A Study on Greedy Algorithms for Ensemble Pruning

Ioannis Partalas, Grigorios Tsoumakas and Ioannis Vlahavas
{partalas,greg,vlahavas@csd.auth.gr}
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki 54124, Greece

January 13, 2012

Abstract

Ensemble selection deals with the reduction of an ensemble of predictive models in order to improve its efficiency and predictive performance. A number of ensemble selection methods that are based on greedy search of the space of all possible ensemble subsets have recently been proposed. They use different directions for searching this space and different measures for evaluating the available actions at each state. Some use the training set for subset evaluation, while others a separate validation set. This paper abstracts the key points of these methods and offers a general framework of the greedy ensemble selection algorithm, discussing its important parameters and the different options for instantiating these parameters.

1 Introduction

Ensemble methods [Dietterich, 1997] have been a very popular research topic during the last decade. They have attracted scientists from several fields including Statistics, Machine Learning, Pattern Recognition and Knowledge Discovery in Databases. Their popularity arises largely from the fact that they offer an appealing solution to several interesting learning problems of the past and the present, such as improving predictive performance over a single model, scaling inductive learning algorithms to large databases, learning from multiple physically distributed datasets and learning from concept-drifting data streams.

Typically, ensemble methods comprise two phases: the *production* of multiple predictive models and their *combination*. Recent work [Margineantu and Dietterich, 1997, Giacinto et al., 2000, Lazarevic and Obradovic, 2001, Fan et al., 2002, Tsoumakas et al., 2004, Caruana et al., 2004, Martinez-Munoz and Suarez, 2004, Banfield et al., 2005, Partalas et al., 2006, Zhang et al., 2006], has considered an additional intermediate phase that deals with the reduction of the

ensemble size prior to combination. This phase is commonly called *ensemble pruning*, *selective ensemble*, *ensemble thinning* and *ensemble selection*, of which we shall use the last one within this paper.

Ensemble selection is important for two reasons: *efficiency* and *predictive performance*. Having a very large number of models in an ensemble adds a lot of computational overhead. For example, decision tree models may have large memory requirements [Margineantu and Dietterich, 1997] and lazy learning methods have a considerable computational cost during execution. The minimization of run-time overhead is crucial in certain applications, such as in stream mining. Equally important is the second reason, predictive performance. An ensemble may consist of both high and low predictive performance models. The latter may negatively affect the overall performance of the ensemble. Pruning these models while maintaining a high diversity among the remaining members of the ensemble is typically considered a proper recipe for an effective ensemble.

A number of ensemble selection methods that are based on a greedy search of the space of all possible ensemble subsets, have recently been proposed [Margineantu and Dietterich, 1997, Fan et al., 2002, Caruana et al., 2004, Martinez-Munoz and Suarez, 2004, Banfield et al., 2005]. They use different directions for searching this space and different measures for evaluating the available actions at each state. Some use the training set for subset evaluation, while others a separate validation set. Most experimental studies compare a limited number of the different options for these parameters. Therefore no clear conclusions and general guidelines exist for greedy ensemble selection.

The above issues motivated this work, which makes the following contributions:

- It highlights the salient parameters of greedy ensemble selection algorithms, offers a critical discussion of the different options for instantiating these parameters and mentions the particular choices of existing approaches. The paper steers clear of a mere enumeration of particular approaches in the related literature, by generalizing their key aspects and providing comments, categorizations and complexity analysis wherever possible.

- It performs an extensive experimental study of several options of greedy ensemble selection algorithms both on homogeneous and heterogeneous classifier ensembles. The analysis of the results leads to several interesting conclusions, that were previously not discussed in the related literature.

The remainder of this paper is structured as follows. Section 2 contains background material on ensemble production and combination. Section 3 presents the generic greedy ensemble selection algorithm. Section 4 gives the details of the experimental setup and Section 5 discusses the results. Finally, Section 6 summarizes this work and outlines the main conclusions.

2 Background

This section provides background material on ensemble methods. More specifically, information about the different ways of producing models are presented as well as different methods for combining the decisions of the models.

2.1 Producing the Models

An ensemble can be composed of either *homogeneous* or *heterogeneous models*. Homogeneous models derive from different executions of the same learning algorithm. Such models can be produced by using different values for the parameters of the learning algorithm, injecting randomness into the learning algorithm or through the manipulation of the training instances, the input attributes and the model outputs [Dietterich, 2000]. Popular methods for producing homogeneous models are *bagging* [Breiman, 1996] and *boosting* [Schapire, 1990].

Heterogeneous models derive from running different learning algorithms on the same data set. Such models have different views about the data, as they make different assumptions about it. For example, a neural network is robust to noise in contrast with a k-nearest neighbor classifier.

2.2 Combining the Models

Common methods for combining an ensemble of predictive models include *voting*, *stacked generalization* and *mixture of experts*.

In voting, each model outputs a class value (or ranking, or probability distribution) and the class with the most votes is the one proposed by the ensemble. When the class with the maximum number of votes is the winner, the rule is called *plurality voting* and when the class with more than half of the votes is the winner, the rule is called *majority voting*. A variant of voting is weighted voting where the models are not treated equally as each of them is associated with a coefficient (weight), usually proportional to its classification accuracy.

Stacked generalization [Wolpert, 1992], also known as *stacking* is a method that combines models by learning a meta-level (or level-1) model that predicts the correct class based on the decisions of the base level (or level-0) models. This model is induced on a set of meta-level training data that are typically produced by applying a procedure similar to k-fold cross validation on the training data. The outputs of the base-learners for each instance along with the true class of that instance form a meta-instance. A meta-classifier is then trained on the meta-instances. When a new instance appears for classification, the output of the all base-learners is first calculated and then propagated to the meta-classifier, which outputs the final result.

The mixture of experts architecture [Jacobs et al., 1991] is similar to the weighted voting method except that the weights are not constant over the input space. Instead there is a gating network which takes as input an instance and outputs the weights that will be used in the weighted voting method for that

specific instance. Each expert makes a decision and the output is averaged as in the method of voting.

3 Greedy Ensemble Selection

Greedy ensemble selection algorithms attempt to find the globally best subset of classifiers by taking local greedy decisions for changing the current subset. An example of the search space for an ensemble of four models is presented in Figure 1.

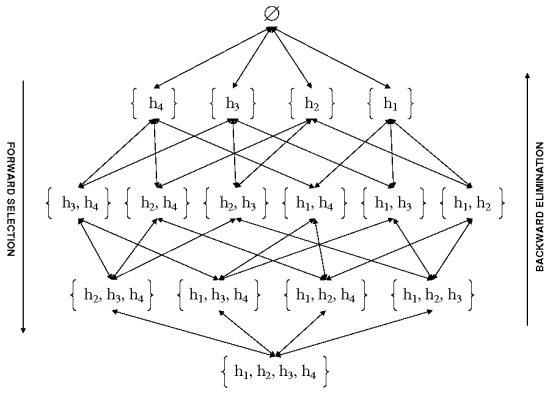


Figure 1: An example of the search space of greedy ensemble selection algorithms for an ensemble of four models

In the following subsections we present and discuss on what we consider to be the main aspects of greedy ensemble selection algorithms: the direction of search, the measure and dataset used for evaluating the different branches of the search and the size of the final subensemble. The notation that will be used is the following.

- $D = \{(x_i, y_i), i = 1, 2, \ldots, N\}$ is an evaluation set of labelled training examples where each example consists of a feature vector x_i and a class label y_i.
- $H = \{h_t, t = 1, 2, \ldots, T\}$ is the set of classifiers or hypotheses of an ensemble, where each classifier h_t maps an instance x to a class label y, $h_t(x) = y$.
- $S \subseteq H$, is the current subensemble during the search in the space of subensembles.

3.1 Direction of Search

Based on the direction of search, there are two main categories of greedy ensemble selection algorithms: *forward selection* and *backward elimination*.

In forward selection, the current classifier subset S is initialized to the empty set. The algorithm continues by iteratively adding to S the classifier $h_t \in H \backslash S$ that optimizes an evaluation function $f_{FS}(S, h_t, D)$. This function evaluates the addition of classifier h_t in the current subset S based on the labelled data of D. For example, f_{FS} could return the accuracy of the ensemble $S \cup h_t$ on the data set D by combining the decisions of the classifiers with the method of voting. Algorithm 1 shows the pseudocode of the forward selection ensemble selection algorithm. In the past, this approach has been used in [Fan et al., 2002, Martinez-Munoz and Suarez, 2004, Caruana et al., 2004] and in the Reduce-Error Pruning with Backfitting (REPwB) method in [Margineantu and Dietterich, 1997].

Algorithm 1 The forward selection method in pseudocode

Require: Ensemble of classifiers H, evaluation function f_{FS}, evaluation set D

1: $S = \emptyset$
2: **while** $S \neq H$ **do**
3: $\quad h_t = \arg\max_{h \in H \backslash S} f_{FS}(S, h, D)$
4: $\quad S = S \cup \{h_t\}$
5: **end while**

In backward elimination, the current classifier subset S is initialized to the complete ensemble H and the algorithm continues by iteratively removing from S the classifier $h_t \in S$ that optimizes the evaluation function $f_{BE}(S, h_t, D)$. This function evaluates the removal of classifier h from the current subset S based on the labelled data of D. For example, f_{BE} could return a measure of diversity for the ensemble $S \setminus \{h_t\}$, calculated on the data of D. Algorithm 2 shows the pseudocode of the backward elimination ensemble selection algorithm. In the past, this approach has been used in the *AID thinning* and *concurrency thinning* algorithms [Banfield et al., 2005].

Algorithm 2 The backward elimination method in pseudocode

Require: Ensemble of classifiers H, evaluation function f_{BE}, evaluation set D

1: $S = H$
2: **while** $S \neq \emptyset$ **do**
3: $\quad h_t = \arg\max_{h \in S} f_{BE}(S, h, D)$
4: $\quad S = S \setminus \{h_t\}$
5: **end while**

The time complexity of greedy ensemble selection algorithms for traversing the space of subensembles is $O(t^2 g(T, N))$. The term $g(T, N)$ concerns the complexity of the evaluation function, which is linear with respect to N and ranges from constant to quadratic with respect to T, as we shall see in the following subsections.

3.2 Evaluation Function

One of the main components of greedy ensemble selection algorithms is the function that evaluates the alternative branches during the search in the space of subensembles. Given a subensemble S and a model h the evaluation function estimates the utility of inserting (deleting) h into (from) S using an appropriate *evaluation measure*, which is calculated on an *evaluation dataset*. Both the measure and the dataset used for evaluation are very important, as their choice affects the quality of the evaluation function and as a result the quality of the selected ensemble.

3.2.1 Evaluation Dataset

One approach is to use the training dataset for evaluation, as in [Martinez-Munoz and Suarez, 2004]. This approach offers the benefit that plenty of data will be available for evaluation and training, but is susceptible to the danger of overfitting.

Another approach is to withhold a part of the training set for evaluation, as in [Caruana et al., 2004, Banfield et al., 2005] and in the REPwB method in [Margineantu and Dietterich, 1997]. This approach is less prone to overfitting, but reduces the amount of data that are available for training and evaluation compared to the previous approach. It sacrifices both the predictive performance of the ensemble's members and the quantity of the evaluation data for the sake of using unseen data in the evaluation. This method should probably be preferred over the previous one, when there is abundance of training data.

