07	81.75	82.54	81.63	5.44
heart-s	82.44	83.09	81.23	81.73	74.44	**84.07**	78.89	83.09	79.01	79.51	3.33
hepatit	83.1	84.3	81.51	84.95	79.35	**85.59**	83.23	84.09	79.14	80.22	3.87
hypo	99.43	94.32	**99.58**	93.3	95.39	95.51	98.75	99.08	99.33	99.45	0.05
iono	**92.99**	89.84	90.98	84.43	83	83.57	90.79	92.78	89.84	90.6	1.71
iris	95.33	**96.00**	94.67	95.33	94	95.56	95.33	94.22	93.56	92.67	0.67
labor	**92.98**	87.72	78.36	89.47	83.63	92.4	87.72	85.96	78.36	82.46	0.00
lungCan	53.75	52.08	**56.25**	47.92	55.21	55.21	**56.25**	52.08	51.04	54.17	**12.50**
lympho	83.24	**83.56**	77.48	83.33	75.68	81.98	77.48	80.63	78.38	78.15	2.03
MagicTe	86.27	86.09	85.58	83.98	76.27	76.08	82.88	**86.43**	84.77	85.29	2.90
nursery	**98.91**	98.78	97	98.1	88.96	90.21	96.97	97.97	95.67	96.73	0.02
ozone	*97.01*	*97.12*	*97.12*	*97.12*	*97.12*	*97.12*	*97.12*	***97.13***	*97.12*	*97.12*	*0.51*
pasture	86.11	77.78	78.7	68.52	**87.04**	75	80.56	76.85	74.07	68.52	2.78
pimaDia	**77.06**	76.56	76.61	75.17	73.26	75.78	75.22	76.22	75.3	75.26	6.77
post-op	*69.78*	***71.11***	***71.11***	***71.11***	***71.11***	***71.11***	***71.11***	***71.11***	***71.11***	***71.11***	***26.67***

Table 19 continued

	Ens	MLP	C4.5	IB5	LWL	NB	NNge	RF	Rid	RIP	Per
pri-tum	48.08	47.79	41.2	45.82	34.42	**48.57**	44.44	45.23	39.82	40.41	**32.15**
segment	**98.00**	96.05	96.62	95.04	78.59	80.69	96.36	97.37	95.83	94.82	0.17
sick	98.45	96.93	**98.51**	96.3	96.55	94.82	96.86	98.16	98.1	98.03	0.13
sonar	81.92	81.89	72.92	**82.53**	74.84	68.27	71.63	79.49	73.24	79.01	0.00
soybean	**94.32**	94.05	91.7	90.14	56.95	92.83	93.02	92.53	90.41	91.85	1.46
T.A.	**57.88**	55.19	51.66	45.25	50.99	49.89	52.98	53.42	43.49	47.9	8.61
titanic	**78.72**	78.66	78.68	78.59	77.9	77.77	78.68	78.68	78.28	78.68	**16.86**
vote	95.82	95.86	95.71	92.8	95.63	90.96	95.4	**96.4**	94.18	95.63	1.61
vowel	**95.54**	92.83	75.81	93.13	35.05	63.54	87.12	94.48	75.93	71.57	0.00
wave	84.21	**85.11**	77.92	79.69	56.93	79.91	82.44	81.7	80.11	79.75	1.34
wine	97.53	**97.75**	93.26	95.88	90.26	97.57	96.25	97.57	91.01	92.51	0.00
yeast	**61.08**	59.43	60.13	59.32	40.7	58.15	59.4	**61.08**	59.74	60.01	**13.68**
zoo	95.25	**95.38**	92.41	94.72	85.48	94.72	94.72	91.75	90.43	86.8	1.98
Acc	84.37	83.40	81.61	80.85	73.48	78.92	81.59	82.94	80.57	80.76	

The column "Per" refers to the percentage of instances that have a $p(\hat{y}_i|x_i)$ greater than or equal to 90 %. The values in italics represent those datasets where filtering with the ensemble filter increased the accuracy over the voting ensemble for all learning algorithms

Table 20 Comparison of the accuracy from a majority voting ensemble trained on unfiltered (Ens) and filtered data (FEns)

