Knowledge-Based Systems 215 (2021) 106738

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A reliable ensemble based approach to semi-supervised learning

Sjoerd de Vries [a,b,*], Dirk Thierens [a]

[a] Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
[b] UMC Utrecht, Heidelberglaan 100, 3584 CX Utrecht, The Netherlands

ARTICLE INFO

Article history:
Received 15 July 2020
Received in revised form 18 November 2020
Accepted 29 December 2020
Available online 14 January 2021

Keywords:
Ensemble learning
Out-of-bag error
Ranking
Self-training
Semi-supervised learning
Wrapper

ABSTRACT

Semi-supervised learning (SSL) methods attempt to achieve better classification of unseen data through the use of unlabeled data than can be achieved by learning from the available labeled data alone. Most SSL methods require the user to familiarize themselves with novel, complex concepts and to ensure the underlying assumptions made by these methods match the problem structure, or they risk a decrease in predictive performance. In this paper, we present the reliable semi-supervised ensemble learning (RESSEL) method, which exploits unlabeled data by using it to generate diverse classifiers through self-training and combines these classifiers into an ensemble for prediction. Our method functions as a wrapper around a supervised base classifier and refrains from introducing additional problem dependent assumptions. We conduct experiments on a number of commonly used data sets to prove its merit. The results show RESSEL improves significantly upon the supervised alternatives, provided that the base classifier which is used is able to produce adequate probability-based rankings. It is shown that RESSEL is reliable in that it delivers results comparable to supervised learning methods if this requirement is not met, while the method also broadens the range of good parameter values. Furthermore, RESSEL is demonstrated to outperform existing self-labeled wrapper approaches.

© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Classification models learn a relationship between a number of input features and a corresponding output label. Most commonly, the methods used to solve classification problems are supervised methods, requiring for each example that its label is known. In many settings, however, the amount of labeled data available is limited, hindering the predictive performance of these classifiers. Oftentimes, the labels have to be generated by experts, e.g., by medical doctors when the objective is to predict infections, a process which can be both time-consuming and expensive. One solution to this problem is the use of semi-supervised learning (SSL) methods.

In contrast to their supervised counterparts, SSL methods use both labeled and unlabeled data to learn from. In order to be useful, these methods should improve upon the predictive accuracy attained by supervised methods, and have been shown to do so under certain conditions. This potential has been recognized and a lot of research has been done in this area.

Widespread use of SSL methods is not yet commonplace in large parts of the data science community, however. This can be attributed to a number of reasons:

1. Many SSL methods rely on concepts that are unfamiliar to a practitioner of supervised methods. A thorough study is required to develop an intuition for these SSL methods and it can be difficult to pick from a plethora of available algorithms for someone who is not knowledgeable in the field.
2. In order to learn from unlabeled data, most semi-supervised methods rely on a number of assumptions, relating the method to the problem structure. When these are not met, the corresponding methods may produce poor results. Furthermore, these assumptions can be difficult to check explicitly.
3. SSL methods may not produce robust results. For example, given a certain initial set of labeled data, the self-training method might succeed in increasing predictive performance, whereas with a different initial set it might degrade and propagate its own errors, which makes the method unreliable.
4. There is an absence of easy-to-use implementations of SSL methods in common machine learning packages. This is evidenced by the lack of such methods in the popular Scikit-learn package [1], which has only two SSL models available at the time of writing.

To overcome these difficulties, an interested practitioner is required to conduct a careful study of the SSL field. Consequently,

* Corresponding author at: Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands.
E-mail address: s.devries1@uu.nl (S. de Vries).

https://doi.org/10.1016/j.knosys.2021.106738
0950-7051/© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

many data scientists instead prefer to stick to supervised methods, even in situations where unlabeled data is readily available and there is a lot of potential for improvement.

The main contribution of this paper is to provide data science practitioners who are acquainted primarily with supervised classification, with a reliable method for improving the predictive accuracy achieved by their supervised models without them having to familiarize themselves with the extensive domain of semi-supervised learning and without them having to discard their existing predictors and restart the modeling process from scratch to incorporate the unlabeled data.

To this end, we developed the reliable semi-supervised ensemble learning (RESSEL) method, a wrapper method which combines ensemble learning with self-training and an early-stopping mechanism based on the out-of-bag error. As a wrapper method, RESSEL ensures practitioners can use the base classifiers they are familiar with to learn from unlabeled data. Self-training is used to enrich these classifiers, after which they are combined into an ensemble, reducing the variance and causing the susceptibility of the self-training process to the initial configuration to be averaged out as each classifier will be presented a different training set. Due to the unsure nature of self-training, the base classifiers will be more diverse after being enriched, to the benefit of the ensemble. To prevent adverse learning, an early-stopping mechanism is in place to prevent individual classifiers from deteriorating, by continuously evaluating them on the out-of-bag error. This ensures the self-training process can at most deteriorate a base classifier to the extent that the out-of-bag set is not representative of the data. The bagging method at the basis of ensemble generation, as well as the concept of the out-of-bag error measurements are already commonly used in supervised learning. Self-training is the most straightforward manner of enriching a classifier with unlabeled data and it does not rely on the same difficult to check assumptions that more complicated methods do. RESSEL therefore provides an accessible manner of incorporating unlabeled data into a supervised workflow.

In our experiments, we show RESSEL has the ability to improve upon common supervised classification methods when its base classifier is able to produce good probability-based rankings. Next, we show RESSEL consistently outperforms competing works in the category of self-labeled wrapper methods. Additionally, we show that the prediction method is reliable: it produces results of the same quality as a supervised model even when the proper conditions for its use are not met and the enriched models are less susceptible to bad hyperparameter choices. The different parts that RESSEL consists of are studied in separation and are shown to all contribute meaningfully to the method in its entirety. Additional attention is directed towards a study of the quality of the rankings produced by the base classifiers used in our experiments, which we then relate to their suitability for use with self-training. Furthermore, we confirm RESSEL performs well even when the amount of labeled data available is very limited.

In the remainder of the paper the following topics are discussed in order: In Section 2 an overview is given of the semi-supervised learning and ensemble learning fields, as well as their combination. In Section 3 the RESSEL method is explained in detail. In Section 4 we review the experimental setup used to evaluate RESSEL. In Section 5 the performance of the method as a whole is validated, as well as that of the parts that it consists of. A detailed study is conducted into the importance of the ranking ability of the base learner and the robustness of RESSEL to changes in the hyperparameters of its base classifier. In Section 6 conclusions are drawn and directions for future research are discussed.

2. Background

Historically, semi-supervised learning and ensemble learning have been thoroughly researched. Their combination, however, remains rather unexplored [2,3]. In the following, a succinct overview of the two separate fields as well as their combination is given. Additional emphasis is placed on the bagging and self-training techniques, as they are especially relevant to our proposed method: reliable semi-supervised ensemble learning (RESSEL).

2.1. Semi-supervised learning

In contrast to the supervised learning setting, semi-supervised learning [4,5] methods attempt to learn using labeled as well as unlabeled data. A distinction can be made between three main types of SSL: regression [6], classification [4], and clustering [7]. In this paper we focus our attention to the classification problem.

Two main categories of semi-supervised classification problems can be distinguished [4]: transductive, in which the examples which are to be classified are given beforehand, and inductive, in which a classifier is trained, later to be used on unseen data. In this paper we propose a method for the latter.

In order for inductive SSL techniques to be able to learn from unlabeled data, these techniques usually make assumptions about the problem structure. Some typical assumptions underlying SSL methods are [4]:

1. Smoothness assumption: If two examples in a high-density region are close, so should be their labels. E.g., maximum margin methods such as Transductive SVM [8] and Semi-Supervised SVM [9].
2. Cluster assumption: Examples in the same cluster should have the same label, or equivalently, the decision boundary between two classes should be in a low-density region. This can be seen as a special case of the smoothness assumption, as one definition of a cluster is to require groups of points in such a cluster to be connectable by high-density regions. E.g., generative models such as expectation maximization with Gaussian mixture models [10] and clustering using genetic algorithms [11].
3. Manifold assumption: The data can be projected onto a lower-dimensional manifold. E.g., graph-based methods such as manifold regularization [12] and Spectral Graph Transducer [13].

These assumptions can be difficult to test in practice. As Zhu states in his literature survey of SSL [14]: "Detecting bad match [of problem structure with model assumption] in advance however is hard and remains an open question". Furthermore, if these underlying assumptions are mismatched with the problem structure, the unlabeled data cannot be helpful and if they are met it is difficult to anticipate to what extent SSL will help [15].

There exists another type of semi-supervised techniques which relies on different assumptions, called self-labeling [16]. Self-labeling techniques iteratively extent the labeled data set with unlabeled examples to which they assign a label, assuming that their own predictions tend to be correct. From this final category of self-labeling, the self-training [17] method is used by RESSEL to learn from unlabeled data.

2.1.1. Self-training

Self-training was first applied in the context of word sense disambiguation [17] and is arguably the simplest method of learning from unlabeled data. It is a wrapper method which takes a pre-trained classifier which we shall refer to as the base classifier and tries to enrich it using unlabeled data. Initially, this base classifier

is trained on a training set consisting of a limited number of labeled examples. Thereafter, the classifier iteratively expands its training set, typically by using its own most confident predictions on the unlabeled data set. The unlabeled examples are ranked according to the probability estimates of the base classifier, after which the highest ranked examples are added to the training set with their most probable label.

Self-training does not make additional assumptions on the input data other then those of its base learner, assuming only its own predictions are likely to be true. More specifically, for self-training to function properly, the base classifier it is used with should produce good probability-based rankings [18]. This is in contrast with most of the SSL methods, which rely on assumptions highly dependent on the problem structure. The main requirement for self-training applies to the base classifier used, which is a supervised learner and often much better understood.

Nevertheless, even when a suitable base classifier is used, self-training is not guaranteed to improve upon the base classifier and might even cause a decrease in its predictive performance [19]. Several methods exist which try to improve upon its base performance, such as Self-Training with Editing (SETRED) [20], which tries to identify and remove mislabeled examples, and more recently Self-Training based on Density Peaks (STDP) [21], which uses the popular density peaks clustering method [22] to integrate the structure of the data space into the self-training process.

Being a wrapper method, self-training is difficult to theoretically analyze independent of its base classifier [14]. Existing theoretical analyses are scarce and apply to specific models such as Gaussian Mixture Models [23] and Neural Networks [24]. Furthermore, no general solution has been found to remedy the main factor which might cause self-training to deteriorate predictive performance: the risk of propagating erroneously assigned labels [16]. In this paper, we attempt to mitigate this risk independent of the base classifier by using the complementary properties offered by the ensemble learning paradigm.

2.2. Ensemble learning

A machine learning ensemble, or multiple classifier system, is a collection of learners. These learners are trained using input data and when new examples are classified, their predictions are combined to arrive at a joint prediction. A good ensemble produces predictions with an accuracy higher than that of any of the classifiers it consists of.

Why is classifier combination so profitable? Three reasons are generally accepted [25,26]:

1. Statistical: In absence of enough data, many learners with an equally good training set accuracy might be found. If a single learner is selected from these, it might not be the optimal one. By using multiple classifiers, an average prediction is produced, reducing this risk of picking a single classifier that does not generalize well. This corresponds to a reduction in variance.
2. Computational: In absence of enough time, learners based on some stochastic search mechanism might get stuck in local optima. By combining multiple learners which have converged to different local optima, the true optimum might be better approximated. This corresponds to a reduction in computational variance.
3. Representational: In absence of enough data, or when the base classifier is insufficiently flexible, the range of possible representations of the decision boundary is limited and might not be suited for a problem if the true decision boundary is very complex. By combining multiple decision boundaries, a wider range of boundaries becomes available. This corresponds to reduction in bias.

In order for an ensemble to operate well, its base classifiers should be both accurate and diverse [27,28]. It is easy to see there should be some diversity, as when all classifiers in the ensemble are identical, all their predictions will be as well. The resulting aggregated prediction will thus be the same as that of any of the composite classifiers.

Ensemble systems have been thoroughly studied and shown to perform very well in practice [29]. While implicit manners of introducing diversity are often very successful, attempts to explicitly measure and introduce diversity have generally not been [28,29].

Some of the most well known ensemble methods are Random Subspace [30], Random Forest [31], (Ada) Boosting [32,33], (Stochastic) Gradient Boosting [34,35] and Bagging [36]. In our method, we employ the bagging technique for ensemble creation.