An alternative approach that has been used in [Caruana et al., 2006], is based on k-fold cross-validation. For each fold an ensemble is created using the remaining folds as the training set. The same fold is used as the evaluation dataset for models and subensembles of this ensemble. Finally, the evaluations are averaged across all folds. This approach is less prone to overfitting as the evaluation of models is based on data that were not used for their training and at the same time, the complete training dataset is used for evaluation.

During testing the above approach works as follows: the k models that where trained using the same procedure (same algorithm, same subset, etc.) form a cross-validated model. When the cross-validated model makes a prediction for an instance, it averages the predictions of the individuals models. An alternative testing strategy that we suggest for the above approach is to train an additional single model from the complete training set and use this single model during testing.

3.2.2 Evaluation Measure

The evaluation measures can be grouped into two major categories: those that are based on *performance* and those on *diversity*.

The goal of performance-based measures is to find the model that maximizes the performance of the ensemble produced by adding (removing) a model to (from) the current ensemble. Their calculation depends on the method used for

ensemble combination, which usually is voting. Accuracy was used as an evaluation measure in [Margineantu and Dietterich, 1997, Fan et al., 2002], while Caruana et al. [2004] experimented with several metrics, including accuracy, root-mean-squared-error, mean cross-entropy, lift, precision/recall break-even point, precision/recall F-score, average precision and ROC area. Another measure is *benefit* which is based on a cost model and has been used in [Fan et al., 2002].

The calculation of performance-based metrics requires the decision of the ensemble on all examples of the pruning dataset. Therefore, the complexity of these measures is $O(|S|N)$. However, this complexity can be optimized to $O(N)$, if the predictions of the current ensemble are updated incrementally each time a classifier is added to/removed from it.

It is generally accepted that an ensemble should contain diverse models in order to achieve high predictive performance. However, there is no clear definition of diversity, neither a single measure to calculate it. In their interesting study, Kuncheva and Whitaker [2003], could not reach into a solid conclusion on how to utilize diversity for the production of effective classifier ensembles. In a more recent theoretical and experimental study on diversity measures [Tang et al., 2006], the authors reached to the conclusion that diversity cannot be explicitly used for guiding the process of greedy ensemble selection. Yet, certain approaches have reported promising results [Martinez-Munoz and Suarez, 2004, Banfield et al., 2005].

One issue that worths mentioning here is how to calculate the diversity during the search in the space of ensemble subsets. For simplicity we consider the case of forward selection only. Let S be the current ensemble and $h \in H \setminus S$ a candidate classifier to add to the ensemble.

One could compare the diversities of subensembles $S' = S \bigcap h_t$ for all candidate $h_t \in H \setminus S$ and select the ensemble with the highest diversity. Any pairwise and non-pairwise diversity measure can be used for this purpose. The time complexity of most non-pairwise diversity measures is $O(|S'|N)$, while that of pairwise diversity measures is $O(|S'|^2N)$. However, a straightforward optimization can be performed in the case of pairwise diversity measures. Instead of calculating the sum of the pairwise diversity for every pair of classifiers in each candidate ensemble S', one can simply calculate the sum of the pairwise diversities only for the pairs that include the candidate classifier h_t. The sum of the rest of the pairs is equal for all candidate ensembles. The same optimization can be achieved in backward elimination too. This reduces their time complexity to $O(|S|N)$.

Existing methods [Martinez-Munoz and Suarez, 2004, Banfield et al., 2005, Tang et al., 2006] use a different approach to calculate diversity during the search. They use pairwise measures to compare the candidate classifier h_t with the current ensemble S, which is viewed as a single classifier that combines the decisions of its members with voting. This way they calculate the diversity between the current ensemble as a whole and the candidate classifier. Such an approach has time complexity $O(|S|N)$, which can be optimized to $O(N)$, if the predictions of the current ensemble are updated incrementally each time a

classifier is added to/removed from it. However, these calculations do not take into account the decisions of individual models.

In the past, the widely known diversity measures *disagreement*, *double fault*, *Kohavi-Wolpert variance*, *inter-rater agreement*, *generalized diversity* and *difficulty* were used for greedy ensemble selection in Tang et al. [2006]. *Concurrency* [Banfield et al., 2005], *margin distance minimization* and *Complementariness* Martinez-Munoz and Suarez [2004] are three diversity measures designed specifically for greedy ensemble selection. We next present these measures using a common notation.

The *complementariness* of a model h_k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$COM_D(h_k, S) = \sum_{i=1}^{N} I\Big(y_i = h_k(x_i)\ AND\ y_i \neq S(x_i)\Big),$$

where $I(true) = 1$, $I(false) = 0$ and $S(x_i)$ is the classification of instance x_i from the subensemble S. This classification is derived from the application of an ensemble combination method to S, which usually is voting. The complementariness of a model with respect to a subensemble is actually the number of examples of D that are classified correctly by the model and incorrectly by the subensemble. A selection algorithm that uses the above measure, tries to add (remove) at each step the model that helps the subensemble classify correctly the examples it gets wrong.

The *concurrency* of a model h_k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$CON_D(h_k, S) = \sum_{i=1}^{N} \Bigg(-2 * I\Big(y_i \neq h_k(x_i)\ AND\ y_i \neq S(x_i)\Big) +$$

$$+2 * I\Big(y_i = h_k(x_i)\ AND\ y_i \neq S(x_i)\Big) + I\Big(y_i = h_k(x_i)\ AND\ y_i = S(x_i)\Big)\Bigg)$$

This measure is very similar to complementariness with the difference that it takes into account two extra cases. A possible extension is to take also under consideration the case that the ensemble S classifies correctly the instance and the model k does not.

The *margin distance minimization* method [Martinez-Munoz and Suarez, 2004] follows a different approach for calculating the diversity. For each classifier h_t an N-dimensional vector, c_t, is defined where each element $c_t(i)$ is equal to 1 if the t^{th} classifier classifies correctly instance i, and -1 otherwise. The vector, C_S of the subensemble S is the average of the individual vectors c_t, $C_S = \frac{1}{|S|}\sum_{t=1}^{|S|} c_t$. When S classifies correctly all the instances the corresponding vector is in the first quadrant of the N-dimensional hyperplane. The objective is to reduce the distance, $d(o, C)$, where d is the Euclidean distance and o a predefined vector placed in the first quadrant. The margin, $MAR_D(h_k, S)$,

of a classifier k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$MAR_D(h_k, S) = d\left(o, \frac{1}{|S|+1}\Big(c_k + C_S\Big)\right)$$

A disadvantage of calculating pairwise diversity measures between the candidate classifier and the current ensemble, is that the decisions of individual models are not considered, as the current ensemble is treated as a whole. We hypothesize that better results can be obtained from a measure that takes into account the *strength* of the current ensemble's decision. In particular, we argue that an example that is incorrectly (correctly) classified by most of the members of the current ensemble, should not affect strongly the evaluation measure, as this is probably a very hard (easy) example. On the other hand, examples that are misclassified by about half of the ensemble's members, are near to change status (correct/incorrect classification) and should strongly affect the measure.

Another issue concerning the concurrency measure in particular, is that it does not take into account the case where the classifier is wrong and the current ensemble correct. This is an important case, as it might lead to misclassification of examples, especially those near to change status.

A measure that deals with this problem is Focused Selection Diversity (FSD) which considers all cases and takes into account the strength of the current ensemble's decision [Partalas et al., 2008][1]. It first defines the following quantities: NT_i, which denotes the number of models in the current ensemble S that classify example i correctly and NF_i that denotes the number of models in S that classify example i incorrectly. The measure is defined as follows:

$$FSD_D(h_k, S) = \frac{1}{|S|}\sum_{i=1}^{N}\Bigg(NT_i * I\Big(y_i = h_k(x_i)\ AND\ y_i \neq S(x_i)\Big) -$$

$$-NF_i * I\Big(y_i \neq h_k(x_i)\ AND\ y_i = S(x_i)\Big) + NF_i * I\Big(y_i = h_k(x_i)\ AND\ y_i = S(x_i)\Big) -$$

$$-NT_i * I\Big(y_i \neq h_k(x_i)\ AND\ y_i \neq S(x_i)\Big)\Bigg)$$

The measure rewards the cases where the model h_k under consideration is correct and penalizes the cases where it is incorrect. If the current ensemble S is incorrect, then the reward/penalty is multiplied with the proportion of correct models in S. On the other hand, if S is correct, then the reward/penalty is multiplied with the proportion of incorrect models in S. This weighting scheme focuses the attention of the measure to examples near to change status, while it overlooks examples that are either very easy or very hard to classify correctly.

[1] An extended and comprehensive version of this work can be found in [Partalas et al., 2010]

3.3 Size of Final Ensemble

Another issue that concerns greedy ensemble selection algorithms, is when to stop the search process, or in other words how many models should the final ensemble include.

One solution is to perform the search until all models have been added into (removed from) the ensemble and select the subensemble with the highest accuracy on the evaluation set. This approach has been used in [Caruana et al., 2004]. Others prefer to select a predefined number of models, expressed as a percentage of the original ensemble [Margineantu and Dietterich, 1997, Fan et al., 2002, Martinez-Munoz and Suarez, 2004, Banfield et al., 2005].

4 Experimental Setup

We experiment on 14 datasets, 10 from the UCI Machine Learning repository [Asuncion and Newman, 2007] and 4 from the *Agnostic vs. Prior Knowledge* challenge[2] and more specifically from the agnostic track. Tables 1 and 2 present the details of these datasets. We avoid using UCI datasets containing less than 900 examples, so that an adequate amount of data is available for training, evaluation and testing.

Id	UCI Folder	Inst	Cls	Cnt	Dsc	MV(%)
d1	car	1728	4	0	6	0.00
d2	cmc	1473	3	2	7	0.00
d3	credit-g	1000	2	7	13	0.00
d4	hypothyroid	3772	4	7	23	5.40
d5	image	2310	7	19	0	0.00
d6	kr-vs-kp (chess)	3196	2	0	36	0.00
d7	tic-tac-toe	958	2	0	9	0.00
d8	vehicle	946	4	18	0	0.00
d9	vowel	990	11	3	10	0.00
d10	waveform-5000	5000	3	21	0	0.00

Table 1: Details of datasets from the UCI Machine Learning repository: Folder in UCI server, number of instances, classes, continuous and discrete attributes, percentage of missing values.

Initially, each dataset is split into three disjunctive parts: a training set D_{Tr}, a selection set D_S and a test set D_{Te}, consisting of 60%, 20% and 20% of the examples in this dataset respectively. Two different ensemble production methods are then used, in order to create an ensemble of 200 models: a) bootstrap sampling (as in bagging), and b) running different learning algorithms with different parameter configurations. To the best of our knowledge, this is

[2] http://www.agnostic.inf.ethz.ch/

Id	Name	Inst	Cls	Attr	MV(%)
d11	ada	4147	2	48	0.00
d12	gina	3153	2	970	0.00
d13	hiva	3846	2	1617	0.00
d14	sylva	13086	2	216	0.00

Table 2: Details of datasets from the *Agnostic vs. Prior Knowledge* challenge: name, number of instances, classes, number of attributes, percentage of missing values.

the only experimental study of ensemble selection algorithms considering both homogeneous and heterogeneous models.