Data set	Ens	FEns 50	FEnse 70	FEns 90	FEns Max
anneal.ORIG	**98.08**	97.57	96.26	86.15	97.57
AP-Breast-Uterus	97.61	**97.69**	97.44	96.15	**97.69**
ar1	90.08	90.08	**90.41**	**92.40**	**92.40**
arrhythmia	**71.11**	70.40	69.69	55.58	70.40
audiology	**78.94**	77.52	72.30	48.58	77.52
autos	**83.51**	82.15	73.56	49.66	82.15
badges2	100.00	100.00	100.00	100.00	100.00
balance-scale	**88.45**	86.46	85.28	71.42	86.46
breast-cancer	73.99	73.71	**74.20**	**74.34**	**74.34**
breast-w	**96.88**	96.71	96.62	93.45	96.71
bupa	**71.30**	70.84	68.35	60.52	70.84
carEval	**96.70**	95.51	91.81	70.02	95.51
chess-KRVKP	**99.53**	99.42	99.26	83.94	99.42
cm1-req	75.73	75.06	**77.53**	**77.53**	**77.53**
colic	85.33	**85.54**	**85.98**	81.52	**85.98**
contact-lenses	76.67	**77.50**	**80.00**	70.83	**80.00**
credit-a	**86.64**	86.26	86.03	85.51	86.26
credit-g	**75.64**	74.52	72.84	70.00	74.52

Table 20 continued

Data set	Ens	FEns 50	FEnse 70	FEns 90	FEns Max
dermatology	97.43	97.43	97.27	91.58	97.43
desharnais	**74.32**	73.09	72.84	69.38	73.09
ecoli	87.44	**87.74**	86.90	64.88	**87.74**
eucalyptus	**65.11**	63.97	61.82	52.83	63.97
eye-movements	**64.76**	59.02	55.26	45.21	59.02
glass	**74.02**	62.52	61.59	49.53	62.52
heart-c	**83.83**	82.38	82.18	80.20	82.38
heart-h	82.93	82.86	**83.40**	82.04	**83.40**
heart-statlog	**82.44**	82.37	81.26	78.59	82.37
hepatitis	83.10	**83.35**	83.10	80.26	**83.35**
hypothyroid	**99.43**	99.37	98.17	94.04	99.37
ionosphere	**92.99**	92.82	91.34	84.67	92.82
iris	**95.33**	94.53	94.13	94.13	94.53
labor	**92.98**	91.58	88.07	82.11	91.58
lungCancer	**53.75**	51.25	53.13	38.75	53.13
lymphography	**83.24**	81.35	80.95	76.35	81.35
MagicTelescope	**86.27**	85.49	84.73	74.91	85.49
nursery	**98.91**	98.55	97.24	90.43	98.55
ozone	97.01	**97.07**	**97.09**	**97.12**	**97.12**
pasture	**86.11**	81.11	78.89	66.67	81.11
pimaDiabetes	**77.06**	76.46	75.81	73.91	76.46
post-opPatient	69.78	**70.22**	**71.11**	**71.11**	**71.11**
primary-tumor	**48.08**	45.19	43.01	38.47	45.19
segment	**98.00**	97.62	96.54	85.65	97.62
sick	**98.45**	98.32	98.17	97.16	98.32
sonar	**81.92**	81.44	80.10	73.75	81.44
soybean	**94.32**	93.85	93.12	67.88	93.85
spambase	**94.95**	94.78	94.17	84.43	94.78
teachingAssistant	**57.88**	54.44	47.15	39.60	54.44
titanic	**78.72**	78.65	78.00	77.60	78.65
vote	**95.82**	95.72	95.68	95.40	95.72
vowel	**95.54**	94.75	86.44	40.14	94.75
waveform-5000	84.21	**84.26**	81.77	63.42	**84.26**
wine	97.53	**97.64**	96.97	96.18	**97.64**
yeast	**61.08**	61.01	60.55	40.50	61.01
zoo	**95.25**	94.65	93.66	87.13	94.65
Ave	84.37	83.40	82.21	73.96	83.62

The value after "FEns" represents the percentage of learning algorithms that have to misclassify an instance for it to be filtered from the training set and "Max" uses the accuracy from the percentage that results in the greatest accuracy. Training with unfiltered data is significantly better than training with filtered data. The values in bold for the "Ens" represent if the majority voting ensemble is greater then the filtered majority voting ensemble. The values in bold for the "FEns" columns represent if using filtered training data results in greater classification accuracy

voting ensemble trained on unfiltered data achieves higher accuracy while the bold values for the "FEns" columns represent if the voting ensemble trained on filtered data achieves higher accuracy than the voting ensemble trained on unfiltered data.