2.2.1. Bagging

The bootstrap aggregation, or bagging, procedure takes a number of bootstrap samples [37] from the data and trains a base learner on each sample. These samples approximate the underlying distribution of the entire data set, while being diverse due to the differences in proportions of the individual data points in each bootstrap sample.

While the diversity among the learners of a bagging ensemble is low relative to that of other ensemble techniques [38], empirical results have proven the technique to be effective in practice, especially in combination with a high variance base learner such as a random tree, illustrated by the success of the Random Forest [31]. Additionally, bagging has the benefit of producing an out-of-bag set for each bootstrap sample [39–41]. This set contains in expectation 37% of the original instances and can be used for error estimation, at no additional cost in terms of setting data apart for this purpose.

2.3. Combination

In this work we combine the paradigms of semi-supervised learning and ensemble learning. One of the first mentions of the combination of these fields was by Roli in 2005 [42]. He states there were few works in the literature at the time and presents possible research directions. Four years later, Zhou [2] presented a theoretical analysis showing the benefits of the combination of the fields in the context of disagreement-based learning. His paper shows that unlabeled data can be used to effectively enhance the diversity of the ensemble members. The combination of these fields was still quite rare and mainly limited to techniques that increased the size of the labeled data set such as Co-Training [43], Tri-Training [44] and semi-supervised boosting methods [45].

In recent years, more methods have adopted the SSL ensemble philosophy. Co-Bagging [46] extends the Co-Training method to the single-view setting by using bagging. In UDEED [47], an ensemble is trained as a whole, using an augmented loss function which encourages the ensemble predictions to be diverse on the unlabeled data. The Semi-Supervised Rotation Forest [48] method extends the supervised Rotation Forest to use semi-supervised local discriminant analysis for feature rotation. With PSEMISEL [49], an ensemble is learned using the random subspace and evolutionary techniques to utilize unlabeled data.

In their survey on ensemble learning, Dong et al. state: “Apart from that, since the incorrectly-labeled samples have negative effects on the performance of models in the label propagation process, it is imperative to develop more effective schemes to reduce negative effects of these samples, which needs us to make more efforts" [50], stressing the need to mitigate possible negative effects in a SSL ensemble context.

Our method, reliable semi-supervised ensemble learning (RESSEL) was constructed to address the two main points raised in this subsection so far:

1. By using self-training, the base learners are enriched with unlabeled data resulting in more diverse learners, to the benefit of the ensemble.
2. By using bagging, the ensemble reduces the variance of the uncertain base learners. Moreover, by monitoring the out-of-bag error during the self-training process, it is ensured the gain in diversity due to self-training is not offset by a loss in accuracy.

Additionally, RESSEL is a wrapper method and can be used in combination with familiar supervised learning algorithms, increasing its easy-of-use. These base learners should, however, have good ranking ability of their probability estimates, a property which we elaborate on in Section 5.3.

Before we dive into RESSEL, we discuss a number of additional related works from the literature. The Robust Semi-Supervised Ensemble Learning (ROSSEL) [51] approach, while it is very similar in name, functions differently in practice. ROSSEL provides robustness in the sense of handling noisy input labels, whereas RESSEL is reliable in the sense that it will not decrease in performance compared to a supervised ensemble, even if non-suitable base learners are chosen or if the hyperparameters used are sub-optimal. Further, at the core of ROSSEL is its label weighting method, which uses a cost function over the pseudo-labels.

In [52], the author combines the self-training, Co-Training and Tri-Training methods into an ensemble. A prediction for a new example is made by application of a maximum-probability voting scheme: if the class membership probability of the most confident of the three methods exceeds a predefined confidence threshold, the example is added with the corresponding label. If none of the individual methods meet this threshold, the majority vote is used to decide the label of the example.

A number of methods have been proposed that attempt to guide the self-training process using the structure of the data. Li et al. [53] employ the concept of local cores, i.e. points of highest density within a cluster. The method finds those clusters in which there are no labeled data points present and labels the local core of the cluster by soliciting an expert or using a co-labeling method. Self-training is then applied to the enhanced labeled data set, with the goal of handling scenarios with very scarce or non-spherical data. Another cluster based method described in [54] combines semi-supervised Fuzzy C-Means clustering with self-training, using a Support Vector Machine as the base classifier, in an iterative process in which only the data points with a high membership degree of belonging to a certain class according to the clustering are eligible for the self-training step. A method which incorporates the structure of the data through a graph-based approach, is Self-training Nearest Neighbor Rule using Cut Edges (SNNRCE) [55]. A relative neighborhood graph is constructed and any samples without any cut edges are labeled. Thereafter, a portion of remaining samples are labeled by self-training with the nearest neighbor method. A statistical test is performed on the newly labeled samples based on the cut edge weight, causing some samples to have their labels adjusted if they exceed a critical value, after which the remainder of the unlabeled examples is classified using the nearest neighbors method.

Tanha et al. construct a self-training based semi-supervised ensemble of Decision Trees (DTs). In their work, the authors improve the ranking ability of the DTs, applying different modifications to the learner to make its probability estimates more reliable.

Finally, two other works used a semi-supervised ensemble specifically with DTs, using the out-of-bag estimate to evaluate performance [56,57]. In contrast to our method, however, their 'Airbag' mechanism is evaluated on an ensemble basis, instead of on the individual trees. In these papers the entire ensemble is enriched in iterations and the training of all base classifiers is halted simultaneously. We evaluate the base learners individually, allowing some classifiers to be further enriched by the unlabeled data while others have already halted the process. Moreover, our method is a wrapper method and suitable for use with more than one base classifier.

Algorithm 1: RESSEL

Input : Labeled data set: L;
Unlabeled data set: U;
Base classifier: C.
Parameters: unlabeled sample fraction u_f.
Output: Ensemble classifier.

$\{C_1, C_2, .., C_k\} \leftarrow$ Duplicate(C)
for $i = 1$ *to* k **do**
  $L_i \leftarrow$ SampleWithReplacement(L)
  $U_i \leftarrow$ SampleWithoutReplacement(U, u_f)
  Calculate complement: $OOB_i \leftarrow L \setminus L_i$
  Calculate class distribution:
  $D_{class_i} \leftarrow$ Distribution(L_i)
  $C_i \leftarrow$ Train C_i on L_i
  $C_i \leftarrow$ RobustSelfTraining(C_i, L_i, U_i, OOB_i, D_{class_i})
end
Ensemble classifier $\leftarrow C_1, C_2, ..., C_k$

3. Reliable semi-supervised ensemble learning

Reliable semi-supervised ensemble learning combines the two paradigms of semi-supervised learning and ensemble learning such that they complement each other. Our method provides the diversity which is much needed by the ensemble through semi-supervised learning, while the ensemble provides the semi-supervised part with robustness, as well as facilitating the early-stopping mechanism. An overview of the RESSEL method can be seen in Fig. 1.

As input RESSEL takes the set $L = \{(X_1, y_1), (X_2, y_2), \ldots, (X_l, y_l)\}$ containing l labeled examples and the set $U = \{(X_{l+1}), (X_{l+2}), \ldots, (X_{l+u})\}$ containing u unlabeled examples, as well as a base classifier C capable of providing probability estimates for its predictions, with chosen values for its hyperparameters.

A step-by-step explanation of the method is given in Algorithm 1. First, the base classifier is duplicated k times (with k the ensemble size) to create a set of classifiers $\{C_1, C_2, \ldots, C_k\}$ which constitutes the homogeneous ensemble. Then, for each of these classifiers C_i, a sample L_i of L and a subset U_i of U are created. L_i is created through sampling with replacement, to obtain a bootstrap sample as is customary in bagging. Each L_i then contains in expectation 63% of the original samples [36]. By taking the complement of L_i in L, the corresponding out-of-bag set OOB_i is obtained, containing on average 37% of the original data. U_i is sampled without replacement, as U is expected to be large compared to L. The fraction of unlabeled data sampled u_f is a tunable parameter. The class distribution D_{class_i} is calculated from the labeled data set L_i and the classifier C_i is trained on L_i as well. Thereafter, this trained base classifier C_i is enriched by the RobustSelfTraining procedure, shown in Algorithm 2.

The RobustSelfTraining method takes a classifier C_i, a set of labeled data L_i, a set of unlabeled data U_i, an out-of-bag set OOB_i and a class distribution D_{class_i}. Furthermore, the number of iterations m for the self-training method, as well as the batch size n are required parameters.

Throughout the self-training process, we maintain the best (lowest) error measured on OOB_i so far, $Error_i^{best}$, in addition to a copy of the classifier for which $Error_i^{best}$ was found, C_i^{best}.

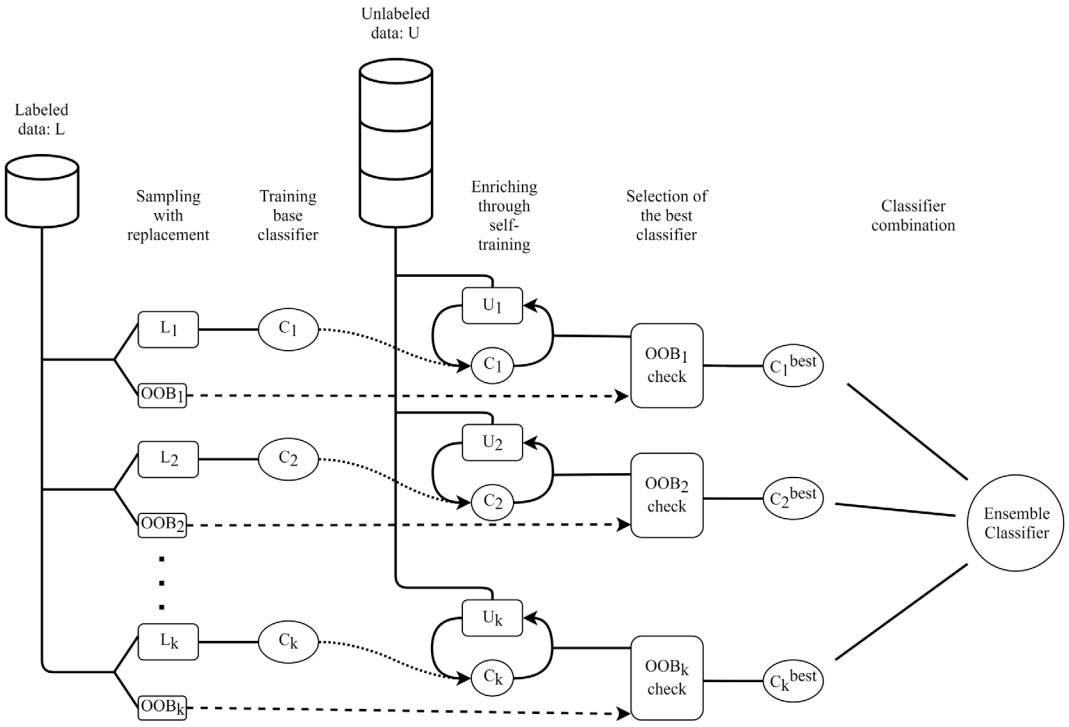

Fig. 1. A schematic overview of the reliable semi-supervised ensemble learning (RESSEL) method.

Algorithm 2: RobustSelfTraining

Input : Trained classifier: C_i; Labeled set: L_i;
Unlabeled set: U_i; OOB set: OOB_i;
Class distribution: D_{class_i}.
Parameters: iterations m; batch size n.
Output: Enriched classifier.

$Error_i^{best} \leftarrow Error(C_i, OOB_i)$
$C_i^{best} \leftarrow C_i$
for $j = 1$ *to* m **do**
  $Prob_i \leftarrow$Predict(C_i, U_i)
  $U_i^{conf} \leftarrow$ ProportionalSelection$(Prob_i, D_{class_i}, n)$
  $L_i \leftarrow L_i \cup U_i^{conf}$
  $C_i \leftarrow$ Train C_i on L_i
  $Error_i^{current} \leftarrow Error(C_i, OOB_i)$
  if $Error_i^{current} < Error_i^{best}$ **then**
    $Error_i^{best} \leftarrow Error_i^{current}$
    $C_i^{best} \leftarrow C_i$
  end
end
Enriched classifier $\leftarrow C_i^{best}$

Then, for the number of iterations m specified, the following procedure is repeated: C_i predicts for each of the examples in U_i its prediction probabilities, $Prob_i$. Then, for each class a ranking is produced based on the probability estimates and the n highest ranked predictions, with corresponding labels, are added to U_i^{conf}. This rank based selection happens in proportion to the original class distribution D_{class_i}. By adding examples based on their rank in proportion to the class distribution, instead of simply looking at the overall ranking in a class independent manner, we ensure not only labels from 'easy' classes are added at the beginning of the learning process. Especially when not all of the unlabeled data in U_i is used, adding the examples according to overall rank might cause 'difficult' classes to be excluded from the learning process. As the ranking is the sole deciding factor on the order in which labels are being added to the base classifier, it is of great importance that the base classifier outputs probability estimates for its predictions from which a good ranking can be produced.