The WEKA machine learning library is used as the source of learning algorithms [Witten and Frank, 2005] in both cases. In the first case we train 200 decision trees, using default parameters. In the second case we train 28 multilayer perceptrons, 66 k-NNs, 100 support vector machines (SVMs), 2 naive Bayes classifiers and 4 decision trees. The different parameters that are used to train these algorithms are the following (default values are used for the rest of the parameters):

- Multilayer perceptrons: we use 6 values for the nodes in the hidden layer {1, 2, 4, 8, 16, 32, 64} and 4 values for the momentum term {0.0, 0.2, 0.5, 0.8}.
- kNNs: we use 22 values for k distributed evenly between 1 and the plurality of the training instances. We also use 3 weighting methods: no-weighting, inverse-weighting and similarity-weighting.
- SVMs: we use 10 values for the complexity parameter {10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 10, 100}, and 10 different kernels. We use 2 polynomial kernels (degree 2 and 3) and 8 radial kernels (gamma {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2}).
- Naive Bayes: we build one model with default parameters and one with kernel estimation.
- Decision trees: we use 2 values for the confidence factor {0.25, 0.5}, and 2 values for Laplace smoothing {true, false}.

In order to evaluate the utility of using a separate selection dataset for the evaluation function, we train two different ensembles of models for each one of the above ensemble production methods: In the first ensemble, D_{Tr} is used for training the models. D_S is then used for evaluation and D_{Te} for testing. In the second ensemble, the dataset $D_{Tr} \bigcup D_S$ is used for both training and evaluation. As in the previous case, D_{Te} is then used for testing. In this way, we make a fair comparison between using just the training set and using a separate dataset for evaluation.

In the next step, we use the greedy ensemble selection algorithm after setting the parameters of *direction*, *evaluation dataset* and *evaluation measure*. We experiment with the direction parameter using both forward (`F`) and backward (`B`) as values. For the evaluation dataset, we use both the training set (`T`) and a separate selection set (`S`) as the evaluation dataset, as explained in the previous paragraph. Concerning the evaluation measure, we use the following 5 measures: accuracy (`ACC`), concurrency (`CON`), complementariness (`COM`), margin (`MAR`) and the proposed measure (`FSD`). Voting is used as the method for model combination within the computation of `ACC`, `CON`, `COM` and `FSD`. The reference vector o in `MAR` is set to $\{0.25, 0.25, \ldots, 0.25\}$. Table 3 shows the acronyms for the different instantiations of the greedy ensemble selection algorithm, which will be used in the section discussing the experimental results.

Acronym	Direction	Evaluation Dataset	Evaluation Measure
FTACC	Forward	Training	Accuracy
FTCON	Forward	Training	Concurrency
FTCOM	Forward	Training	Complementariness
FTMAR	Forward	Training	Margin
FTFSD	Forward	Training	Focused Selection Diversity
FSACC	Forward	Selection	Accuracy
FSCON	Forward	Selection	Concurrency
FSCOM	Forward	Selection	Complementariness
FSMAR	Forward	Selection	Focused Selection Diversity
BTACC	Backward	Training	Accuracy
BTCON	Backward	Training	Concurrency
BTCOM	Backward	Training	Complementariness
BTMAR	Backward	Training	Margin
BTFSD	Backward	Training	Focused Selection Diversity
BSACC	Backward	Selection	Accuracy
BSCON	Backward	Selection	Concurrency
BSCOM	Backward	Selection	Complementariness
BSMAR	Backward	Selection	Margin
BSFSD	Backward	Selection	Focused Selection Diversity

Table 3: Acronym, search direction, evaluation dataset and evaluation measure for the different instantiations of the greedy ensemble selection algorithm.

We follow the approach of [Caruana et al., 2004] and select the subensemble with the highest classification accuracy on the selection set (using voting), instead of using an arbitrary percentage of models. We record the size of the resulting ensemble and its classification accuracy on the test set, using voting for model combination. The whole experiment is performed 10 times for each dataset and the results are averaged.

We also calculate the performance of the complete ensemble of 200 models (`ALL`) using voting for model combination and the best single model (`BSM`) based

on the performance of the models on the evaluation dataset. The set $D_{Tr} \bigcup D_S$ is used for training the models in both cases, as well as for evaluation in the second case.

The source code that was used for conducting the experiments is available at the following URL: `http://mlkd.csd.auth.gr/ensemblepruning.html`.

5 Results and Discussion

In this section we present and discuss the results that came up from the experiments we conducted for both homogeneous and heterogeneous models. We also report results concerning our submission to the *Agnostic vs. Prior Knowledge* challenge.

5.1 Homogeneous Models

Results concerning homogeneous models are analyzed from the perspectives of predictive performance, final ensemble size and the relationship between them.

5.1.1 Predictive Performance

Tables 6 and 7 in the Appendix present the classification accuracy and corresponding rank respectively for each algorithm and dataset along with the mean accuracy and rank across all datasets. Following the recommendation of Demsar [2006], we start the performance comparison of the different algorithms based on their average rank across all datasets. We notice that the two best performing algorithms are `BTFSD` and `FTFSD`. This shows that FSD leads to the highest predictive performance irrespectively of the search direction, when the training set is used for evaluation.

Figure 2 presents aggregates of the mean ranks for the different values of the search direction (a), evaluation dataset (b) and evaluation measure (c) parameters, as well as aggregates for the different values of parameter pairs (d-f).

A first interesting remark based on Figure 2(b), is that the mean rank of algorithms that use the selection set (14.21) for evaluation is much larger than the mean rank of algorithms that use the training set (6.78), indicating a clear superiority of the latter. The Wilcoxon signed-ranks test [Wilcoxon, 1945] is employed in order to verify this hypothesis. In specific, 10 tests are performed for the corresponding pairs of algorithms that use the same values for the direction and evaluation measure parameters but different values for the evaluation dataset parameter (`xSxxx` vs. `xTxxx`). For a confidence level of 95%, the differences between all of the 10 pairs of algorithms are significant. We therefore conclude that the use of the training set offers significant merits to ensemble selection algorithms in homogeneous ensembles.

Algorithms that use a separate selection set for evaluation, use less data for training, leading to models with lower predictive performance. In addition,

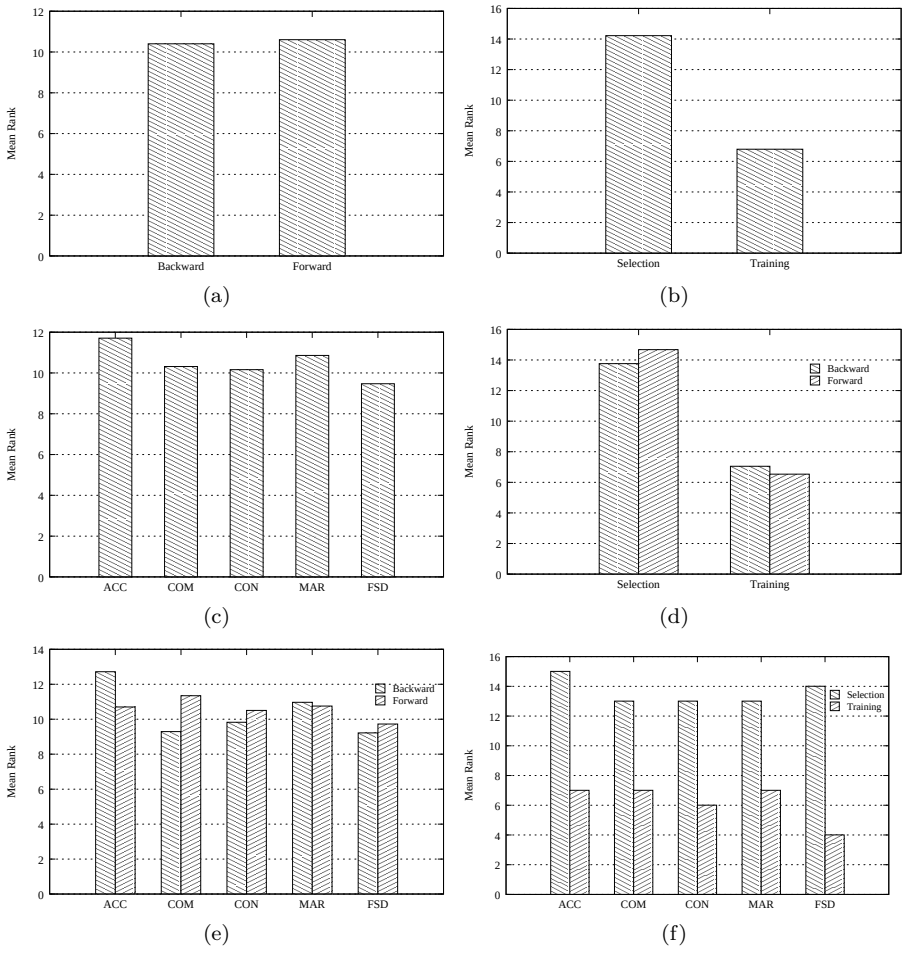


Figure 2: Mean rank across all datasets for different values of parameters and parameter pairs.

the selection set is much smaller than the training set, making the evaluation less resilient to noise. On the other hand, using the training set for evaluation purposes is generally considered an unreliable approach, because models usually overfit the training set and performance evaluations are optimistic. Here, we notice that this doesn't hold. We believe that this happens, because the ensemble is composed of homogeneous models, and as a consequence all models are equally affected by overfitting. This makes the amount of training data to play the decisive role in the performance of ensemble selection algorithms.

Concerning the evaluation measures, the mean ranks of the algorithms are the following in ascending order: `FSD` 9.47, `CON` 10.16, `COM` 10.31, `MAR` 10.86 and `ACC` 11.71. We notice that the proposed measure has the best average performance, but the difference with the rest of the measures is not large. Figure 2(f) shows the mean ranks of the algorithms grouped by the values of the evaluation measure and evaluation dataset parameters. Focusing on the results of the significantly better algorithms that use the training set for evaluation, we notice that the superiority of `FSD` is more evident. However, the differences with the rest of the measures are still rather small.

In order to investigate whether there are significant differences among the performance of the different measures for the `xTxxx` algorithms, we initially employ Friedman's test, which shows critical differences among the algorithms. Following the recommendation of Demsar [2006], we proceed to the post-hoc Nemenyi test [Nemenyi, 1963]. Figure 3 graphically represents the result of the test with 90% confidence, $q_{0.10} = 2.920$ and $CD = 3.341$. The best ranks are to the right and the groups of algorithms that are not significantly different are connected with a bold line. We notice that there are two groups of similar algorithms. The only statistically significant difference is that of `FTFSD` over `BTMAR`.

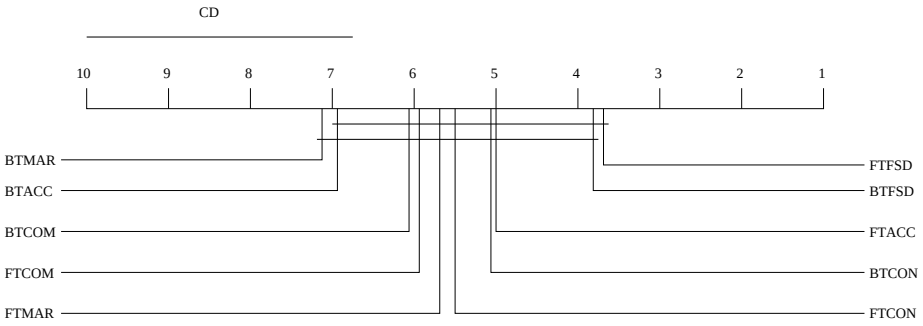


Figure 3: Graphical representation of the Nemenyi test for the algorithms that use the training set for evaluation.