References

Bishop CM, Nasrabadi NM (2006) Pattern recognition and machine learning, vol 1. Springer, New York

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

Collobert R, Sinz F, Weston J, Bottou L (2006) Trading convexity for scalability. In: Proceedings of the 23rd international conference on machine learning, pp 201–208

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Dietterich TG (2000) Ensemble methods in machine learning. In: Multiple classifier systems, Lecture Notes in Computer Science, vol 1857. Springer, Berlin, pp 1–15

Freund Y (1990) Boosting a weak learning algorithm by majority. In: Proceedings of the third annual workshop on computational learning theory, pp 202–216

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

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

Ho TK, Basu M (2002) Complexity measures of supervised classification problems. IEEE Trans Pattern Anal Mach Intell 24:289–300

John G.H (1995) Robust decision trees: removing outliers from databases. In: Knowledge discovery and data mining, pp 174–179

Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51(2):181–207

Lawrence ND, Schölkopf B (2001) Estimating a kernel fisher discriminant in the presence of label noise. In: In Proceedings of the 18th international conference on machine learning, pp 306–313

Lee J, Giraud-Carrier C (2011) A metric for unsupervised metalearning. Intell Data Anal 15(6):827–841

Nettleton DF, Orriols-Puig A, Fornells A (2010) A study of the effect of different types of noise on the precision of supervised learning techniques. Artif Intell Rev 33(4):275–306

Ng AY, Jordan MI (2001) On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In: Advances in neural information processing systems, vol 14, pp 841–848

Opitz DW, Maclin R (1999) Popular ensemble methods: an empirical study. J Artif Intell Res 11:169–198

Orriols-Puig A, Macià N, Bernadó-Mansilla E, Ho TK (2009) Documentation for the data complexity library in C++. Tech. Rep. 2009001, La Salle—Universitat Ramon Llull

Peterson AH, Martinez TR (2005) Estimating the potential for combining learning models. In: Proceedings of the ICML Workshop on meta-learning, pp 68–75

Polikar R (2006) Ensemble based systems in decision making. IEEE Circuits Syst Mag 6(3):21–45

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

Rebbapragada U, Brodley CE (2007) Class noise mitigation through instance weighting. In: Proceedings of the 18th European conference on machine learning, pp 708–715

Sáez JA, Luengo J, Herrera F (2013) Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. Pattern Recognit 46(1):355–364

Salojärvi J, Puolamäki K, Simola J, Kovanen L, Kojo I, Kaski S (2005) Inferring relevance from eye movements: feature extraction. Tech. Rep. A82, Helsinki University of Technology

Sayyad Shirabad J, Menzies T (2005) The PROMISE repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Canada . http://promise.site.uottawa.ca/SERepository/

Schapire RE (1990) The strength of weak learnability. Mach Learn 5:197–227

Segata N, Blanzieri E, Cunningham P (2009) A scalable noise reduction technique for large case-based systems. In: Proceedings of the 8th international conference on case-based reasoning: case-based reasoning research and development, pp 328–342

Servedio RA (2003) Smooth boosting and learning with malicious noise. J Mach Learn Res 4:633–648

Smith MR, Martinez T (2011) Improving classification accuracy by identifying and removing instances that should be misclassified. In: Proceedings of the IEEE international joint conference on neural networks, pp 2690–2697

Smith MR, Martinez T (2014) Reducing the effects of detrimental instances. In: Proceedings of the 13th international conference on machine learning and applications, pp 183–188

Smith MR, Martinez T, Giraud-Carrier C (2014) An instance level analysis of data complexity. Mach Learn 95(2):225–256
Stiglic G, Kokol P (2009) GEMLer: gene expression machine learning repository. University of Maribor, Faculty of Health Sciences. http://gemler.fzv.uni-mb.si/
Teng C (2003) Combining noise correction with feature selection. Data warehousing and knowledge discovery, Lecture Notes in Computer Science, vol 2737, pp 340–349
Teng CM (2000) Evaluating noise correction. In: PRICAI, pp 188–198
Thomson K, McQueen RJ (1996) Machine learning applied to fourteen agricultural datasets. Tech. Rep. 96/18, The University of Waikato
Tomek I (1976) An experiment with the edited nearest-neighbor rule. IEEE Trans Syst Man Cybern 6:448–452
Verbaeten S, Van Assche A (2003) Ensemble methods for noise elimination in classification problems. In: Proceedings of the 4th international conference on multiple classifier systems, pp 317–325
Wilson DL (1972) Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans Syst Man Cybern 2–3:408–421
Wilson DR, Martinez TR (2000) Reduction techniques for instance-based learning algorithms. Mach Learn 38(3):257–286
Zeng X, Martinez TR (2001) An algorithm for correcting mislabeled data. Intell Data Anal 5:491–502
Zeng X, Martinez TR (2003) A noise filtering method using neural networks. In: Proceedings of the international workshop of soft computing techniques in instrumentation, measurement and related applications
Zhu X, Wu X (2004) Class noise vs. attribute noise: a quantitative study of their impacts. Artif Intell Rev 22:177–210