After U_i^{conf} has been generated, C_i is retrained on the union of L_i and U_i^{conf}. Using RESSEL with a base learner which can be incrementally learned would result in a significant speedup, preventing the retraining on L_i at each iteration.

The out-of-bag error $Error_i^{current}$ is measured on the OOB_i set and subsequently used for the early-stopping mechanism. A check is performed to see if the $Error_i^{current}$ is lower than the best out-of-bag error found so far, $Error_i^{best}$, and if it is, $Error_i^{best}$ is updated and the current classifier is copied to replace C_i^{best}.

After the mth iteration, when the self-training procedure is finished, the copy of the classifier at the moment of its best performance, C_i^{best}, is returned. This early-stopping mechanism ensures that the classifier does not suffer from erroneous labels being propagated during the volatile self-training procedure.

The ensemble produces its final prediction through plurality voting: Each of its constituent classifiers predicts a label for the example to be classified and the label which obtained the most votes is ultimately assigned as the ensemble prediction. Ties are resolved randomly.

As RESSEL is a wrapper method, its computational complexity varies based on the base learner it is used with. Therefore, we provide the formulas for the computational complexity of the RESSEL method, given a base learner of known computational complexity with respect to the number of data points it is trained on. We assume both the number of self-training iterations m and the corresponding batch size n to be $\lfloor\sqrt{|U| \cdot u_f}\rfloor$. These values reflect the standard settings for the RESSEL method and will be further explained in Section 4.1.

A distinction can be made between base learners which can be trained incrementally and those which cannot:

1. If a base leaner can be trained incrementally, the total training time required for the RESSEL method depends

on the initial training time of the base classifier on x data points, denoted Init(x), as well as the incremental update step when adding x data points, denoted Incr(x). The total training time of the system will then be $k{\cdot}[\text{Init}(l){+}\ m{\cdot}\text{Incr}(n)]$ or equivalently $k{\cdot}\ [\text{Init}(|L|)\ +\ \lfloor\sqrt{|U|\cdot u_f}\rfloor{\cdot}\ \text{Incr}(\lfloor\sqrt{|U|\cdot u_f}\rfloor)]$.

2. A base learner which cannot be trained incrementally will have to be retrained on all of the data added in the previous steps of the self-training process for every following self-training iteration, taking Init(x) time each iteration. This adds a significant amount of additional time to the process and the total training time of the ensemble classifier will be $k{\cdot}[\Sigma_{i=0}^{m}\ \text{Init}\ (l{+}i{\cdot}n)]$ or equivalently $k{\cdot}[\Sigma_{i=0}^{\lfloor\sqrt{|U|\cdot u_f}\rfloor}\ \text{Init}(|L|{+}\ i\cdot\lfloor\sqrt{|U|\cdot u_f}\rfloor)]$

4. Experiments

We conducted a number of different experiments to gain insight into the performance of RESSEL. The main experiment serves to evaluate the effectiveness of the method when it comes to learning from unlabeled data. We compare our method to a number of supervised alternatives, to confirm it is able to increase predictive accuracy using these data. Then we repeat the experiment using high-dimensional data sets, to investigate if the performance of the method extends to this more complex setting.

Next, we compare RESSEL to a number of existing self-labeled wrapper approaches. In their work on self-labeled techniques for SSL from 2013 [16], Triguero et al. conduct a thorough empirical comparison of a large number of representative self-labeled methods. Their results showed the Tri-Training [44] and Co-Bagging [46] methods had the best average performance of all of the wrapper methods in the inductive phase. In our comparison, we thus compare to these two methods, as well as to the newer Self-Training based on Density Peaks (STDP) [21] method, as it is a representative method of the field in recent years.

Thereafter, we study the three components integrated into RESSEL: (1) self-training; (2) early-stopping; and (3) bagging, in separation. We investigate the difference between the RESSEL method using probability estimates for producing ranks and it using only the actual, predicted labels, to measure the effectiveness of self-training. We conduct further experiments to gain insight into the assumption that RESSEL needs a base learner which produces good rankings. Then, we examine the overall benefit of the early-stopping mechanism and further illustrate its effect by focusing on individual data sets for the SVM classifier with 10% of the data available having labels. Furthermore, we analyze the effect ensemble learning has on RESSEL by varying the ensemble size, as well as comparing the effect bagging has when self-training is used to when it is not.

Subsequently, we investigate if RESSEL can be used when the amount of labeled examples available is very limited. We measure its performance relative to the same supervised methods that were used before, as the number of labeled examples is increased from ten to one hundred.

In our final experiment, we investigate the robustness of RESSEL to different hyperparameter choices.

4.1. Parameters

Both the RESSEL method and the base classifiers that were used in the experiments require a number of parameters to be set. We discuss the default values that we used in this section and mention deviations from the defaults on an experiment to experiment basis.

In our experiments, we used five classifiers implemented by the Scikit-learn package [1]. These classifiers along with the default values we used for their parameters are shown in Table 1. In the following we explain our choices:

Table 1
Default classifier parameter values.

Classifier (parameter)	Value
GNB	
All	default
SVM	
Kernel	rbf
C	1.0
γ	scale
KNN	
Neighbors	10
RDT	
Depth	4
Features	$\sqrt{n_{features}}$
LR	
Implementation	SGD
Loss	log

1. Gaussian Naive Bayes (GNB) does not have many consequential parameters, so we opted to use the default settings.
2. The Support Vector Machine (SVM) has a number of parameters to consider. The most important are the type of kernel, the regularization parameter C and the kernel coefficient γ. In our experiments we used the default values. In Section 5.7 we study how these parameters affect a bagging ensemble and RESSEL with SVM as their base classifier.
3. For the K-Nearest Neighbor (KNN) algorithm, the number of neighbors is the most important parameter. By default, it is set to 5. We set it to 10, as this increases the generalization capability of the algorithm and more importantly, its probability estimates become more meaningful with an increased number of neighbors: the probability estimate of an instance belonging to a certain class is the fraction of nearest neighbors of that instance that belong to that class. By doubling the number of neighbors from 5 to 10, the number of possible different probability estimate values increases from 6 to 11.
4. For the Decision Tree (DT) algorithm, we adjusted the tree depth and the number of features considered at each split. We set the depth to 4, instead of using the default value of not restricting the depth at all. This should prevent the individual trees from overfitting the training data, as well as increasing the reliability of their probability estimates: the final probability estimate is determined by the proportion of the class labels in the leaves of the tree, thus when there are more instances captured in a leave by restricting the amount of leaves implicitly by restricting the tree depth, these estimates are based on a larger number of samples. We set the number of features considered for each split to be $\sqrt{n_{features}}$, to make the tree into a Random Decision Tree (RDT), as these have been shown to perform well in an ensemble context, known as a Random Forest [31].
5. Scikit-learn offers a couple of different implementations for Logistic Regression (LR). We used the Stochastic Gradient Descent (SGD) method, with its loss parameter set to log loss. This way, the classifier is actually a LR model, trained with SGD. We preferred this implementation over the default LR classifier from Scikit-learn, as the latter suffered from widely varying convergence rates for the different data sets we used.

In addition to the parameters of the base classifiers, the RESSEL method has a number of tunable parameters, as explained in Section 3. The default settings used in the experiments, can be seen in Table 2. These default parameter choices are explained in the following.

Table 2
Default experiment parameter values.

Setting	Value
Labeled fraction	[0.05, 0.1, 0.2]
Unlabeled fraction	0.35
Ensemble size	25
Bootstrap	true
Stratify	false

To simulate different settings of semi-supervised learning, we looked at the data sets with 5%, 10%, and 20% of the data as labeled data and the remainder as unlabeled data. Our expectation is that as more data becomes available, the potential benefits of semi-supervised learning and in extension RESSEL will become smaller.

We set the fraction of the unlabeled data presented to each different base classifier to be 0.35. The rationale behind this is to strike a balance between having a large amount of unlabeled data to learn from, while introducing additional diversity by not presenting the same portion of unlabeled data to each classifier in the ensemble. Learning from different subsets of the unlabeled data instead of the whole set can be effective, even for a single classifier, as the subsets are thought to be more representative of the underlying distribution of the data [43]. When 5% of the data is labeled, the fraction of the labeled data used to the unlabeled data used will be on average:

$$\frac{0.05 \cdot 0.63}{0.95 \cdot 0.35} \simeq \frac{1}{10.6}, \tag{1}$$

while in case of 20% labeled data used, this will still be:

$$\frac{0.2 \cdot 0.63}{0.8 \cdot 0.35} \simeq \frac{1}{2.2}. \tag{2}$$

In our experiments, we fixed both the number of self-training iterations m and the corresponding batch size n to be $\lfloor\sqrt{|U| \cdot u_f}\rfloor$. This ensures nearly all of the unlabeled data made available to a classifier is used in the self-training procedure. Experiments with different values of the batch size and number of iterations showed the method was not very sensitive to these parameters, as long as a large part of the unlabeled data was used and the batch size was not overly large, thereby limiting the number of iterations.

We set the ensemble size to be 25, as the test set accuracy had for the most part converged at that number of classifiers. Results confirming this are presented in Section 5.5. We set bootstrap to true, meaning the labeled sets L_i were sampled with replacement, as is standard in the bagging procedure. RESSEL allows for the option to generate the bootstrap samples in a stratified manner, but we set this option to false, as our experiments did not show this to improve over using standard bootstrapping.

4.2. Data sets and preprocessing

In our experiments we used a number of data sets from the University of California Irvine (UCI) [60] and Knowledge Extraction based on Evolutionary Learning (KEEL) [61] repositories.

The data sets which were used can be observed in Table 3 and were picked based on a number of criteria:

1. Minimum size. At the 5% labeled fraction setting with 75% of the data as train data, the fraction of data in the labeled and out-of-bag sets is $0.75 \cdot 0.05 \cdot 0.63 \simeq 0.024$ and $0.75 \cdot 0.05 \cdot 0.37 \simeq 0.014$ respectively. For the smallest data set we used, Australian Credit with 690 entries, this resulted in approximately 16 unique instances per labeled set and 10 unique instances in the out-of-bag set for each base classifier, which is already quite limited.
2. Maximum size. Ensemble learning and self-training with classifiers that cannot be trained incrementally are both computationally expensive methods, extending the training time required by the base classifier as shown in Section 3. Repeated experiments with RESSEL can therefore become quite time-consuming with large data sets.
3. Problem diversity. We included a mix of binary and multi-class problems, searched for different class balances and a different number of (transformed) variables.

To enable each of the algorithms to learn adequately from these data sets, we preprocessed them in a consistent matter:

In case of the UCI data sets, we took recommendations from [59] on how to interpret and/or improve the quality of some of these data sets. The KEEL data sets had been more consistently formatted, and thus required fewer such semantic improvements.

In preparing these data sets for use by multiple kinds of algorithms we made a distinction between the following variable types, processing each differently:

1. Binary: Binarized
2. Ordinal categorical: OrdinalEncoded, Standardized
3. Nominal categorical: OneHotEncoded
4. Numerical: Standardized

The outcome variables were kept as unique categories.

4.3. Validation

In our experiments, we generated a different train-test split 100 times, using 25% of the data for testing and 75% for training. Experiments were conducted with 5, 10 and 20% of the train data as labeled data and the remainder as the unlabeled data, to simulate different semi-supervised settings.

When comparing two methods with each other over multiple data sets, we use the non-parametric Wilcoxon signed rank test to measure statistical significance [62].