Concerning the search direction parameter, we notice that the mean rank of algorithms that search in the forward direction (10.60) is approximately the same with the mean rank of the algorithms that search in the backward direction (10.40). We therefore conclude that the search direction does not significantly

affect the performance of ensemble selection algorithms in homogeneous ensembles.

We also calculate the performance of the simpler ensemble methods `ALL` and `BSM`, which were mentioned in the previous section. For simplicity of presentation, we compare these two methods against the best (`BTFSD`) and worst (`BSACC`) ensemble selection algorithms. The mean ranks of the four methods are in ascending order `BTFSD` 1.643, `ALL` 1.821, `BSACC` 2.786 and `BSM` 3.75. Figure 4 shows the result of the Nemenyi test with 90% confidence, $q_{0.10} = 2.459$ and $CD = 1.119$.

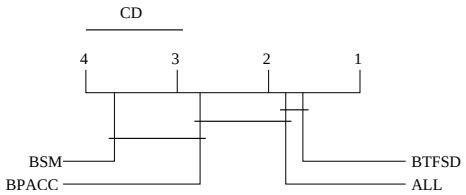


Figure 4: Graphical representation of the Nemenyi test for `ALL`, `BSM`, `BTFSD` and `BSACC`.

The results show that greedy ensemble selection may lead to worse results than using all models, if a proper algorithm isn't used. This happens, because the individual performance of the different models is approximately the same and performance gains from ensemble selection are in general limited.

5.1.2 Ensemble Size

Table 8 in the Appendix shows the size of the final ensemble that is selected from each algorithm on each dataset as well as the average size across all datasets. Figure 5 presents aggregates of the results for the different values of the search direction (a), evaluation dataset (b) and evaluation measure (c) parameters.

One interesting result is that algorithms that search in the forward direction produce smaller ensembles than those that search in the backward direction. In fact, the mean number of models selected by the `Bxxxx` algorithms (58.06) is more than double the mean number of models selected by the `Fxxxx` algorithms (24.57). Given that the direction parameter doesn't affect the predictive performance of algorithms significantly, we suggest the use of the forward direction for ensemble selection in homogeneous ensembles.

A second interesting result is that algorithms that use the selection set for evaluation produce smaller ensembles than those that use the training set. Although `xSxxx` algorithms produce very small ensembles, they are not recommended due to their significantly worse predictive performance. Concerning the evaluation measures, we notice that the mean number of models selected by all algorithms apart from `xxMAR` is approximately the same. The latter group of algorithms select on average more than twice the mean number of models that select the rest of the algorithms.

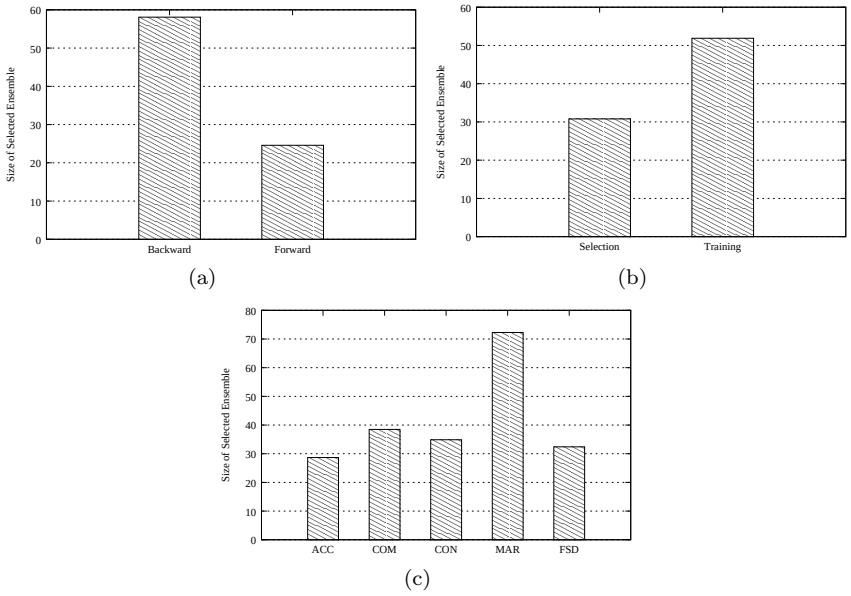


Figure 5: Mean size of selected ensemble across all datasets and respective algorithms.

5.1.3 Predictive Performance vs. Ensemble Size

Figure 6 shows the relationship between ensemble size and predictive performance. We notice that size and performance are not correlated. The 5 best performing algorithms produce ensembles of mean size between 25 and 60, while the 5 worst performing ones produce ensembles of mean size between 15 and 29.

Figures 7 to 11 depict the accuracy curves of the `xTxxx` algorithms during ensemble selection on both the evaluation and test sets for one indicative dataset (d1). Note that the final subensemble that is selected by the algorithms, is the one that corresponds to the maximum of the evaluation set accuracy curve. In the figures we observe that this maximum point corresponds to a near optimal point in the test set accuracy curve. This shows that the greedy ensemble selection algorithm manages this way to select an appropriate size for the final subensemble, which allows it to achieve high generalization performance.

5.2 Heterogeneous Models

We proceed to the analysis of the results concerning heterogeneous models following the same pattern as in the case of homogeneous models. In this case, we additionally present a discussion on the different types of algorithms that are selected in the final ensemble.

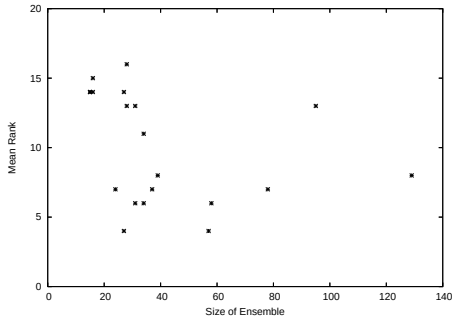
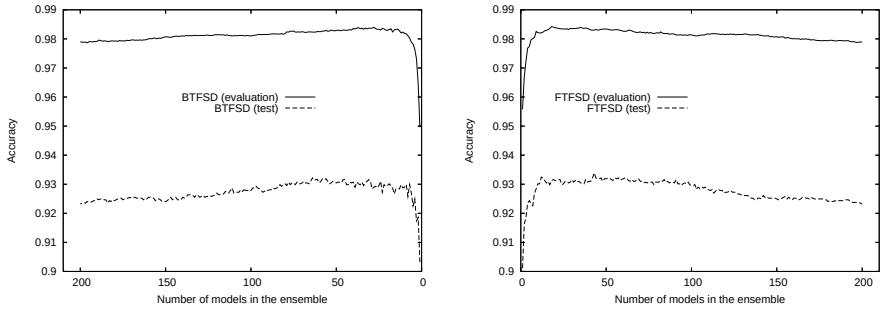
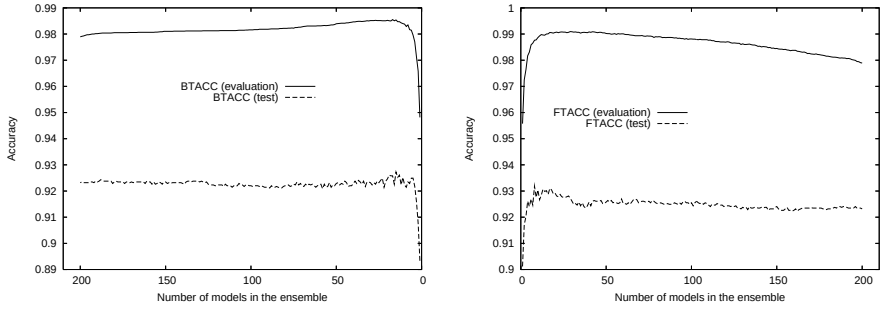


Figure 6: Mean rank against mean size of selected ensemble of all algorithms across all datasets.



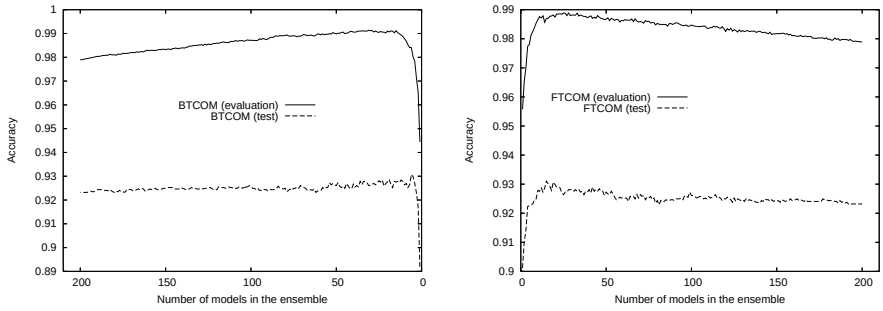
(a) (b)

Figure 7: Accuracy of `BTFSD` and `FTFSD` against the number of models.



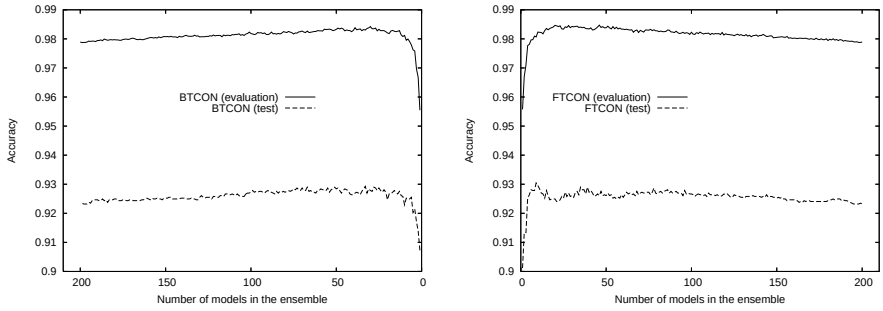
(a) (b)

Figure 8: Accuracy of `BTACC` and `FTACC` against the number of models.



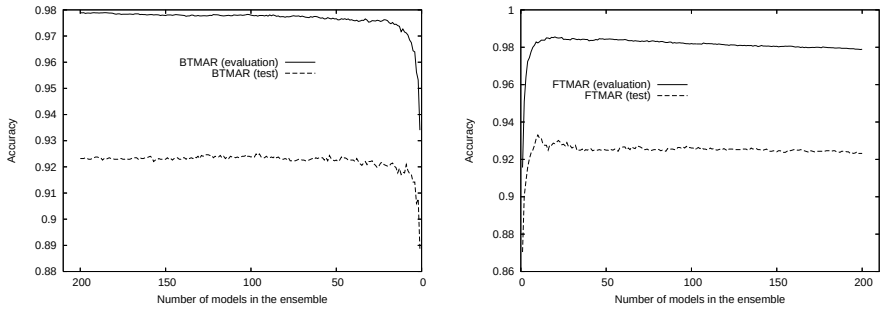
(a) (b)

Figure 9: Accuracy of `BTCOM` and `FTCOM` against the number of models.



(a) (b)

Figure 10: Accuracy of `BTCON` and `FTCON` against the number of models.



(a) (b)

Figure 11: Accuracy of `BTMAR` and `FTMAR` against the number of models.

5.2.1 Predictive Performance

Tables 9 and 10 at the Appendix present the classification accuracy and the corresponding ranks for each algorithm on each dataset. We notice that the best performing algorithms are `BSCON` and `BSFSD` occupying the first and the second place respectively.

Figure 12 depicts the aggregates of the mean ranks for the different values of the search direction (a), evaluation dataset (b) and evaluation measure (c) parameters, as well as aggregates for the different values of parameter pairs (d-f).

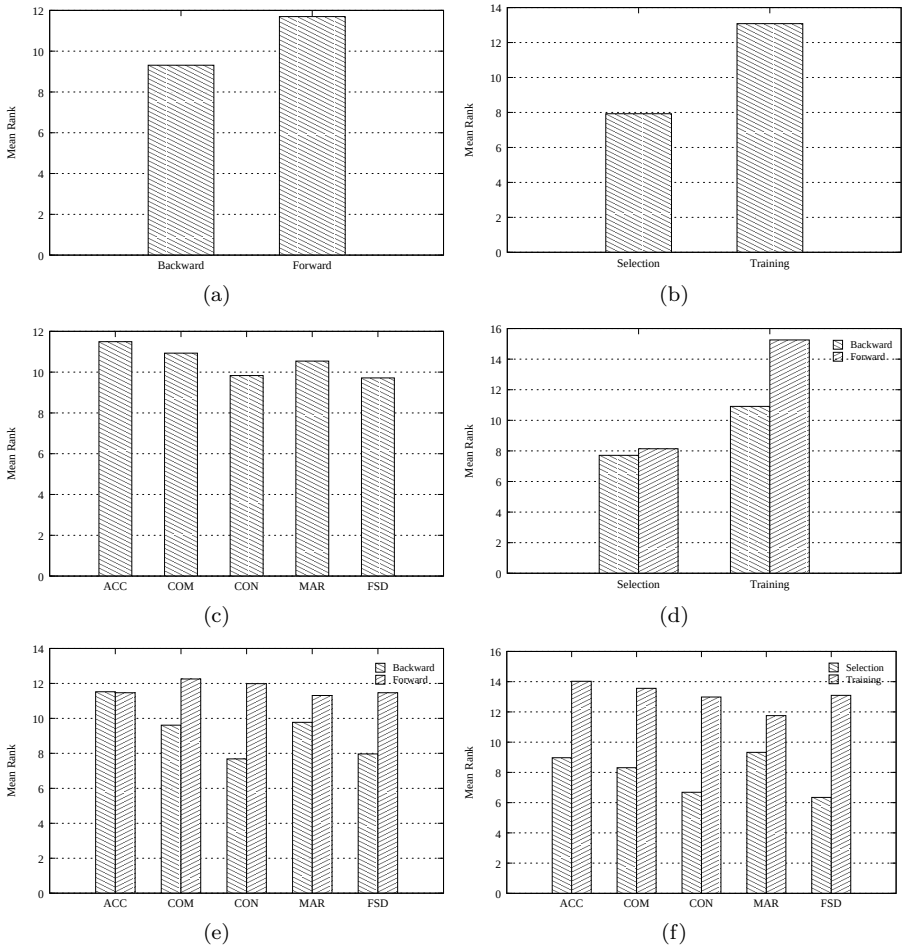


Figure 12: Mean rank across all datasets for different values of parameters and parameter pairs.

As we observe in Figure 12(a), the mean rank of the `Bxxxx` algorithms (9.307) is better than that of the `Fxxxx` algorithms (11.693). In Figure 12(e), we notice

that for some measures (CON, FSD) the difference between the Bxxxx and Fxxxx algorithms is quite large. In order to investigate the impact of the direction parameter, we perform the Wilcoxon signed-ranks test for the corresponding pairs of algorithms that use the same values for the evaluation measure and the evaluation dataset parameters, but different values for the direction parameter. For a confidence level of 95% three significant differences were detected for the following pairs of algorithms: BSACC-FSACC, BTFSD-FTFSD and BTCON-FTCON.

An interesting remark is the substantially higher performance of xSxxx algorithms compared to xTxxx algorithms, as shown in Figures 12(b) and 12(f). To verify this observation we perform the Wilcoxon signed-ranks test for the corresponding pairs of algorithms. The results of the tests show that all the differences, apart from those of the pair BSMAR-BTMAR, are significant. This leads to the conclusion that using a separate selection set improves the efficiency of the algorithms, in contrast to the case of homogeneous models. In this case, there are many different algorithms participating in the ensemble and the level of overfitting may differ among them, turning the use of a separate selection set into a necessity.

Concerning the evaluation measures, the mean ranks of the algorithms are the following: FSD 9.71, CON 9.83, MAR 10.536, COM 10.929 and ACC 11.491. Similarly to the case of homogeneous models, the proposed measure has the best average rank, which shows its robustness. The CON measure has similar performance, which shows that when we consider all combinations of correct/incorrect classification by the current ensemble and the candidate classifier gives extra strength to the measure. In contrast, algorithms that use the COM measure, which takes into account only one combination, come short of performance compared to those that use CON and FSD.

We next proceed to the Nemenyi test in order to detect significant differences among the xSxxx algorithms. Figure 13 depicts a graphical representation of the test with 90% confidence, $q_{0.10} = 2.920$ and $CD = 3.341$. We can identify two groups of algorithms. Significant differences exist between the pairs BSCON-BSACC and BSFSD-BSACC.

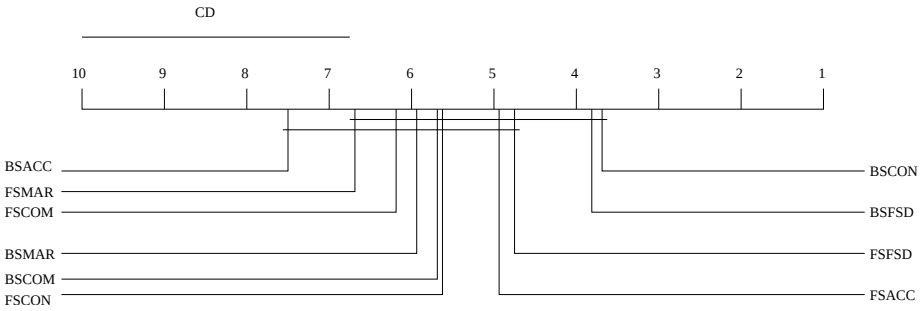


Figure 13: Graphical representation of the Nemenyi test for the xSxxx algorithms.

Figure 14 presents the graphical representation of the Nemenyi test for the

best performing algorithm BSCON, the worst performing algorithm BTACC and the methods ALL and BSM with 90% confidence, $q_{0.10} = 2.459$ and $CD = 1.119$. The mean ranks of the four algorithms in ascending order are 1.357 BSCON, 2.714 ALL, 2.964 BSM and 2.964 BTACC. The BSCON algorithm is significant better than the other algorithms, which clearly shows that ensemble selection leads to high predictive performance. We identify only one group of algorithms which includes the worst performing ones.

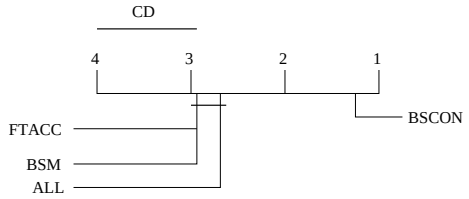


Figure 14: Graphical representation of the Nemenyi test for ALL, BSM, BTACC and BSCON.

An interesting observation is the identical performance of FTACC and BSM, which is verified by looking at the analytical accuracy results for each dataset. In addition, the algorithms FTCOM, FTCON and FTFSD also have the same performance, which means that they act similar to the BSM algorithm. This behavior can be verified by the fact that FTACC, FTCOM, FTCON and FTFSD select just one model, the one with the highest accuracy in the evaluation set, as Table 11 in the Appendix shows.

5.2.2 Ensemble Size

Table 11 at the Appendix presents the size of the selected ensemble for each algorithm on each dataset and additionally the average size across all datasets. Figure 15 shows aggregates of the mean size of the selected ensemble for the different values of the search direction (a), evaluation dataset (b) and evaluation measure (c) parameters, as well as for pairs of values of the direction and evaluation dataset parameters (d).

We first notice in Figure 15(a) that algorithms searching in the backward direction tend to produce much larger ensembles (45.54) than those searching in the forward direction (6.83). This conclusion appeared in the case of homogeneous models too. In the previous section we concluded that the direction parameter does not affect significantly the performance of the greedy ensemble selection algorithm. Based on this conclusion the recommended direction for an ensemble selection algorithm is the forward direction.

In Figure 15(b) we notice that xSxxx algorithms produce smaller ensembles (17.88), than the xTxxx algorithms (34.49). In addition, in Figure 15(d), we notice that in the xSxxx algorithms, the backward direction produces ensembles twice the size (24.90) of the ensembles that are produced from the equivalent algorithms in the forward direction (10.86). As for the evaluation measures,

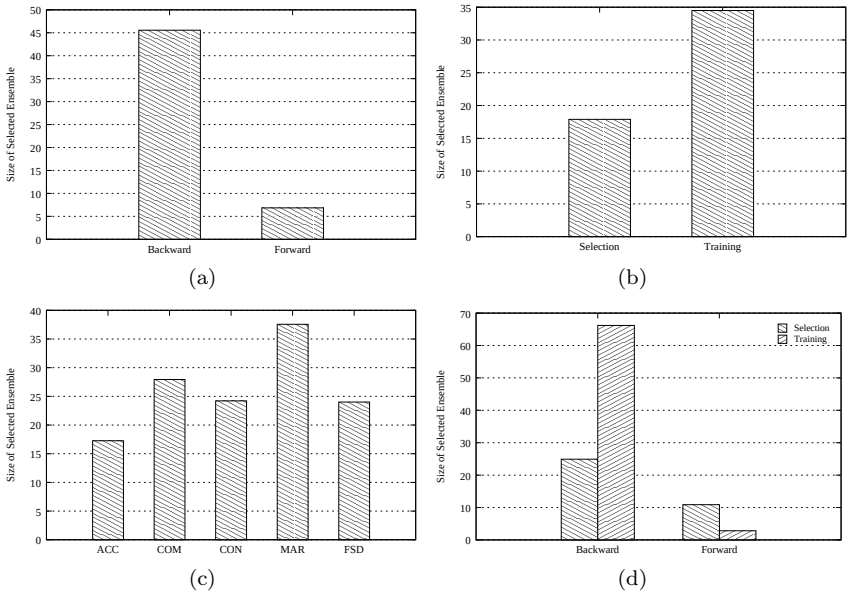


Figure 15: Mean size of selected ensemble across all datasets and respective algorithms.

the `xxMAR` algorithms produce the largest ensembles, `xxACC` the smaller, while `xxCOM`, `xxCON` and `xxFSD` select approximately the same number of models.

5.2.3 Predictive Performance vs. Ensemble Size

Figure 16 presents the relationship between the ensemble size and the predictive performance expressed in rank. The first conclusion is that size and performance are not correlated, similar to the case of homogeneous ensemble. The 5 best performing algorithms select ensembles of size between 7 and 25 while the 5 worst performing algorithms between 1 and 76.

Figures 17 to 21 show the accuracy curve both on the evaluation set and the test set during the ensemble selection for one indicative dataset (d11). Firstly, we notice that the ensemble selection procedure improves the classification accuracy substantially using a small number of models. Also, it is clearly showed that the final subensemble that is selected by the algorithms based on the evaluation dataset, corresponds to a near-optimal point on the curve of the test set. Another interesting observation is that all algorithms manage to exclude most of the models that harm the performance considerably. These models are placed at the end of the curve for each algorithm, as it can be noticed in the figures. This behavior leads to final subensembles of both small size and high predictive performance.