To account for the multiple testing problem in comparing multiple methods to each other, over multiple data sets, we use a different method. We opted for a non-parametric test instead of a parametric test like repeated-measures ANOVA, as assumptions of normality ans sphericity cannot be guaranteed [62]. We heed the advice from García et al. [63] to use not the Friedman Test, but instead the Friedman Aligned-Ranks Test, as the number of algorithms compared in our two main experiments in Sections 5.1 and 5.2 is small (five and four respectively). When a statistical difference is found, we apply the Finner method for post-hoc analysis of the results in a pairwise fashion, compared to RESSEL as the control method. For all statistical comparisons we assume a significance level of $\alpha = 0.05$.

5. Results

In what follows, we present the results from the experiments we conducted to evaluate the performance of RESSEL in a variety of settings.

5.1. Comparison with supervised methods

We investigate if RESSEL succeeds in learning from unlabeled data by comparing the method to four alternatives:

1. Single classifier (SinClf): a single base classifier, trained on all of the labeled data;
2. Simple ensemble (SimEns): an ensemble of classifiers which are all trained on all of the labeled data;

Table 3
Data set characteristics.

Data set	Size	Variables	Transformed	Classes	Class balance (%)
Abalone[a]	4177	8	10	3	35/34/32
Australian Credit	690	14	39	2	56/44
Car Evaluation	1728	6	6	4	70/22/4/4
Contraceptive Method Choice	1473	9	9	3	43/35/23
German	1000	20	52	2	70/30
Nursery	12960	8	10	5	33/33/31/3/0
Pima Indian Diabetes	768	8	8	2	65/35
Red wine [58][b]	1599	11	11	2	53/47
Solar Flare	1066	11	37	6	31/22/20/14/9/4
Spambase	4597	57	57	2	61/39
Titanic	2201	3	6	2	68/32
Vehicle Silhouettes	846	18	18	4	26/26/25/24
Vowel	990	13	13	11	9/9/9/9/9/9/9/9/9/9/9
White wine [58][b]	4898	11	11	2	67/33
Yeast	1484	8	8	10	31/29/16/11/3/3/2/2/1/0

[a]Transformed into a classification problem with 3 ring (age) outcome classes: <9, $9-10$, >10, as suggested by [59].
[b]Transformed into the binary problem of grading wines with outcome either <6 or ≥ 6.

3. Bagging: an ensemble for which the individual classifiers are trained on bootstrap samples;
4. RESSEL$^-$: the same as RESSEL itself, but without the early-stopping mechanism.

RESSEL$^-$ thus returns the classifier after the last iteration of the self-training process, instead of the classifier after the iteration of self-training where it had the lowest out-of-bag error. This method is actually semi-supervised, but is included here as it functions as a baseline for RESSEL to improve upon.

The results of this comparison over all of the aforementioned data sets in terms of mean classifier accuracy, for three different setting of the amount of labeled data available and five different base classifiers for a total of fifteen settings, are presented in Table 4.

We observe RESSEL significantly outperforms the single classifier trained on all available labeled data in each of the fifteen settings.

The results show that for the GNB, SVM and KNN base classifiers, the performance of the simple ensemble is exactly that of the single classifier. These methods are deterministic and an ensemble learned on all training data will consequently consist of identical single classifiers, providing no additional benefit. RESSEL is therefore found to significantly outperform these simple ensembles, as was the case for the single classifier.

Random decision trees, as well as logistic regression trained with stochastic gradient descent, are stochastic by nature. Therefore training the same base classifier on identical data a number of times yields different classifiers, that prove effective in an ensemble. This explains why we observe a large increase in mean accuracy from the single classifier for simple ensembles with these base classifiers and not all setting are found to perform significantly worse than RESSEL.

Ensemble generation through bootstrapping proves to provide the necessary diversity to the deterministic classifiers to cause improvement over the single classifier and the simple ensemble, as can be observed for each of these settings. For RDT and LR the effect of bootstrapping is more difficult to isolate in the presence of non-determinism. Overall, the results suggest the bagging ensemble performs better on average, however, than the simple ensemble, with only a slight accuracy loss observed for the RDT_5 setting.

We conclude that the bagging ensemble is the most effective of the supervised methods in our experiments, and therefore focus our attention to the comparison of RESSEL to the bagging ensemble.

It can be seen that on average RESSEL achieves better results than bagging for each of the different settings. These differences are found to be statistically significant when SVM and KNN are the base classifiers, for all settings except for SVM_{20}. These base classifiers meet the assumption of providing good rankings, needed for RESSEL to do well, which we show in Section 5.3. The behavior of RESSEL with these base classifiers as the proportion of labeled data increases from 5% to 20% is as expected: the largest improvement is found when little labeled data is available and as more labeled data is made available, a smaller improvement is observed. As the proportion of labeled data approaches 100%, the potential increase in predictive accuracy of SSL methods compared to supervised methods converges to zero. This explains why for the SVM_{20} setting, the difference is no longer found to be significant.

When GNB is used as the base classifier, RESSEL improves in average accuracy, yet the differences are not found to be statistically significant. It can be seen from the supervised methods that the GNB classifier does not benefit nearly as much from adding more labeled data than the other methods do, improving by only 1.32 on average when data is increased fourfold from 5% to 20% for the single classifier. In contrast, SVM and KNN improve by 6.91 and 6.63 respectively. Given that GNB improves comparatively little from having more labeled data to learn from, it is expected that the improvements due to learning from unlabeled data will be smaller as well.

The results show that there is some improvement of RESSEL over the bagging ensembles with RDT and LR as base learners. This difference is only found to be significant for the LR_{20} setting,

Table 4
Supervised baseline comparison. The mean accuracy of the methods over all data sets under different settings, consisting of a base classifier and a labeled data percentage, is displayed. Results which are found to be statistically significantly different from RESSEL are printed in bold. The percentage of labeled data is denoted by the subscript.

Setting	SinClf	SimEns	Bagging	RESSEL$^-$	RESSEL
GNB_5	**60.15**	**60.15**	61.55	**57.80**	62.31
GNB_{10}	**61.00**	**61.00**	61.97	**59.41**	62.73
GNB_{20}	**61.47**	**61.47**	62.37	**60.53**	62.82
SVM_5	**67.75**	**67.75**	**68.08**	**68.13**	68.88
SVM_{10}	**71.63**	**71.63**	**71.80**	**71.95**	72.29
SVM_{20}	**74.66**	**74.66**	74.78	74.81	74.97
KNN_5	**63.59**	**63.59**	**63.73**	64.76	65.19
KNN_{10}	**67.38**	**67.38**	**67.56**	**67.83**	68.23
KNN_{20}	**70.22**	**70.22**	**70.42**	70.67	70.82
RDT_5	**61.07**	67.54	67.53	**65.42**	67.58
RDT_{10}	**63.06**	69.89	70.16	**68.82**	70.48
RDT_{20}	**64.77**	71.34	71.73	70.99	71.96
LR_5	**64.65**	**67.74**	68.09	**67.81**	68.39
LR_{10}	**66.39**	70.13	70.51	70.17	70.78
LR_{20}	**68.20**	**72.05**	**72.25**	**72.08**	72.65

Table 5
High-dimensional data set characteristics.

Data set	Size	Variables	Classes	Class balance (%)
Internet Ads[a]	3279	1558	2	86/14
Madelon	2600	500	2	50/50
Malware	6248	244	2	90/10
Mice Protein Expression[a]	1080	77	8	14/14/13/13/13/13/13/10
Musk	6598	166	2	85/15
QSAR Androgen Receptor	1687	1024	2	88/12
SECOM[a]	1567	474	2	93/7

[a]These data sets contained missing values. All missing values were imputed using the median value of the non-missing data for the corresponding variables.

however. If anything we would expect the method to perform the best in the setting where relatively few labeled examples are used to train the base learners. Why these methods do not seem to benefit nearly as much from RESSEL as the other base classifiers can be explained by the further study of the self-training and early-stopping components of RESSEL, which follow in Sections 5.3 and 5.4.

The performance of RESSEL compared to RESSEL$^-$ is discussed in detail in Section 5.4.

5.1.1. High-dimensional data sets

To validate that the previous results extend to high-dimensional data sets, we repeated the experiment using seven new data sets. These data sets each contain more variables than the data sets from Table 3, ranging from 77 to 1558 variables per set. A description of these data sets is given in Table 5.

The results of the repeated experiment on these high-dimensional data sets are shown in Table 6. We observe the results follow a similar pattern to the results found in the previous experiment shown in Table 4: Averaged over the seven data sets, RESSEL outperforms the single classifier, the simple ensemble and the bagging ensemble for every setting, with the exception of the simple ensemble for RDT_5 and RDT_{10}. The difference in performance is found to be statistically significant for the single classifier for all settings except for GNB_5 and KNN_5, although the average accuracy of RESSEL is shown to be much higher for these settings as well. For the GNB, KNN and SVM base classifiers, results for the simple ensemble are identical to those of the single classifier, as is to be expected, while the simple ensembles consisting of RDT and LR manage to improve upon their single classifier variants, as before. The statistically significant differences between RESSEL and bagging are found for the SVM and KNN settings, as before, with KNN_5 an exception. The comparison of RESSEL to RESSEL$^-$ is reserved for Section 5.4.

From these observations we conclude the performance of RESSEL extends to data sets of higher dimensionality.

5.2. Comparison with semi-supervised wrapper methods

In this section, we compare RESSEL to some of the best-performing semi-supervised wrapper methods from the literature: Co-Bagging, Tri-Training and Self-Training based on Density Peaks (STDP). The results of this experiment are shown in Table 7.

We observe that averaged over all 15 data sets, RESSEL manages to outperform each of the other methods, for every setting. The difference is found to be statistically significant for every setting when compared to Co-Bagging and STDP, as well as for Tri-Training for the RDT, LR, SVM_5 and KNN_5 settings.

We further investigate the performance of the different algorithms by calculating the average difference in accuracy compared to the base classifier used, trained on only the available labeled data (SinClf in Table 4), as well as the standard deviation of this difference over all data sets. The results are shown in Table 8. We observe that all methods except for RESSEL fail to consistently improve upon the supervised baseline.

From the reported standard deviation, we observe an ordering in the variability of the increased predictive accuracy over the base classifier of the methods for the GNB, SVM and KNN settings: Co-Bagging > STDP > RESSEL > Tri-Training. Tri-training thus produces the least variable results when compared to its base classifier, followed by RESSEL. For the RDT and LR base classifier, Tri-training remains the least variable method, while the results become a lot more variable for RESSEL especially. This added variability is in part due to the stochastic nature of these base classifiers. Additionally, we observe that this added variability is

Table 6
Supervised baseline comparison on high-dimensional data sets. The mean accuracy of the methods over all data sets under different settings, consisting of a base classifier and a labeled data percentage, is displayed. Results which are found to be statistically significantly different from RESSEL are printed in bold. The percentage of labeled data is denoted by the subscript.

Setting	SinClf	SimEns	Bagging	RESSEL$^-$	RESSEL
GNB_5	76.05	76.05	76.45	**72.64**	77.80
GNB_{10}	**75.07**	**75.07**	78.49	**72.29**	79.09
GNB_{20}	**72.59**	**72.59**	76.25	**70.96**	76.59
SVM_5	**80.17**	**80.17**	**80.35**	82.58	82.70
SVM_{10}	**84.33**	**84.33**	**84.37**	85.95	85.97
SVM_{20}	**87.67**	**87.67**	**87.57**	88.45	88.38
KNN_5	77.78	77.78	77.84	79.22	79.14
KNN_{10}	**80.41**	**80.42**	**80.44**	81.69	81.55
KNN_{20}	**83.45**	**83.45**	**83.53**	84.42	84.36
RDT_5	**76.23**	81.05	80.21	78.45	80.39
RDT_{10}	**78.06**	83.15	82.51	**81.01**	83.11
RDT_{20}	**79.87**	84.97	84.51	83.73	85.83
LR_5	**78.41**	79.75	79.24	82.95	83.19
LR_{10}	**83.66**	84.78	84.43	85.72	86.00
LR_{20}	**86.69**	87.31	87.55	87.52	87.74

Table 7
Semi-supervised method comparison. The mean accuracy of the methods over all data sets under different settings, consisting of a base classifier and a labeled data percentage, is displayed. Results which are found to be statistically significantly different from RESSEL are printed in bold. The percentage of labeled data is denoted by the subscript.