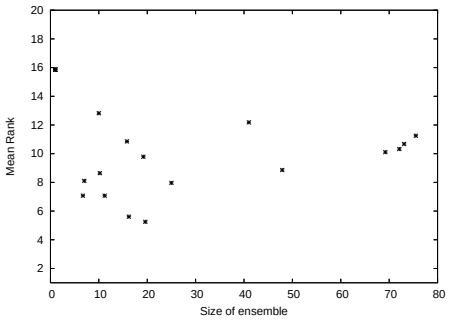
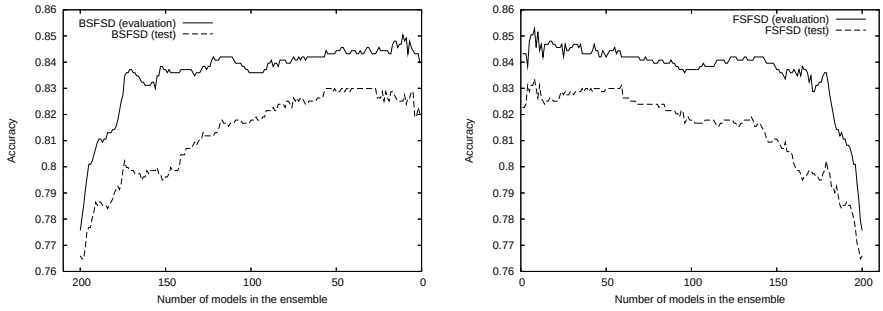
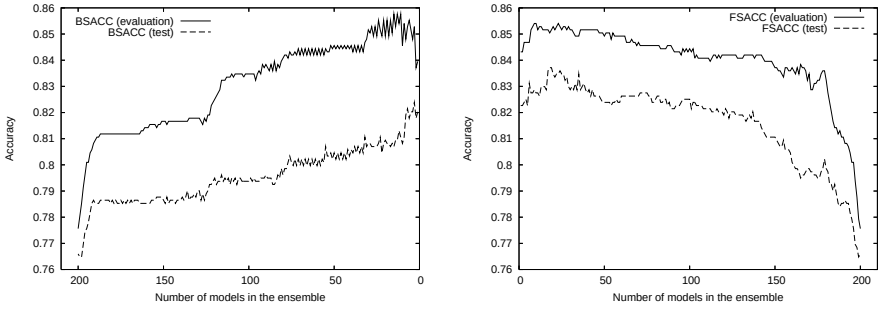


Figure 16: Mean rank against mean size of selected ensemble of all algorithms across all datasets.



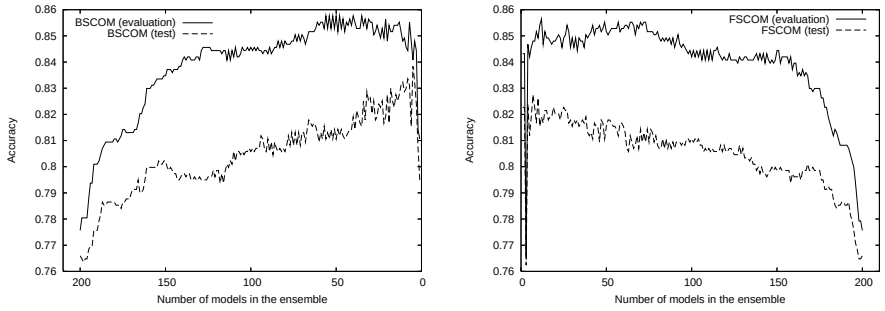
(a) (b)

Figure 17: Accuracy of `BSFSD` and `FSFSD` against the number of models.



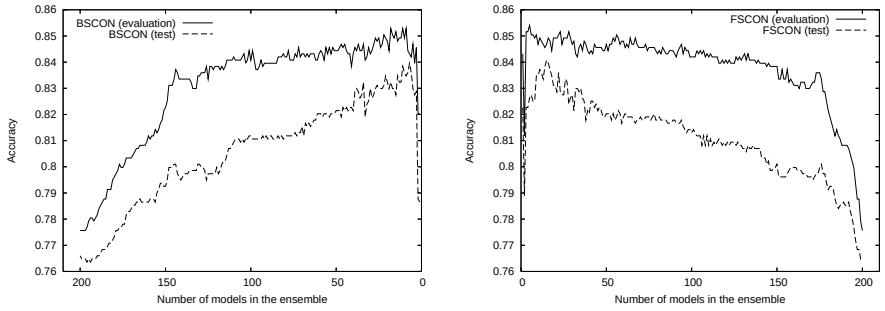
(a) (b)

Figure 18: Accuracy of `BSACC` and `FSACC` against the number of models.



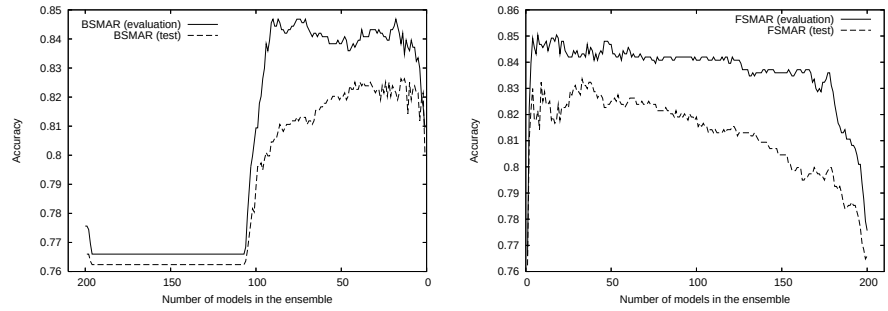
(a) (b)

Figure 19: Accuracy of `BSCOM` and `FSCOM` against the number of models.



(a) (b)

Figure 20: Accuracy of `BSCON` and `FSCON` against the number of models.



(a) (b)

Figure 21: Accuracy of `BSMAR` and `FSMAR` against the number of models.

5.2.4 Type of Models

Table 4 presents the average type of models selected from each algorithm. The results are averaged for all datasets. Figure 22 presents aggregates for the different evaluation measures across all datasets.

	NN	kNN	SVM	NB	DT
BSACC	8.8	5.7	2.1	0.0	0.1
BSCOM	10.9	5.0	7.3	0.9	2.0
BSCON	7.9	1.4	8.0	0.5	1.4
BSMAR	23.8	23.3	2.5	0.0	0.0
BSFSD	5.7	1.2	8.1	0.4	1.3
BTACC	10.4	21.9	8.6	0.1	0.2
BTCOM	19.4	30.5	24.4	0.4	1.4
BTCON	14.1	22.9	28.9	0.6	2.8
BTMAR	25.6	31.2	10.4	0.1	0.3
BTFSD	14.4	25.1	28.4	0.7	3.2
FSACC	4.6	4.8	2.6	0.1	0.4
FSCOM	3.8	2.0	2.4	0.2	0.6
FSCON	2.1	0.6	2.4	0.3	0.4
FSMAR	7.5	4.1	6.5	0.6	1.5
FSFSD	2.2	0.3	2.7	0.2	0.2
FTACC	0.2	0.8	0.0	0.0	0.0
FTCOM	0.2	0.8	0.0	0.0	0.0
FTCON	0.2	0.8	0.0	0.0	0.0
FTMAR	3.0	4.0	1.8	0.6	0.7
FTFSD	0.2	0.8	0.0	0.0	0.0
Averages	8.25	9.36	7.35	0.28	0.825

Table 4: Average number of models for all datasets, selected from each algorithm.

The `xxACC` algorithms select on average 8.3 kNN, 6.0 NN and 3.33 SVM models. The `xxCOM` algorithms exhibit a balance in the type of models that they select (8.58 kNN, 9.58 NN, 8.53 SVM). The `xxCON` and `xxFSD` algorithms select mostly SVM models at an average of 9.83 and 9.8 respectively. Finally, in the case of `xxMAR` algorithms, the kNN and NN models dominate as their average selection is 15.68 and 14.98 respectively.

5.3 Results on the *Agnostic vs. Prior Knowledge* challenge

In order to provide results for comparisons with other methods that participated in the challenge, we run the FSFSD algorithm on the challenge datasets (d11, d12, d13, d14).

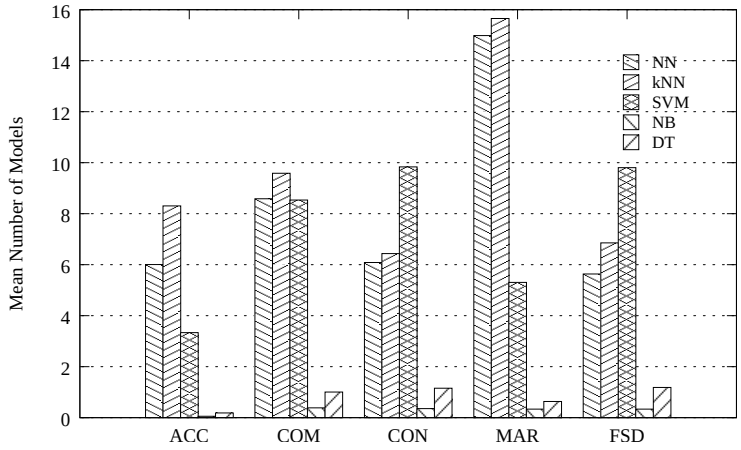


Figure 22: Aggregates for the different measures concerning the type of models that are selected.

We use a mixed ensemble of 250 models consisting of the 200 heterogeneous models that were previously described, and 50 additional models using the rule learners Ripper [Cohen, 1995] and PART [Witten and Frank, 1998], the ensemble methods boosting, bagging, and decorate [Melville and Mooney, 2004] using the C4.5 algorithm as the base classifier, and finally, the RandomForest [Breiman, 2001] algorithm. We use the training set for producing the models and the validation set for ensemble selection.

Table 5 presents the results of the FSFSD algorithm on each dataset of the *Agnostic vs. Prior Knowledge* challenge for the train, validation and test sets, using the balanced error and area under the curve measures, which are calculated automatically by the challenge organizers.

	Balanced Error			Area Under Curve		
	Train	Valid	Test	Train	Valid	Test
ada	0.1243	0.2004	0.2293	0.9711	0.8877	0.9007
gina	0.0041	0.0285	0.051	1.0	0.9932	0.9871
hiva	0.2417	0.2986	0.3141	0.9505	0.7313	0.7476
sylva	0.0059	0.0148	0.0224	0.9998	0.9994	0.9996

Table 5: Results for the FSFSD algorithm on the *Agnostic vs. Prior Knowledge* challenge datasets.

6 Conclusions

The contribution of this paper is two-fold. First of all, it presents a general framework for the greedy ensemble selection algorithm by abstracting the main

aspects of existing methods. These aspects are the direction of search, the evaluation dataset, the evaluation measure and the size of the final ensemble. The second contribution is the systematic experimental study we performed for the several options of greedy ensemble selection algorithms on both homogeneous and heterogeneous ensembles.

The analysis of the results bring out several interesting conclusions. In the homogeneous models, the use of all available data for both training and selection leads to better results than using a separate selection set. In heterogeneous models this conclusion is contradictory, as the use of a separate selection set offers significantly better performance than using all data. In both cases, the direction parameter does not affect significantly the performance and based on this conclusion we suggest the use of the forward direction as it produces smaller ensembles than the backward direction. Concerning the size of the ensemble that is selected, we concluded that selecting it based on the maximum accuracy on the evaluation set, leads to small ensembles with high predictive performance.