	Co-Bagging	Tri-Training	STDP	RESSEL
GNB_5	**55.46**	60.08	**54.81**	62.31
GNB_{10}	**56.69**	60.76	**55.83**	62.73
GNB_{20}	**57.86**	61.40	**57.36**	62.82
SVM_5	**66.28**	**67.39**	**66.27**	68.88
SVM_{10}	**70.12**	71.58	**70.49**	72.29
SVM_{20}	**73.66**	74.90	**74.05**	74.97
KNN_5	**62.61**	**63.28**	**60.11**	65.19
KNN_{10}	**65.30**	67.35	**64.52**	68.23
KNN_{20}	**68.57**	70.22	**68.40**	70.82
RDT_5	**64.44**	**63.80**	**58.74**	67.58
RDT_{10}	**66.96**	**66.20**	**60.61**	70.48
RDT_{20}	**69.26**	**68.01**	**63.10**	71.96
LR_5	**66.29**	**66.14**	**64.92**	68.39
LR_{10}	**68.88**	**68.55**	**67.45**	70.78
LR_{20}	**70.91**	**70.17**	**69.24**	72.65

Table 8
Difference to base classifier. The difference in mean accuracy of the methods to the base classifier, as well as the standard deviation (shown in parentheses) over the data sets and the number of data sets for which the average difference was positive (imp) is displayed. Each setting consists of a base classifier and a labeled data percentage, with this percentage of labeled data denoted by the subscript.

	Co-Bagging		Tri-Training		STDP		RESSEL	
	mean (std)	imp	mean (std)	imp	mean (std)	imp	mean (std)	imp
GNB_5	−4.69 (5.38)	3	−0.07 (0.98)	6	−5.34 (4.56)	1	2.16 (3.07)	10
GNB_{10}	−4.31 (5.48)	2	−0.23 (0.53)	4	−5.16 (4.91)	2	1.73 (2.35)	11
GNB_{20}	−3.62 (5.05)	3	−0.07 (1.17)	9	−4.11 (4.15)	2	1.35 (2.06)	11
SVM_5	−1.47 (2.85)	4	−0.36 (0.52)	4	−1.48 (1.31)	0	1.13 (1.25)	12
SVM_{10}	−1.51 (1.68)	3	−0.04 (0.28)	7	−1.13 (1.12)	1	0.66 (0.83)	14
SVM_{20}	−1.00 (1.44)	4	0.24 (0.22)	12	−0.61 (0.70)	2	0.30 (0.47)	10
KNN_5	−0.99 (2.68)	4	−0.31 (0.61)	6	−3.48 (2.77)	0	1.60 (1.50)	13
KNN_{10}	−2.09 (1.97)	3	−0.03 (0.34)	6	−2.87 (2.29)	0	0.85 (0.64)	14
KNN_{20}	−1.65 (1.76)	2	−0.00 (0.42)	8	−1.82 (1.51)	2	0.60 (0.55)	14
RDT_5	3.38 (2.75)	14	2.73 (1.22)	15	−2.33 (3.22)	4	6.51 (3.05)	15
RDT_{10}	3.90 (3.41)	12	3.14 (1.44)	15	−2.45 (2.92)	2	7.43 (3.70)	15
RDT_{20}	4.48 (3.67)	14	3.24 (1.58)	15	−1.68 (2.15)	3	7.18 (4.10)	15
LR_5	1.64 (2.34)	11	1.49 (1.02)	15	0.27 (1.25)	9	3.74 (1.70)	15
LR_{10}	2.49 (2.54)	12	2.16 (1.21)	15	1.06 (1.74)	12	4.40 (1.90)	15
LR_{20}	2.72 (2.09)	14	1.98 (0.78)	15	1.04 (1.20)	14	4.46 (2.23)	15

paired with much greater improvement over the base classifier for these settings.

The number of data sets for which an improvement was found on average, so a positive difference between the method and SinClf, is shown in the columns named 'imp' for improvement. We observe RESSEL improves the base classifier for more data sets than Co-Bagging and STDP for all settings. RESSEL improves the base classifier for as many or more data sets as Tri-Training for each of the settings, with the notable exception of SVM_{20}, where Tri-Training improves on 12/15 data sets and RESSEL on only 10. These findings confirm RESSEL to be a reliable wrapper method, whose ability to consistently improve upon a supervised alternative is not as dependent on the data set as it is for Co-Bagging, Tri-Training and STDP.

5.3. Effectiveness of self-training

In order to understand what makes RESSEL able to learn from unlabeled data, we focus our attention to the component which introduces SSL into the method: self-training.

For self-training to be effective, a base classifier needs to be able to provide itself with a measure of confidence in its predictions, which is used to rank instances in terms of class membership probability. To measure the extent to which the base classifiers meet this assumption in conjunction with RESSEL, we adjusted the method in which labeled examples are added. Instead of adding the examples with the highest ranks for each class to the labeled set, instances are selected randomly from all examples which are predicted to have a certain label, irregardless of their rank. In each iteration of the process, the class proportions are still preserved as before. We denote this variant of RESSEL the 'No rank' version, while the original is called 'Rank Based'.

The results of this experiment are shown in Table 9. For the GNB, SVM and KNN base classifiers, we observe that there is a noticeable improvement when using the probability estimate based ranking for selection of data points to add to the labeled set. This difference is found to be statistically significant for SVM and KNN for the 5% and 10% labeled data settings. The difference decreases as the amount of labeled data increases, as is expected. These results are consistent with the results from Table 4 and prove that RESSEL learns from and improves by using unlabeled data when these base learners rank examples based on their probability estimates.

Furthermore, we observe that the differences are much smaller for RDT and LR and that they do not follow the expected pattern of the size of the differences decreasing with increasing labeled data size. We conclude that these methods do not benefit from using their own probability estimates to impose a ranking on the unlabeled data and thus they are not suitable for use with self-training and in extension RESSEL.

Table 9
Effectiveness of self-training. The mean accuracy (difference) over all data sets is shown. Differences which are found to be statistically significantly different are printed in bold. The percentage of labeled data is denoted by the subscript.

Setting	No rank	Rank based	Difference
GNB_5	61.43	62.31	0.88
GNB_{10}	62.25	62.73	0.48
GNB_{20}	62.50	62.82	0.32
SVM_5	67.75	68.88	**1.13**
SVM_{10}	71.62	72.29	**0.67**
SVM_{20}	74.88	74.97	0.09
KNN_5	63.71	65.19	**1.48**
KNN_{10}	67.52	68.23	**0.71**
KNN_{20}	70.58	70.82	0.24
RDT_5	67.46	67.58	0.12
RDT_{10}	70.16	70.48	**0.32**
RDT_{20}	71.86	71.96	0.10
LR_5	68.19	68.39	0.20
LR_{10}	70.84	70.78	−0.06
LR_{20}	72.60	72.65	0.05

In case of RDT, these results are in line with our expectations. Decision trees are known to provide poor probability estimates [64,65] and earlier studies have been conducted to improve these estimates and make them suitable for use with self-training [18,57]. The difference observed in the RDT_{10} setting is found to be statistically significant, still its value is rather small and the fact that the differences for the 5% and 20% settings are not found to be significant, lead us to believe this to be due to chance.

For the LR classifier, the results are not as we initially suspected, as it is generally considered to have well calibrated probability estimates. In the following, we investigate further and show how well calibrated probability estimates do not guarantee good ranking quality and how this causes RESSEL to work well with one base learner, but not with another.

5.3.1. Assumption: Ranking quality

As stated in Section 2.1.1, for self-training to be effective it is important that the base classifier to which it is applied has good ranking ability. Ordinarily, a classifier ranks unlabeled instances based on its probability estimates.

The term 'good probability estimates' is often used to describe well calibrated probability estimates, i.e., when a classifier outputs a class probability estimate of 0.8, it is expected to correctly

predict 80% of such cases. Logistic Regression is known to be a well calibrated classifier. Although it seems logical a classifier with good probability estimates would be a good ranker, this is not necessarily true.

We searched the literature for research into the ranking performance of the base classifiers we used in our experiments. GNB is generally found to be a good ranker [66–68]. In a comparison of GNB, SVM and DTs, SVM is found to perform similarly to GNB in ranking and GNB and SVM both perform much better than DTs [66]. A lot of research has been conducted into DTs and it is generally accepted that they provide poor probability estimates and have poor ranking capability [64,69]. In order to work well with self-training, a number of adjustments have been proposed to turn them into better rankers [18]. The KNN method is found to have decent ranking capabilities, even though its probability estimates are questionable [70]. As mentioned before, LR is generally considered a good probability estimator, yet not much work is devoted to its ability as a ranker.

In these works, the quality of ranking is commonly evaluated by the Area Under the Receiver Operating Characteristics Curve (AUC). The reason for using the AUC is that it has been shown in the binary classification context to be the probability of a positively labeled instance to be ranked above a negatively labeled instance and to be equivalent to the Wilcoxon statistic [71]. It is thus a natural measure of ranking capability.

To understand the behavior of LR compared to the other base classifiers used with RESSEL, we conducted two experiments. For these experiments we generated synthetic data, with two classes distributed according to multivariate normal distributions with a variance of 2. The centers of these Gaussians where set to be -1 and +1 in their first variable, while set to 0 for the remaining dimensions.

In the first experiment, we measured the AUC and the accuracy for each of the classifiers, as the problem dimensionality was increased from 2 to 50 dimensions. Each classifier might be differently suited to the problem, as measured by the accuracy of the classifier. To account for problem suitability and in order to focus purely on the ranking ability, we divided the AUC by the accuracy of the classifier. A synthetic data set was generated 1000 times, with 250 labeled points of each class for training, as well as 500 points of each class for testing. For the KNN and RDT learners, we experimented with different values for the number of neighbors and the maximum tree depth respectively, to validate the parameter choices we explained in Section 4.1 which we believed would have a large impact on their confidence measures. The results of this experiment are shown in Fig. 2.

From this graph we make a number of observations. RDT performed the worst of all learners, for both values of the maximum tree depth (4 and 8). As we hypothesized in Section 4.1 a smaller tree depth might lead to better probability estimates, and from this experiment we conclude it leads to better ranking ability as well. Ranking performance deteriorates as the problem dimension increases for both depth settings. For the KNN method, we observe that increasing the number of neighbors from 5, to 10 and finally to 20 causes the method to be a better ranker. This is to be expected as well, as the number of possible values the probability estimates can occupy is limited to be one more than the number of neighbors. By including more neighbors, a more distinctive ranking can be made. GNB is found to perform the best of all learners, for all problem dimensions except for four dimensions and fewer. In the really low dimensions, LR has the best ranking performance. As the problem complexity increases, however, LR declines in ranking ability and from 15 dimensions onward it performs worse than GNB, SVM and KNN_{20}. Interestingly, the ranking ability of SVM actually increases with problem dimensionality and it is found to perform comparably

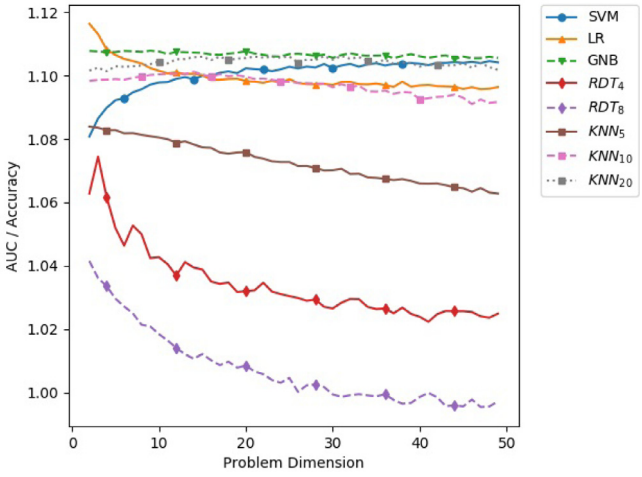

Fig. 2. Classifier ranking ability as the problem dimension increases. The parameter values for the maximum tree depth of the RDT and the number of neighbors of the KNN methods are shown in subscript.

to GNB and KNN_{20} for the maximum problem dimensions we explored. This experiment suggests that LR might not be as good of a ranker as it is an estimator: its ranking ability decreases as the relatively simple problem becomes more complex.

Additionally, we conducted an experiment where the problem dimension was fixed to 15 and we projected the probability estimates of the learners on the test set. The same number of examples were generated, once, for the train and test sets. The resulting scatter plots are shown in Fig. 3.