Appendix

This section provides the tables that present the results of the experiments for each algorithm on each dataset for both homogeneous and heterogeneous models. Due to space limitation, the tables are presented in landscape form. More specifically, Tables 6, 7 and 8 present the classification accuracies, the corresponding ranks and the size of each selected ensemble respectively for the homogeneous models. Tables 9, 10 and 11 present the classification accuracies, the corresponding ranks and the size of each selected ensemble respectively for the heterogeneous models.

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	Avrg
BSACC	90.8	52.7	78.0	99.4	96.7	99.1	89.8	73.7	85.9	83.3	84.2	92.2	95.4	99.0	87.2
BSCOM	91.9	51.6	80.0	99.4	97.2	99.2	92.7	74.4	87.6	83.5	84.3	92.5	95.6	99.2	87.8
BSCON	91.6	52.0	79.5	99.5	97.1	99.2	91.2	73.4	85.9	83.6	84.0	92.7	95.6	99.1	87.5
BSMAR	90.7	52.9	77.5	99.4	97.0	99.1	89.2	74.0	86.6	84.1	84.3	93.0	95.6	99.1	87.3
BSFSD	91.5	52.3	79.0	99.5	97.2	99.2	91.3	73.3	86.1	83.5	84.0	93.2	95.6	98.9	87.5
BTACC	92.7	53.5	76.5	99.5	97.5	99.3	93.7	76.2	89.1	83.5	84.3	93.0	95.4	99.3	88.1
BTCOM	92.8	52.9	78.0	99.5	97.7	99.3	93.9	75.6	89.2	83.8	84.3	93.8	95.7	99.2	88.3
BTCON	92.8	53.4	76.5	99.6	97.7	99.4	94.0	75.5	89.0	83.9	84.2	94.3	95.7	99.3	88.2
BTMAR	92.4	53.8	76.5	99.5	97.7	99.2	93.4	75.0	88.9	83.8	84.3	93.8	95.6	99.2	88.1
BTFSD	92.8	53.4	78.0	99.6	97.7	99.4	94.8	75.9	89.5	83.9	84.3	94.3	95.8	99.2	88.5
FSACC	91.4	53.4	77.0	99.4	96.8	99.1	90.2	73.6	86.8	83.7	84.1	91.6	95.4	99.2	87.3
FSCOM	91.2	52.6	76.0	99.4	97.0	99.1	92.3	73.3	87.3	83.5	84.2	92.5	95.4	99.2	87.4
FSCON	91.4	52.6	78.5	99.5	97.3	99.1	91.3	73.8	86.3	83.3	83.8	92.4	95.6	99.1	87.4
FSMAR	91.4	52.5	79.0	99.4	97.0	99.1	90.9	74.3	87.4	83.5	84.3	91.7	95.4	99.0	87.5
FSFSD	91.8	53.2	79.0	99.4	97.1	99.1	90.9	73.7	86.1	83.4	84.1	91.6	95.4	99.2	87.4
FTACC	92.8	53.5	76.5	99.6	97.8	99.4	94.0	76.2	87.1	83.8	84.0	93.5	95.8	99.4	88.1
FTCOM	93.0	53.7	75.5	99.6	97.7	99.3	94.2	75.4	88.1	83.5	84.1	93.7	95.8	99.3	88.1
FTCON	92.7	53.7	76.5	99.5	97.7	99.3	94.1	76.7	88.2	83.7	84.2	94.1	95.8	99.3	88.3
FTMAR	92.8	53.8	75.0	99.6	97.4	99.3	93.9	75.4	89.6	84.0	84.5	93.8	95.4	99.2	88.1
FTFSD	93.1	53.2	78.0	99.6	97.8	99.4	94.7	76.0	89.1	83.4	84.4	94.1	95.8	99.3	88.4

Table 6: Classification accuracy for each algorithm on each dataset for homogeneous models.

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	Avrg
BSACC	19.0	14.0	8.5	17.0	20.0	17.0	19.0	15.5	19.5	19.5	11.5	17.0	17.0	18.5	16.643
BSCOM	11.0	20.0	1.0	17.0	12.5	11.5	11.0	11.0	10.0	13.5	6.0	14.5	10.5	10.5	11.429
BSCON	13.0	19.0	2.0	10.0	14.5	11.5	15.0	18.0	19.5	10.0	18.0	13.0	10.5	16.0	13.571
BSMAR	20.0	12.5	11.0	17.0	17.0	17.0	20.0	13.0	15.0	1.0	6.0	11.5	10.5	16.0	13.393
BSFSD	14.0	18.0	4.0	10.0	12.5	11.5	13.5	19.5	17.5	13.5	18.0	10.0	10.5	20.0	13.75
BTACC	8.5	5.5	15.0	10.0	9.0	7.0	9.0	2.5	4.5	13.5	6.0	11.5	17.0	4.0	8.786
BTCOM	5.0	12.5	8.5	10.0	5.5	7.0	7.5	6.0	3.0	6.0	6.0	6.0	6.5	10.5	7.143
BTCON	5.0	8.0	15.0	3.5	5.5	2.5	5.5	7.0	6.0	3.5	11.5	1.5	6.5	4.0	6.071
BTMAR	10.0	1.5	15.0	10.0	5.5	11.5	10.0	10.0	7.0	6.0	6.0	6.0	10.5	10.5	8.536
BTFSD	5.0	8.0	8.5	3.5	5.5	2.5	1.0	5.0	2.0	3.5	6.0	1.5	3.0	10.5	4.679
FSACC	16.0	8.0	12.0	17.0	19.0	17.0	18.0	17.0	14.0	8.5	15.0	19.5	17.0	10.5	14.893
FSCOM	18.0	15.5	18.0	17.0	17.0	17.0	12.0	19.5	12.0	13.5	11.5	14.5	17.0	10.5	15.214
FSCON	16.0	15.5	6.0	10.0	11.0	17.0	13.5	14.0	16.0	19.5	20.0	16.0	10.5	16.0	14.357
FSMAR	16.0	17.0	4.0	17.0	17.0	17.0	16.5	12.0	11.0	13.5	6.0	18.0	17.0	18.5	14.321
FSFSD	12.0	10.5	4.0	17.0	14.5	17.0	16.5	15.5	17.5	17.5	15.0	19.5	17.0	10.5	14.571
FTACC	5.0	5.5	15.0	3.5	1.5	2.5	5.5	2.5	13.0	6.0	18.0	9.0	3.0	1.0	6.5
FTCOM	2.0	3.5	19.0	3.5	5.5	7.0	3.0	8.5	9.0	13.5	15.0	8.0	3.0	4.0	7.464
FTCON	8.5	3.5	15.0	10.0	5.5	7.0	4.0	1.0	8.0	8.5	11.5	3.5	3.0	4.0	6.643
FTMAR	5.0	1.5	20.0	3.5	10.0	7.0	7.5	8.5	1.0	2.0	1.0	6.0	17.0	10.5	7.179
FTFSD	1.0	10.5	8.5	3.5	1.5	2.5	2.0	4.0	4.5	17.5	2.0	3.5	3.0	4.0	4.857

Table 7: Corresponding ranks for each algorithm on each dataset for homogeneous models.

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	Avrg
BSACC	13.1	30.8	38.0	77.5	12.3	74.8	14.7	14.6	23.1	38.8	23.6	18.0	9.0	8.0	28.3
BSCOM	23.2	26.1	47.0	92.9	24.1	68.3	25.8	19.8	48.2	42.7	15.7	13.0	17.0	15.0	34.2
BSCON	14.6	23.7	38.0	74.9	14.0	27.6	20.9	9.8	24.2	41.8	16.6	20.0	3.0	106.0	31.0
BSMAR	108.4	80.0	7.0	93.0	93.7	94.0	96.8	72.3	128.3	135.8	101.9	109.0	14.0	199.0	95.2
BSFSD	18.3	16.5	14.0	92.9	21.2	50.2	23.4	15.9	33.9	32.5	19.3	47.0	4.0	6.0	28.2
BTACC	21.0	60.0	32.0	30.3	25.2	8.6	70.5	47.0	117.8	60.5	44.6	17.0	8.0	11.0	39.5
BTCOM	36.0	57.2	40.0	33.3	96.2	14.0	144.3	135.7	170.9	153.5	38.8	143.0	6.0	37.0	78.9
BTCON	37.8	67.6	62.0	39.0	52.8	14.2	85.1	81.6	135.2	127.5	40.2	21.0	14.0	43.0	58.6
BTMAR	147.0	160.0	102.0	67.1	153.9	52.8	131.3	128.8	154.6	163.2	158.5	189.0	3.0	199.0	129.2
BTFSD	32.9	65.9	34.0	44.8	55.6	22.4	114.9	75.5	133.8	114.4	38.3	46.0	15.0	12.0	57.5
FSACC	12.5	21.4	16.0	1.6	6.5	2.1	16.3	17.2	33.3	35.4	18.8	10.0	5.0	19.0	15.3
FSCOM	20.5	16.9	17.0	6.0	10.8	7.5	18.4	23.0	25.9	39.9	19.5	10.0	7.0	7.0	16.3
FSCON	15.1	18.4	37.0	4.6	16.4	4.8	14.6	10.3	21.4	27.6	14.0	11.0	3.0	13.0	15.0
FSMAR	24.0	38.5	19.0	6.0	21.4	17.2	20.0	27.3	31.5	38.2	41.5	28.0	7.0	67.0	27.6
FSFSD	16.4	18.3	49.0	2.9	12.5	3.8	12.6	18.2	23.3	28.9	19.6	12.0	4.0	11.0	16.6
FTACC	25.5	60.8	19.0	6.4	12.6	6.9	38.3	33.8	17.6	84.1	37.4	65.0	8.0	27.0	31.5
FTCOM	21.6	53.9	19.0	11.6	20.5	9.4	18.8	28.9	13.9	48.4	31.7	45.0	6.0	13.0	24.4
FTCON	25.2	54.1	27.0	7.6	24.7	6.7	24.1	41.8	21.3	101.4	34.5	76.0	4.0	39.0	34.8
FTMAR	22.5	59.2	46.0	28.0	26.4	12.3	46.2	52.0	37.9	85.3	39.7	47.0	5.0	11.0	37.0
FTFSD	24.6	57.7	24.0	8.5	23.7	9.5	21.6	31.2	19.9	56.7	34.3	41.0	16.0	13.0	27.2

Table 8: Size of each selected ensemble for homogeneous models.