From this experiment it is immediately clear why LR and RDT are not suited for use with self-training. The plots show neither method has an obvious color gradient in their probability estimates towards the decision boundary. Furthermore, they are very confident about some predictions which are clearly on the wrong side of the boundary. When these points are added early on in the self-training process due to their high confidence, they can cause the learner to degrade in predictive performance in the following iterations. We observe instead a gradual change in probability estimates for the other methods, most clearly visible for SVM and GNB. A ranking based on these probability estimates will cause the self-training procedure to include points further from the decision boundary first and only later include points near the decision boundary about which it is less certain.

In conclusion, LR violates the assumption of being a good ranker, needed for the self-training process to not degrade. Due to the early-stopping mechanism, however, this does not lead to a degradation in ensemble performance compared to the supervised ensemble.

5.4. Effectiveness of early-stopping

To study the effect of the early-stopping mechanism, we refer again to Table 4. We observe that for each classifier and proportion of labeled data used, RESSEL with early-stopping is found to outperform RESSEL$^-$, in which the self-training procedure is completed and the final classifier is used in the ensemble. For 10 out of the 15 settings, this improvement is found to be significant.

Especially when self-training by itself is not effective, that is for RDT and LR, the early-stopping mechanism makes a large difference. For these classifiers, the self-training process is more prone to error and might cause the individual learners to compound errors made during the process, leading to poor final predictors. It is expected that the early-stopping mechanism will mitigate these errors.

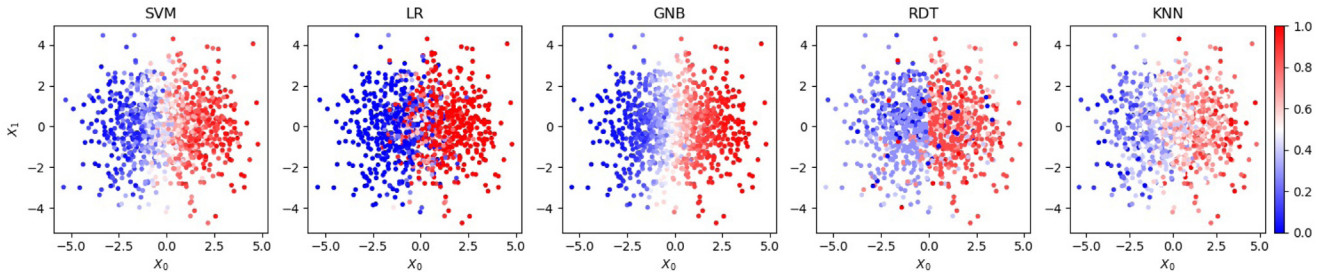

Fig. 3. Scatterplots of the probability estimates of the different base classifiers for two Gaussian distributions in 15 dimensions.

This does not, however, explain why we observe increased performance for RESSEL with RDT and LR over bagging due to the early-stopping mechanism, since they do not benefit from self-training, as we showed in Section 5.3. We suspect the increase is due to both of these methods being stochastic methods. During the self-training process, these classifiers go through many more training iterations than their counterparts in the bagging ensemble do, which are trained only once. Some of these iterations are likely to result in a better predictor by chance, which are then retained by the early-stopping mechanism. The increase in predictive accuracy is therefore not achieved by semi-supervised learning.

The large difference between RESSEL$^-$ and RESSEL for GNB cannot be explained by early-stopping counteracting its inability to learn through self-training because of its bad ranking ability, as for RDT and LR, since we observed GNB improved by using ranked based self-training in Section 5.3 and had good ranking ability on the synthetic data in Section 5.3.1. We measured the standard deviation of the difference between RESSEL$^-$ and RESSEL over all data sets for the 10% setting and found the following values for the different base classifiers: GNB: 2.83, SVM: 0.58, KNN: 0.58, RDT: 1.54 and LR 0.75. The results for GNB are thus very variable per data set. Perhaps the extent to which the conditional independence assumption of Naive Bayes is met highly influences its ranking ability on a per data set basis.

Taking a closer look at SVM_{10} in Table 10, it can be seen that while the mean performance over all data sets for RESSEL is indeed better and statistically significant, this is not guaranteed for every individual data set. In three out of the fifteen data sets studied, RESSEL$^-$ performed better. An explanation might be that while an individual learner decreases slightly in accuracy by advancing the learning process further, the diversity gain makes up for this decrease in an ensemble context.

The results of the high-dimensional data set experiment, shown in Table 6, follow the same pattern. The advantage of RESSEL over RESSEL$^-$ is large for GNB and also clearly visible for RDT and LR, although the differences are not found to be statistically significant in case of RDT and LR, with the exception of the RDT_{10} setting.

The differences between RESSEL and RESSEL$^-$ for the SVM and KNN base classifiers are smaller than before, however, with RESSEL$^-$ having the higher average accuracy in four out of six of these settings. The differences are not found to be statistically significant for any of these settings. For these data sets specifically, the self-training mechanism used with an appropriate base classifier performs quite well already and the additional advantage of error-checking using the out-of-bag set provides no advantage. Nevertheless, it is advised to use RESSEL rather than RESSEL$^-$ on high-dimensional data sets, as the observed difference in results is small and might be due to the specific nature of these data sets, while using RESSEL$^-$ carries an additional risk of decreased performance due to the absence of out-of-bag error-checking.

Table 10
Effectiveness of early-stopping for the SVM classifier with 10% labeled data. The mean accuracy (difference) is shown.

Data Set	RESSEL$^-$	RESSEL	Difference
Abalone	61.35	63.02	1.67
Australian	83.25	82.39	−0.86
Car	88.49	88.69	0.20
Contraceptive	49.75	51.02	1.27
German	71.07	71.81	0.74
Nursery	95.11	95.33	0.22
Pima	72.58	73.06	0.48
Red	72.24	72.47	0.23
Solar	72.42	73.01	0.59
Spambase	90.87	90.82	−0.05
Titanic	77.89	78.13	0.24
Vehicle	61.78	62.39	0.61
Vowel	51.23	51.25	0.02
White	75.45	75.46	0.01
Yeast	55.71	55.53	−0.18
Mean	71.95	72.29	0.34
Wilcoxon	5.85e−03	1.00e+00	–

5.5. Effectiveness of ensemble learning

Next, we study the added value the ensemble learning component brings to RESSEL. In considering the effectiveness of the ensemble, it has to be stressed that the early-stopping mechanism is made possible by the bagging procedure. Therefore, it is impossible to study the effect of ensemble learning in ablation, as we cannot compare to a single classifier trained on all data with self-training and early-stopping. Additionally, as early-stopping was shown to be effective in Section 5.4, the effectiveness of the ensemble is already partly validated.

In the following, we will study the effectiveness of the ensemble as it grows in size from a single classifier to 75 learners. Thereafter, we make a comparison between the advantage gained through ensemble learning by a single classifier, learned on all available labeled data, with and without self-training being used. This comparison is made without the early-stopping mechanism. Hereby we seek to answer the question if ensemble learning is helpful to self-training. All experiments are conducted with 10% of the training data as labeled data.

Table 11 shows the gain in average accuracy of RESSEL on the test set, as the ensemble grows from a single classifier to 75 classifiers. Overall, it can be seen that there is a large initial gain in accuracy, which has for the most part converged at ensemble size 25. Especially RDT and LR still show some increase when further increasing the size to 75, of 0.29 and 0.26 respectively, but compared to the total increase of 5.66 and 3.18 gained by increasing the size from 1 to 25, these improvements are minor.

This improvement in accuracy could be attributed to bagging alone, yet we are interested to see if there is an additional effect due to the combination with self-training and early-stopping.

To answer this question, we have calculated the average differences between RESSEL and bagging for different ensemble

Table 11
Effectiveness of RESSEL for varying ensemble sizes. The column with size 1 shows the single classifier accuracy, with the other columns showing the difference in accuracy to this single classifier. 10% of the training data is labeled.

	1	5	10	25	50	75
GNB	61.01	1.29	1.76	1.72	1.84	1.82
SVM	70.06	1.50	1.88	2.23	2.23	2.07
KNN	65.55	1.85	2.28	2.68	2.82	2.86
RDT	64.82	3.76	4.71	5.66	5.74	5.95
LR	67.60	2.26	2.85	3.18	3.30	3.44

Table 12
Effectiveness of ensemble learning. The average accuracy difference between RESSEL and bagging is shown. 10% of the training data is labeled.

	1	5	10	25	50	75
GNB	0.55	0.52	0.76	0.76	0.64	0.78
SVM	0.39	0.36	0.43	0.49	0.44	0.41
KNN	0.82	0.82	0.68	0.67	0.73	0.65
RDT	3.35	1.84	1.09	0.32	−0.07	−0.20
LR	2.01	0.95	0.57	0.27	0.20	0.24

sizes. The results are shown in Table 12. We observe that when self-training is effective, for SVM, KNN and to a lesser extend GNB, the advantage RESSEL has over bagging due to self-training with early-stopping, fluctuates a little for the different sizes, but remains mostly constant. This shows that self-training and early-stopping do not just offset accuracy loss due to limited ensemble sizes.

The RDT and LR methods, for which self-training is not effective, do see bagging gaining in accuracy over RESSEL as the ensemble size increases. We suspect this is due to the effect of early-stopping helping to find a good configuration, as discussed in Section 5.4, is diminished as more configurations are combined due to the larger ensemble size.

To further test our hypothesis that ensemble learning is enhanced by the diversity introduced through self-training, we compare the gain in classification accuracy between:

1. A single classifier enriched by self-training without early-stopping (SinClf+ST), which forms an ensemble through bagging (RESSEL$^-$).
2. The same classifier which does not undergo self-training (SinClf) and forms an ensemble through bagging.

This first ensemble thus corresponds to the RESSEL$^-$ method, and the second ensemble is a regular bagging ensemble. The results of these experiments are shown in Tables 13 and 14 respectively.

We observe that the appropriate single classifiers (SVM, GNB and KNN) enriched through self-training (Table 13) benefit much more on average from ensemble learning than a single classifier which is not enriched (Table 14), confirming our hypothesis. For the SVM and KNN, the overall accuracy ends up higher for RESSEL$^-$ than for the bagging ensemble as well, as we previously observed in Table 4.

Unsurprisingly, RDT and LR are found to benefit more from bagging than from RESSEL$^-$, having better overall accuracy as well with the bagging method.

5.6. Sensitivity to labeled data set size

The predictions obtained through self-training are sensitive to the composition of the initial labeled data set and the performance of the method when limited training data is available is therefore not self-evident. Similarly, bagging and especially the early-stopping mechanism used in RESSEL, are affected: measuring the out-of-bag error can be useful when the out-of-bag set

Table 13
Effectiveness of ensemble learning with self-training. The mean accuracy (difference) is shown. 10% of the training data is labeled.

Base classifier	SinClf+ST	RESSEL$^-$	Difference
GNB	57.91	59.41	1.50
SVM	70.67	71.95	1.28
KNN	66.22	67.83	1.61
RDT	62.41	68.82	6.41
LR	67.91	70.17	2.26

Table 14
Effectiveness of ensemble learning without self-training. The mean accuracy (difference) is shown. 10% of the training data is labeled.

Base classifier	SinClf	Bagging	Difference
GNB	60.77	61.97	1.20
SVM	71.59	71.80	0.21
KNN	67.37	67.56	0.19
RDT	63.32	70.16	6.84
LR	66.61	70.51	3.90

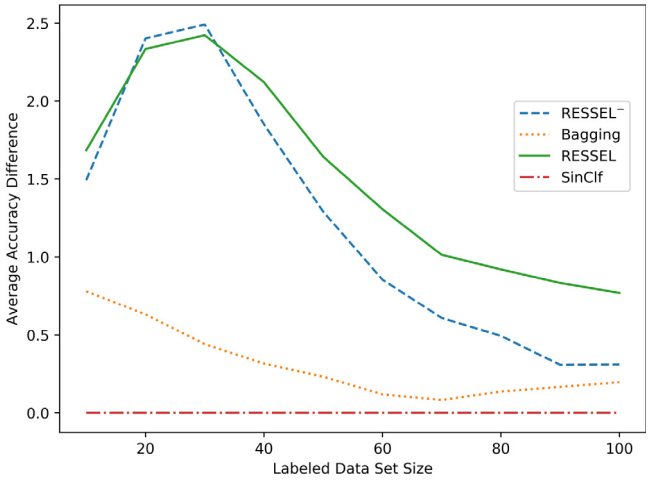

Fig. 4. The average accuracy difference of RESSEL$^-$, Bagging, RESSEL and SinClf to SinClf over the fifteen data sets as the training set size increases. A SVM was used as the base classifier.

is representative of an unseen test set, as this error then sufficiently approximates the generalization error. As mentioned in Section 2.2.1 in expectation 37% of the labeled data ends up in the out-of-bag set. When RESSEL is presented a training set consisting of ten examples, the out-of-bag set would most likely contain only three or four of these examples. The error measurement on such a small set cannot be expected to be a good approximation to the generalization error.