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	Avrg
BSACC	98.8	53.7	74.6	94.0	97.4	99.2	97.9	81.3	94.0	85.5	80.7	93.3	96.7	99.4	89.0
BSCOM	99.0	54.1	75.0	99.3	98.5	99.2	98.7	82.1	95.4	85.2	81.1	93.5	96.6	99.3	89.8
BSCON	99.4	55.1	75.2	99.3	98.3	99.4	98.7	84.0	96.0	86.0	83.2	93.7	96.2	99.5	90.3
BSMAR	98.1	56.1	75.1	93.5	98.3	99.2	98.5	82.3	93.5	85.8	80.5	91.0	96.7	99.5	89.2
BSFSD	99.3	56.5	75.2	99.3	98.3	99.4	98.8	83.8	95.8	86.0	82.5	93.5	96.4	99.5	90.3
BTACC	90.6	48.0	74.2	90.7	98.5	96.9	9.54	80.5	93.8	73.8	77.2	94.0	97.0	97.8	86.3
BTCOM	89.4	45.6	75.8	94.4	98.9	99.2	76.9	82.9	94.3	84.1	77.1	92.4	96.6	99.4	86.2
BTCON	91.4	47.6	75.9	90.8	98.3	99.2	98.3	83.2	94.4	84.7	75.8	93.3	96.7	99.4	87.8
BTMAR	89.6	51.7	75.6	94.4	98.3	98.1	95.2	82.4	94.5	85.9	82.5	91.9	96.6	99.3	88.3
BTFSD	92.1	47.3	75.8	90.8	98.3	99.2	98.2	83.1	94.4	83.9	76.6	93.3	96.7	99.4	87.8
FSACC	99.7	53.7	75.0	99.5	98.1	99.2	98.4	74.6	97.5	86.7	82.9	93.5	96.4	99.5	89.6
FSCOM	99.7	52.7	71.5	99.3	98.1	99.2	98.4	75.7	97.5	85.0	81.9	93.7	96.4	99.5	89.2
FSCON	99.7	56.1	71.5	99.3	98.1	99.2	97.9	75.7	97.5	85.4	82.6	93.7	96.4	99.5	89.5
FSMAR	98.9	53.1	74.0	99.5	97.8	99.2	97.9	81.4	95.3	84.7	82.1	93.7	96.7	99.2	89.5
FSFSD	99.7	51.7	74.0	99.5	98.1	99.2	98.4	78.7	97.5	86.2	83.4	93.7	96.4	99.5	89.7
FTACC	99.4	42.9	69.5	90.7	98.1	95.5	95.8	64.5	99.0	72.7	75.8	80.8	95.2	97.5	84.1
FTCOM	99.4	42.9	69.5	90.7	98.1	95.5	95.8	64.5	99.0	72.7	75.8	80.8	95.2	97.5	84.1
FTCON	99.4	42.9	69.5	90.7	98.1	95.5	95.8	64.5	99.0	72.7	75.8	80.8	95.2	97.5	84.1
FTMAR	99.0	43.2	73.9	94.8	98.1	96.1	96.9	69.1	98.2	72.7	77.7	80.8	96.9	98.5	85.4
FTFSD	99.4	42.9	69.5	90.7	98.1	95.5	95.8	64.5	99.0	72.7	75.8	80.8	95.2	97.5	84.1

Table 9: Classification accuracy for each algorithm on each dataset for heterogeneous models.

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	Avrg
BSACC	14.0	6.5	10.0	12.0	20.0	8.0	11.0	10.0	18.0	7.0	10.0	11.0	5.0	9.5	10.857
BSCOM	11.5	5.0	8.5	6.0	2.5	8.0	2.5	8.0	12.0	9.0	9.0	8.0	9.0	12.5	7.964
BSCON	7.0	4.0	5.5	6.0	6.5	1.5	2.5	1.0	10.0	3.5	2.0	4.0	16.0	4.0	5.25
BSMAR	15.0	2.5	7.0	13.0	6.5	8.0	4.0	7.0	20.0	6.0	11.0	15.0	5.0	4.0	8.857
BSFSD	10.0	1.0	5.5	6.0	6.5	1.5	1.0	2.0	11.0	3.5	5.5	8.0	13.0	4.0	5.607
BTACC	18.0	12.0	11.0	18.0	2.5	15.0	18.0	11.0	19.0	15.0	13.0	1.0	1.0	16.0	12.179
BTCOM	20.0	15.0	2.5	10.5	1.0	8.0	20.0	5.0	17.0	13.0	14.0	13.0	9.0	9.5	11.25
BTCON	17.0	13.0	1.0	14.5	6.5	8.0	8.0	3.0	15.5	11.5	18.0	11.0	5.0	9.5	10.107
BTMAR	19.0	10.5	4.0	10.5	6.5	14.0	19.0	6.0	14.0	5.0	5.5	14.0	9.0	12.5	10.679
BTFSD	16.0	14.0	2.5	14.5	6.5	8.0	9.0	4.0	15.5	14.0	15.0	11.0	5.0	9.5	10.321
FSACC	2.5	6.5	8.5	2.0	14.0	8.0	6.0	15.0	7.5	1.0	3.0	8.0	13.0	4.0	7.071
FSCOM	2.5	9.0	15.5	6.0	14.0	8.0	6.0	13.5	7.5	10.0	8.0	4.0	13.0	4.0	8.643
FSCON	2.5	2.5	15.5	6.0	14.0	8.0	11.0	13.5	7.5	8.0	4.0	4.0	13.0	4.0	8.107
FSMAR	13.0	8.0	12.5	2.0	19.0	8.0	11.0	9.0	13.0	11.5	7.0	4.0	5.0	14.0	9.786
FSFSD	2.5	10.5	12.5	2.0	14.0	8.0	6.0	12.0	7.5	2.0	1.0	4.0	13.0	4.0	7.071
FTACC	7.0	18.5	18.5	18.0	14.0	18.5	15.5	18.5	2.5	18.0	18.0	18.0	18.5	18.5	15.857
FTCOM	7.0	18.5	18.5	18.0	14.0	18.5	15.5	18.5	2.5	18.0	18.0	18.0	18.5	18.5	15.857
FTCON	7.0	18.5	18.5	18.0	14.0	18.5	15.5	18.5	2.5	18.0	18.0	18.0	18.5	18.5	15.857
FTMAR	11.5	16.0	14.0	9.0	14.0	16.0	13.0	16.0	5.0	18.0	12.0	18.0	2.0	15.0	12.821
FTFSD	7.0	18.5	18.5	18.0	14.0	18.5	15.5	18.5	2.5	18.0	18.0	18.0	18.5	18.5	15.857

Table 10: Corresponding ranks for each algorithm on each dataset for heterogeneous models.

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	Avrg
BSACC	21.7	13.0	16.6	3.0	1.0	13.0	50.9	35.3	9.6	17.0	15.0	16.0	1.0	9.0	15.8
BSCOM	43.0	29.0	16.8	10.0	8.0	14.0	15.9	27.8	19.2	40.0	56.0	18.0	19.0	34.0	25.0
BSCON	27.9	19.0	28.3	7.0	16.0	41.0	14.1	21.5	14.1	29.0	18.0	2.0	26.0	11.0	19.6
BSMAR	44.7	22.0	53.7	29.0	38.0	80.0	69.3	80.5	43.6	47.0	90.0	66.0	1.0	7.0	47.9
BSFSD	31.4	7.0	18.2	10.0	13.0	41.0	14.9	24.5	8.0	10.0	11.0	7.0	25.0	6.0	16.2
BTACC	99.9	49.0	9.9	4.0	11.0	4.0	97.4	178.2	58.6	10.0	10.0	39.0	1.0	4.0	41.0
BTCOM	142.9	18.0	72.2	10.0	12.0	122.0	140.5	184.6	59.7	24.0	9.0	143.0	60.0	60.0	75.5
BTCON	155.7	30.0	69.1	6.0	16.0	86.0	120.7	181.4	50.0	22.0	6.0	119.0	58.0	50.0	69.2
BTMAR	95.2	48.0	69.9	28.0	164.0	93.0	94.7	175.4	79.5	53.0	29.0	55.0	1.0	38.0	73.1
BTFSD	159.3	31.0	69.9	6.0	16.0	87.0	141.5	181.6	49.6	23.0	8.0	127.0	59.0	51.0	72.1
FSACC	1.0	45.0	11.0	2.0	2.0	5.0	3.0	2.0	5.0	55.0	9.0	12.0	2.0	4.0	11.2
FSCOM	1.0	59.0	1.0	4.0	1.0	1.0	9.0	6.0	10.0	36.0	12.0	1.0	1.0	1.0	10.2
FSCON	1.0	38.0	1.0	9.0	1.0	1.0	1.0	12.0	7.0	16.0	5.0	1.0	1.0	5.0	7.0
FSMAR	9.7	17.0	38.3	6.0	2.0	43.0	8.8	27.1	11.2	48.0	7.0	3.0	16.0	33.0	19.2
FSFSD	1.0	3.0	18.0	1.0	1.0	1.0	3.0	21.0	13.0	18.0	8.0	1.0	1.0	4.0	6.7
FTACC	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FTCOM	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FTCON	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FTMAR	4.8	2.0	44.5	17.0	2.0	23.0	8.9	2.1	3.2	2.0	3.0	3.0	11.0	14.0	10.0
FTFSD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 11: Size of each selected ensemble for heterogeneous models.

References

A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Ensemble diversity measures and their application to thinning. *Information Fusion*, 6 (1):49–62, 2005.

L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–40, 1996.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning*, page 18, 2004.

R. Caruana, A. Munson, and A. Niculescu-Mizil. Getting the most out of ensemble selection. In *Sixth International Conference in Data Mining (ICDM '06)*, 2006.

W.W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

T. G. Dietterich. Machine-learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.

T. G. Dietterich. Ensemble Methods in Machine Learning. In *Proceedings of the 1st International Workshop in Multiple Classifier Systems*, pages 1–15, 2000.

W. Fan, F. Chu, H. Wang, and P. S. Yu. Pruning and dynamic scheduling of cost-sensitive ensembles. In *Eighteenth national conference on Artificial intelligence*, pages 146–151. American Association for Artificial Intelligence, 2002.

Giorgio Giacinto, Fabio Roli, and Giorgio Fumera. Design of effective multiple classifier systems by clustering of classifiers. In *15th International Conference on Pattern Recognition, ICPR 2000*, pages 160–163, 3–8 September 2000.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

L.I. Kuncheva and C.J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51: 181–207, 2003.

Aleksandar Lazarevic and Zoran Obradovic. Effective pruning of neural network classifiers. In *2001 IEEE/INNS International Conference on Neural Networks, IJCNN 2001*, pages 796–801, 15–19 July 2001.

D. Margineantu and T. Dietterich. Pruning adaptive boosting. In *Proceedings of the 14th International Conference on Machine Learning*, pages 211–218, 1997.

G. Martinez-Munoz and A. Suarez. Aggregation ordering in bagging. In *International Conference on Artificial Intelligence and Applications (IASTED)*, pages 258–263. Acta Press, 2004.

P. Melville and R. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion: Special Issue on Diversity in Multiclassifier Systems*, pages 99–111, 2004.

P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

I. Partalas, G. Tsoumakas, I. Katakis, and I. Vlahavas. Ensemble pruning via reinforcement learning. In *4th Hellenic Conference on Artificial Intelligence (SETN 2006)*, pages 301–310, May 18–20 2006.

Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. Focused ensemble selection: A diversity-based method for greedy ensemble selection. In *Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 117–121. IOS Press, 2008.

Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. An ensemble uncertainty aware measure fordirectedhillclimbing ensemble pruning. *Machine Learning*, 81:257–282, 2010.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5: 197–227, 1990.

E. K. Tang, P. N. Suganthan, and X. Yao. An analysis of diversity measures. *Machine Learning*, 65(1):247–271, 2006.

G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective Voting of Heterogeneous Classifiers. In *Proceedings of the 15th European Conference on Machine Learning, ECML2004*, pages 465–476, 2004.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

I.H. Witten and E. Frank. Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning, ICML98*, pages 144–151, 1998.

I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

Yi Zhang, Samuel Burer, and W. Nick Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338, 2006.