As we have stated in Section 2.1.1, the self-training method is difficult to analyze independent of its base classifier. Providing a performance guarantee on the RESSEL method, in which self-training is combined with bagging and out-of-bag error measurement, is an even more complex task and is not within the scope of this paper. Instead we empirically evaluate the performance of our method as the labeled data set size increases.

As before, we compare RESSEL to RESSEL without the early-stopping mechanism (RESSEL$^-$), to Bagging and to the single classifier (SinClf) over the fifteen data sets shown in Table 3, using SVM as the base classifier. The performance of the methods was measured 200 times for each labeled data set size, using a randomly sampled subset of the original data as the labeled data set. The data set size was increased from ten to one hundred, in intervals of ten.

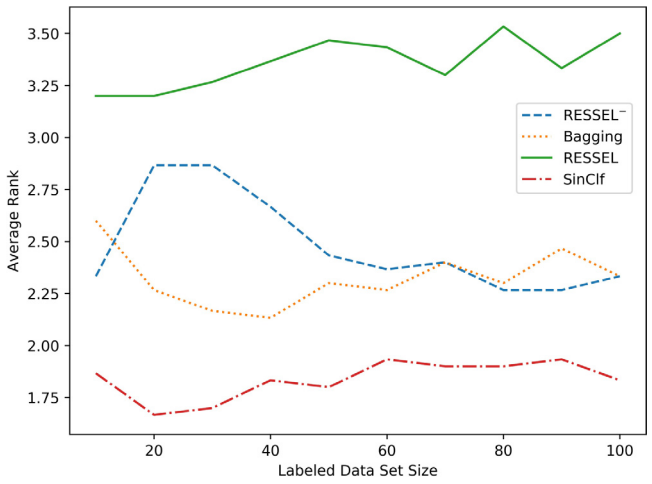

Fig. 5. The average rank of RESSEL⁻, Bagging, RESSEL and SinClf over the fifteen data sets as the training set size increases. The best performing method on a particular data set was assigned rank 4, the worst performing method rank 1. A SVM was used as the base classifier.

The average accuracy of the four methods over all data sets is shown in Fig. 4. We observe that both RESSEL and RESSEL⁻ outperform the supervised methods over the entire domain. In the lower data set sizes, RESSEL and RESSEL⁻ perform very similarly, while the performance gap increases as more data becomes available and RESSEL⁻ approaches Bagging. These trends can be seen to continue in Table 4, from which we observe for the SVM base classifier that RESSEL⁻ is almost equal in performance to Bagging as more labeled data becomes available, while still outperforming SinClf. RESSEL retains the best overall performance.

Moreover, we ranked the performance of each of the methods for each data set: the method with the highest average accuracy for a certain sample size obtained rank 4, the second highest rank 3, the second lowest rank 2 and the lowest method rank 1. The average ranks of the four methods over all data sets are shown in Fig. 5. We observe similar trends to those of the average accuracy, although RESSEL⁻ is not performing as well in terms of rank in the lower data size domain compared to Bagging as before.

From these observations we conclude that in the limited sample size regime the benefit of early-stopping is limited: the performance of RESSEL⁻ is almost equal to that of RESSEL. However, as more labeled samples become available to the methods, and thus the size and representativeness of the out-of-bag set increase, RESSEL outperforms RESSEL⁻ in classification accuracy.

Furthermore, the experiment shows that even in the limited sample regime the combination of Bagging with self-training can be beneficial. Bagging by itself performs better than the single classifier does and when combined with self-training and optionally early-stopping the predictive accuracy further increases.

We conclude both RESSEL and RESSEL⁻ are comparatively safe to use in the limited sample domain, as they outperform the supervised alternatives. The effectiveness of the early-stopping mechanism is shown to increase along with the sample size, however, and it is worthwhile to explicitly check whether RESSEL or RESSEL⁻ is the better choice if the amount of labeled data available is exceedingly small.

5.7. Robustness to choice of parameters

In addition to our research into the influence of the core components which constitute RESSEL, we studied the sensitivity of the method to the choice of the parameter settings of the base classifier. In our experiments, we used the SVM classifier, as it has been shown to work well with our method, as well as having a number of important parameters. This makes it a prime candidate for the study.

For these experiments, we used the Contraceptive, Red Wine and Titanic data sets, which represent a combination of different problems in terms of number of variables and outcome variable type. We varied the regularization parameter C and the kernel coefficient γ, looking at 21 different values for each, equidistant on a logarithmic scale. C was plotted from 0.01 to 1000 and γ in the range 0.0001 to 10, to capture the interesting part of the parameter landscape. In our experiments, each measurement of the test set accuracy for the specified parameter combination was repeated 50 times.

In Figs. 6, 7 and 8 we observe the effects that varying C and γ have. On the left side of these figures, we observe the validation accuracy of the ensemble constructed using bagging. In the middle, the validation accuracy of the RESSEL method is shown. The rightmost figures depict the difference in accuracy, subtracting the bagging accuracy from that of the RESSEL method.

An interesting effect presents itself from these images. There appears to be a band of parameter values near the boundary between the good parameter values and those where the classifier performs suboptimally, in which RESSEL performs much better than bagging does. This effect is clearly visible for each of the data sets.

For the Red Wine data set shown in Fig. 6, we observe that the area enclosed by this extended band is light red everywhere, suggesting that for all decent parameter values RESSEL increases predictive accuracy. Fig. 7 shows results for the Contraceptive data set. The enclosed area is predominantly red, with a few light blue colored dots, likely due to chance. In Fig. 8 we see once more that near the boundary of good parameter values RESSEL improves over bagging by a lot, although inside this band there is an area where the unlabeled data does not seem help and the difference is slightly negative.

These results underline the reliability of the RESSEL method: the hyperparameter settings of the base learners can be more loosely chosen, while results remain good, effectively extending the range of good parameter values.

6. Conclusion

In this paper, we recognize the scarcity of easy-to-use methods for incorporating unlabeled data into classification models. Many methods for semi-supervised learning are available, but they require the user who is accustomed with supervised method to acquaint themselves with an entirely new field of research.

We propose the reliable semi-supervised ensemble learning (RESSEL) method. RESSEL is based on the principle that ensemble learning and semi-supervised learning can complement each other. Our method combines bagging with self-training and introduces an early-stopping mechanism based on the out-of-bag error.

From the results of applying RESSEL to a number of commonly used data sets, we conclude that RESSEL is able to improve a given base classifier through the use of unlabeled data, provided that the base classifier can produce an effective ranking of the unlabeled instances from its probability estimates. When compared to previous works in the field we find RESSEL to perform significantly better, increasing the performance of a base classifier more effectively and consistently.

Further experiments show that the main components of RESSEL all contribute meaningfully to the effectiveness of the method as a whole. Additionally, we validate the importance of the good ranking assumption for the base classifiers. Finally, we show that RESSEL is robust to the choice of base classifier hyperparameters, effectively increasing the range of good parameter values.

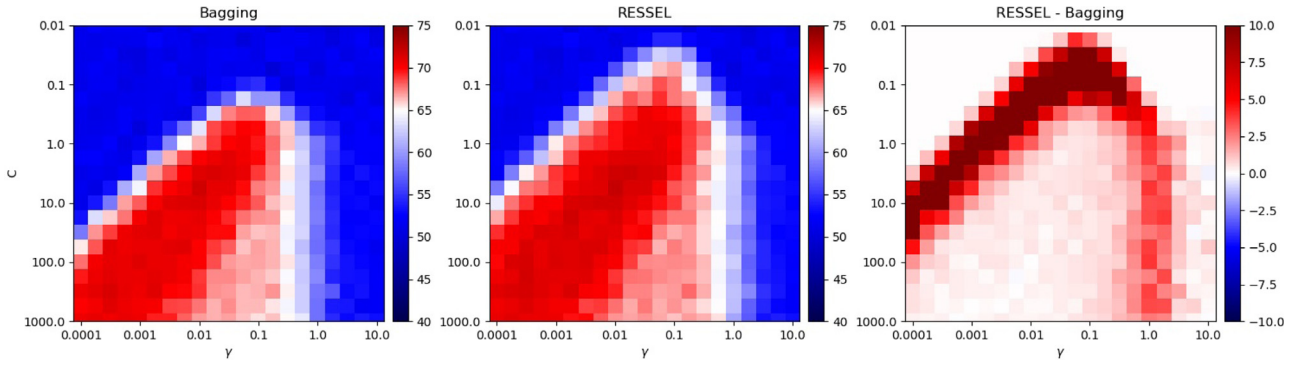

Fig. 6. Red Wine grid. The validation error is shown for different combinations of parameter values for the C and γ parameters, for a Bagging ensemble, the RESSEL method and their difference: RESSEL–Bagging.

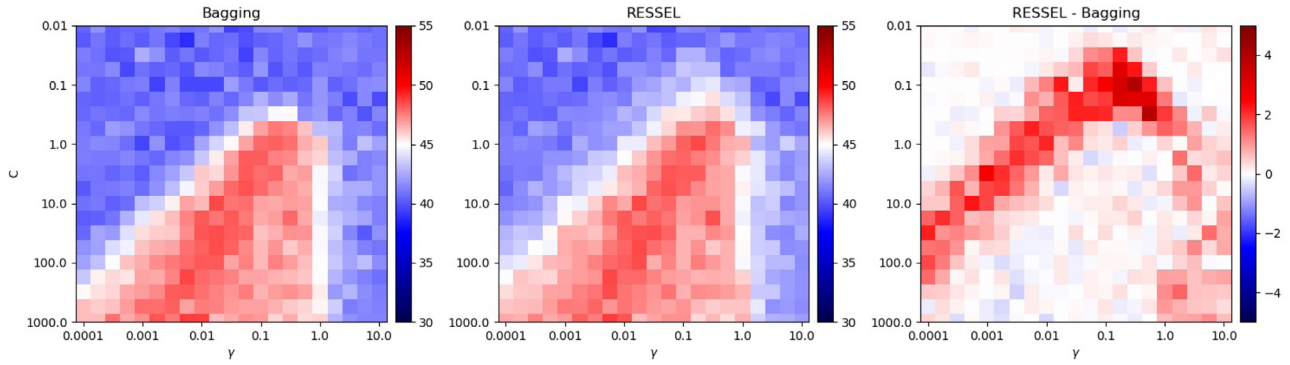

Fig. 7. Contraceptive grid. The validation error is shown for different combinations of parameter values for the C and γ parameters, for a Bagging ensemble, the RESSEL method and their difference: RESSEL–Bagging.

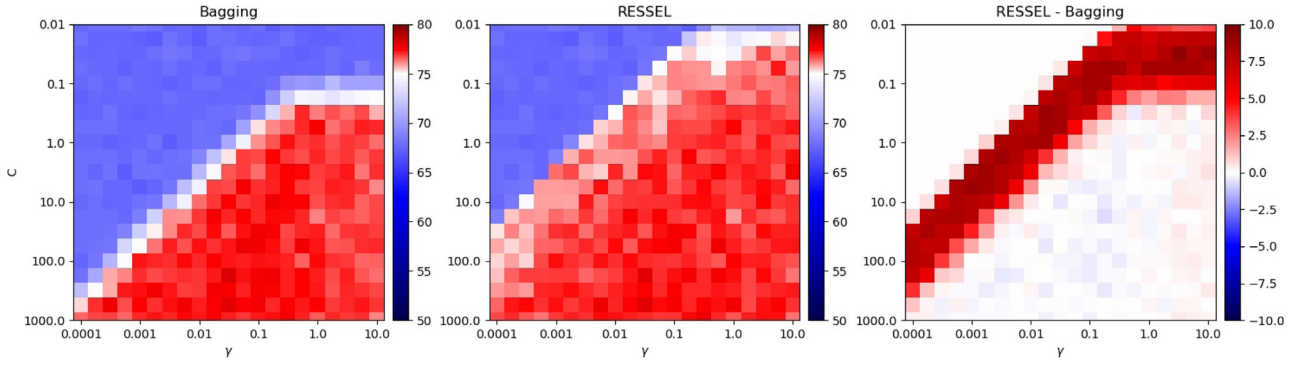

Fig. 8. Titanic grid. The validation error is shown for different combinations of parameter values for the C and γ parameters, for a Bagging ensemble, the RESSEL method and their difference: RESSEL–Bagging.

When the effective ranking assumption is not met, RESSEL still produces predictions equal in accuracy to those of supervised alternatives, and when the ranking assumption is met it provides significant improvements. The method relies on familiar concepts from supervised learning, as well as the basic semi-supervised learning method of self-training. We conclude that RESSEL thus answers the need for a reliable, easy-to-use method which can be used to incorporate unlabeled data to improve upon supervised predictors.

Future research could include conducting a thorough investigation into what causes RESSEL to increase the range of suitable hyperparameter values of its base classifiers. Additionally, more base learners could be examined for suitability for use with RESSEL by investigating their ranking ability, and additional research into the variability we observed in our experiments with Gaussian Naive Bayes when used in conjunction with self-training could be valuable. Furthermore, intelligent aggregation techniques could be implemented to replace the plurality vote mechanism.

CRediT authorship contribution statement

Sjoerd de Vries: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Visualization. **Dirk Thierens:** Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the University Medical Center Utrecht, Netherlands.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[2] Z.-H. Zhou, When semi-supervised learning meets ensemble learning, in: Int. Workshop Multiple Classifier Systems, 2009, pp. 529–538.
[3] Z.-H. Zhou, When semi-supervised learning meets ensemble learning, Front. Electr. Electron. Eng. China 6 (1) (2011) 6–16.
[4] O. Chapelle, B. Schölkopf, A. Zien, Semi-supervised learning, in: Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, USA, 2006.
[5] X. Zhu, A.B. Goldberg, Introduction to semi-supervised learning, in: Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool, San Rafael, CA, USA, 2009.
[6] G.K. Kostopoulos, S. Karlos, S. Kotsiantis, O. Ragos, Semi-supervised regression: A recent review, J. Intell. Fuzzy Systems 35 (2) (2018) 1483–1500, http://dx.doi.org/10.3233/JIFS-169689.
[7] N. Grira, M. Crucianu, N. Boujemaa, Unsupervised and semi-supervised clustering: a brief survey, in: A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence (6th Framework Programme), 2005, pp. 1–12.
[8] T. Joachims, Transductive inference for text classification using support vector machines, in: Int. Conf. Machine Learning, 1999, pp. 200–209.
[9] K.P. Bennett, A. Demiriz, Semi-supervised support vector machines, in: Advances in Neural Information Processing Systems, 1999, pp. 368–374.
[10] K. Nigam, A.K. McCallum, S. Thrun, T. Mitchell, Text classification from labeled and unlabeled documents using EM, Mach. Learn. 39 (2000) 103–134.
[11] A. Demiriz, K.P. Bennett, M.J. Embrechts, Semi-supervised clustering using genetic algorithms, in: Proc. Artificial Neural Networks in Engineering, 1999, pp. 809–814.
[12] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[13] T. Joachims, Transductive learning via spectral graph partitioning, in: Proc. 20th Int. Conf. Machine Learning, 2003, pp. 290–297.
[14] X. Zhu, Semi-Supervised Learning Literature Survey, Tech. Rep., (1530) Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, USA, 2005.
[15] A. Singh, R.D. Nowak, X. Zhu, Unlabeled data: Now it helps, now it doesn't, in: Advances in Neural Onformation Processing Systems, 2008, pp. 1513–1520.
[16] I. Triguero, S. García, F. Herrera, Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study, Knowl. Inf. Syst. 42 (2) (2015) 245–284, http://dx.doi.org/10.1007/s10115-013-0706-y.
[17] D. Yarowsky, Unsupervised word sense disambiguation rivaling supervised methods, in: Proc. 33rd Annu. Meeting Association for Computational Linguistics, 1995, pp. 189–196, http://dx.doi.org/10.3115/981658.981684.
[18] J. Tanha, M. van Someren, H. Afsarmanesh, Semi-supervised self-training for decision tree classifiers, Int. J. Mach. Learn. Cybern. 8 (2015) 355–370, http://dx.doi.org/10.1007/s13042-015-0328-7.
[19] Y. Guo, X. Niu, H. Zhang, An extensive empirical study on semi-supervised learning, in: Proc. 2010 IEEE Int. Conf. Data Mining, 2010, pp. 186–195, http://dx.doi.org/10.1109/ICDM.2010.66.
[20] M. Li, Z.-H. Zhou, SETRED: Self-training with editing, in: Pacific-Asia Conf. Knowledge Discovery and Data Mining, 2005, pp. 611–621.
[21] D. Wu, M. Shang, X. Luo, J. Xu, H. Yan, W. Deng, G. Wang, Self-training semi-supervised classification based on density peaks of data, Neurocomputing 275 (2018) 180–191.
[22] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[23] S. Oymak, T.C. Gulcu, Statistical and algorithmic insights for semi-supervised learning with self-training, 2020, http://arxiv.org/abs/2006.11006.
[24] C. Wei, K. Shen, Y. Chen, T. Ma, Theoretical analysis of self-training with deep networks on unlabeled data, 2020, http://arxiv.org/abs/2010.03622.
[25] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman & Hall, London, U.K., 2012.
[26] T.G. Dietterich, Ensemble methods in machine learning, in: 1st Int. Workshop Multiple Classifier Systems, 2000, pp. 1–15.
[27] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990) 993–1001, http://dx.doi.org/10.1109/34.58871.
[28] L.I. Kuncheva, Diversity in multiple classifier systems, Inf. Fusion 6 (1) (2005) 3–4, http://dx.doi.org/10.1016/j.inffus.2004.04.009.
[29] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, Hoboken, NJ, USA, 2004.
[30] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844, http://dx.doi.org/10.1109/34.709601.
[31] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, http://dx.doi.org/10.1023/A:1010933404324.
[32] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (2) (1990) 197–227, http://dx.doi.org/10.1007/BF00116037.
[33] Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, J. Comput. System Sci. 55 (1) (1997) 119–139.
[34] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Statist. (2001) 1189–1232.
[35] J.H. Friedman, Stochastic gradient boosting, Comput. Stat. Data Anal. 38 (4) (2002) 367–378, http://dx.doi.org/10.1016/S0167-9473(01)00065-2.
[36] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140, http://dx.doi.org/10.1007/BF00058655.
[37] B. Efron, R.J. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, London, U.K., 1993.
[38] J.J. Rodríguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: A new classifier ensemble method, IEEE Trans. Pattern Anal. Mach. Intell. 28 (10) (2006) 1619–1630.
[39] R. Tibshirani, Bias, variance and prediction error for classification rules, Tech. rep., Department of Statistics, University of Toronto, Toronto, Canada, 1996.
[40] D.H. Wolpert, W.G. Macready, An efficient method to estimate bagging's generalization error, Mach. Learn. 35 (1) (1999) 41–55.
[41] L. Breiman, Out-of-bag Estimation, Tech. rep., Department of Statistics, University of California, Berkeley, CA, USA, 1996.
[42] F. Roli, Semi-supervised multiple classifier systems: Background and research directions, in: Int. Workshop Multiple Classifier Systems, 2005, pp. 1–11.
[43] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proc. 11th Ann. Conf. Computational Learning Theory, 1998, pp. 92–100.
[44] Z.-H. Zhou, M. Li, Tri-training: Exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1529–1541.
[45] K.P. Bennett, A. Demiriz, R. Maclin, Exploiting unlabeled data in ensemble methods, in: Proc. 8th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2002, pp. 289–296.
[46] M.F. Hady, F. Schwenker, Co-training by committee: a new semi-supervised learning framework, in: Proc. IEEE Int. Conf. Data Mining Workshops, 2008, pp. 563–572.
[47] M.-L. Zhang, Z.-H. Zhou, Exploiting unlabeled data to enhance ensemble diversity, Data Min. Knowl. Discov. 26 (1) (2013) 98–129, http://dx.doi.org/10.1007/s10618-011-0243-9.
[48] X. Lu, J. Zhang, T. Li, Y. Zhang, Hyperspectral image classification based on semi-supervised rotation forest, Remote Sens. 9 (9) (2017) 924.
[49] Z. Yu, Y. Lu, J. Zhang, J. You, H.-S. Wong, Y. Wang, G. Han, Progressive semisupervised learning of multiple classifiers, IEEE Trans. Cybern. 48 (2) (2018) 689–702, http://dx.doi.org/10.1109/TCYB.2017.2651114.
[50] X. Dong, Z. Yu, W. Cao, Y. Shi, Q. Ma, A survey on ensemble learning, Front. Comput. Sci. 14 (2) (2020) 241–258, http://dx.doi.org/10.1007/s11704-019-8208-z.
[51] Y. Yan, Z. Xu, I.W. Tsang, G. Long, Y. Yang, Robust semi-supervised learning through label aggregation, in: Proc. 30th AAAI Conf. Artificial Intelligence, 2016, pp. 2244–2250.
[52] I. Livieris, A new ensemble self-labeled semi-supervised algorithm, Informatica 43 (2) (2019) 221–234.
[53] J. Li, Q. Zhu, Q. Wu, D. Cheng, An effective framework based on local cores for self-labeled semi-supervised classification, Knowl.-Based Syst. 197 (105804) (2020).
[54] H. Gan, N. Sang, R. Huang, X. Tong, Z. Dan, Using clustering analysis to improve semi-supervised classification, Neurocomputing 101 (2013) 290–298.
[55] Y. Wang, X. Xu, H. Zhao, Z. Hua, Semi-supervised learning based on nearest neighbor rule and cut edges, Knowl.-Based Syst. 23 (6) (2010) 547–554.
[56] C. Leistner, A. Saffari, J. Santner, H. Bischof, Semi-supervised random forests, in: IEEE 12th Int. Conf. Computer Vision, 2009, pp. 506–513, http://dx.doi.org/10.1109/ICCV.2009.5459198.
[57] J. Levatić, M. Ceci, D. Kocev, S. Džeroski, Semi-supervised classification trees, J. Intell. Inf. Syst. 49 (3) (2017) 461–486, http://dx.doi.org/10.1007/s10844-017-0457-4.
[58] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decis. Support Syst. 47 (4) (2009) 547–553, http://dx.doi.org/10.1016/j.dss.2009.05.016.

[59] N. Macià, E. Bernadó-Mansilla, Towards UCI+: A mindful repository design, Inform. Sci. 261 (2014) 237–262, http://dx.doi.org/10.1016/j.ins.2013.08.059.
[60] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2017, http://archive.ics.uci.edu/ml.
[61] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287.
[62] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[63] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Inform. Sci. 180 (10) (2010) 2044–2064, http://dx.doi.org/10.1016/j.ins.2009.12.010.
[64] F. Provost, P. Domingos, Tree induction for probability-based ranking, Mach. Learn. 52 (3) (2003) 199–215.
[65] D. Mease, A.J. Wyner, A. Buja, Boosted classification trees and class probability/quantile estimation, J. Mach. Learn. Res. 8 (2007) 409–439.
[66] J. Huang, J. Lu, C.X. Ling, Comparing naive Bayes, decision trees, and SVM with AUC and accuracy, in: Proc. 3rd IEEE Int. Conf. on Data Mining, 2003, pp. 553–556.
[67] H. Zhang, J. Su, Naive bayesian classifiers for ranking, in: Proc. European Conf. Machine Learning, 2004, pp. 501–512.
[68] B. Wang, B. Spencer, C.X. Ling, H. Zhang, Semi-supervised self-training for sentence subjectivity classification, in: Proc. 21st Canadian Conf. Artificial Intelligence, 2008, pp. 344–355, http://dx.doi.org/10.1007/978-3-540-68825-9_32.
[69] D.D. Margineantu, T.G. Dietterich, Improved class probability estimates from decision tree models, in: Nonlinear Estimation and Classification, in: Lecture Notes in Statistics, Springer-Verlag, New York, NY, USA, 2001, pp. 169–184.
[70] L. Jiang, H. Zhang, J. Su, Learning k-nearest neighbor naive bayes for ranking, in: Int. Conf. Advanced Data Mining and Applications, 2005, pp. 175–185.
[71] J.A. Hanley, B.J. McNeil, The meaning and use of the area under a receiver operating characteristic (ROC) curve, Radiology 143 (1) (1982) 29–36.