Classifier ensemble methods in feature selection

Article *in* Neurocomputing · September 2020

DOI: 10.1016/j.neucom.2020.07.113

1 author:

Hakan Ezgi Kızılöz
Türk Hava Kurumu Üniversitesi
23 PUBLICATIONS **221** CITATIONS

SEE PROFILE

Classifier ensemble methods in feature selection

Hakan Ezgi Kiziloz

University of Turkish Aeronautical Association, Ankara, Turkey

Abstract

Feature selection has become an indispensable preprocessing step in an expert system. Improving the feature selection performance could guide such a system to make better decisions. Classifier ensembles are known to improve performance when compared to the use of a single classifier. In this study, we aim to perform a formal comparison of different classifier ensemble methods on the feature selection domain. For this purpose, we compare the performances of six classifier ensemble methods: a greedy approach, two average-based approaches, two majority voting approaches, and a meta-classifier approach. In our study, the classifier ensemble involves five machine learning techniques: Logistic Regression, Support Vector Machines, Extreme Learning Machine, Naïve Bayes, and Decision Tree. Experiments are carried on 12 well-known datasets, and results with statistical tests are provided. The results indicate that ensemble methods perform better than single classifiers, yet, they require a longer execution time. Moreover, they can minimize the number of features better than existing ensemble algorithms, namely Random Forest, AdaBoost, and Gradient Boosting, in a less amount of time. Among ensemble methods, the greedy based method performs well in terms of both classification accuracy and execution time.

Keywords: Feature selection, Multiobjective optimization, Machine learning, Classifier ensemble

1. Introduction and background

The increase in data of real-world problems may be useful to extract valuable information. However, it can also make data analysis challenging. Data mining and machine

Email address: hakanezgi@etu.edu.tr (Hakan Ezgi Kiziloz)

Preprint submitted to Neurocomputing *doi:10.1016/j.neucom.2020.07.113*

learning techniques may suffer from a massive amount of data, also known as the curse of dimensionality [1]. It is crucial to clean the data before processing to build efficient machine learning models. There are many preprocessing tools to decrease the amount of data, one of which is feature selection [2]. Feature selection is the task of reducing the number of features by removing irrelevant or redundant features from the data. Eliminating these features leads to better learning performance and decreases computation time [3].

Feature selection methods can be filter-based, wrapper-based, or embedded [4, 5]. Filter-based methods measure relevance between features using designated methods, such as the chi-square test or correlation coefficient. Therefore, the best feature subset is selected independently from a learning algorithm. Wrapper-based methods, on the contrary, incorporate learning algorithms in the feature selection process. For this reason, wrapper-based methods are computationally more expensive than filter-based methods. However, they mostly generate more useful feature subsets as the subsets are evaluated in the training phase [6]. Embedded methods are a combination of both, which integrates classification into feature selection. In this study, we apply a wrapper-based method by combining multiple classifiers in the training phase to find the most informative feature subsets.

Feature selection requires multiobjective optimization as there are two objectives; (i) minimizing the number of features while (ii) maximizing the learning performance. A multiobjective optimization problem may have multiple solutions rather than one solution. These solutions are called non-dominated since they cannot dominate each other in both objectives [7]. A solution is called to dominate another one if it performs better in one objective while performing better or the same in the remaining ones.

There is a wide range of studies about feature selection. Nevertheless, it is still a hot topic [5]. A recent survey by Xue et al. [8] discusses the challenges of feature selection and presents contributions of various feature selection methods. Evolutionary algorithms are commonly used for feature selection [9–11]. In recent years, nature-inspired evolutionary algorithms are very prominent, and they are successfully applied in many optimization problems, including feature selection [12–16]. These algorithms aim to simulate the behaviour of species in nature to find the optimal solution. For this reason, researchers mathematically

model animal behaviours in certain events such as hunting of whales or wolves, food procurement of bees or ants, and mating of fireflies. Due to the multiobjective nature of the problem, we require a multiobjective selection algorithm. In this study, we employ NSGA-II [17], a genetic algorithm specialized for multiobjective optimization, with respect to its popularity and proven performance in this domain [7].

Combining predictive models to improve learning performance is known as the ensemble approach [18]. Experiment results show that an effective ensemble model can perform better than a single model [19, 20]. The ensemble approach may also eliminate overfitting as well as boosting the overall performance [21]. Moreover, this approach is commonly used to tackle the class imbalance problem [22, 23]. A study by Tsymbal et al. [24] shows that ensemble feature selection methods can increase the accuracy and the diversity of the solutions. In our study, we exploit multiple classifiers to develop a robust ensemble model for the feature selection process.

The main contributions of this study can be described as follows. Feature selection is an essential preprocessing step for an expert system. Wrapper-based feature selection methods utilize a classifier to evaluate the performance of each subset. Classifier ensembles combine predictive models of different classifiers. This approach is known to improve classification performance, and there exists a wide range of ensemble methods and algorithms. However, up to our best knowledge, ensemble methods have never been analyzed extensively in the feature selection domain. Hence, the most fruitful ensemble method is yet to be discovered. In this study, we investigate the effects of using different classifier ensemble methods in the feature selection domain. For this purpose, we analyze six different classifier ensemble methods and compare their performances in four metrics: accuracy, number of features, number of non-dominated solutions, and execution time. We utilize five different machine learning techniques in the classifier ensembles and report the test results on 12 well-known datasets. We also compare the results with existing ensemble algorithms and with the state-of-the-art algorithms. Moreover, we elaborate on our findings with statistical test results.

The rest of the manuscript is organized as follows. In Section 2, the multiobjective selection algorithm and machine learning techniques are introduced. Also, model implementation

is described in detail in Section 2. The experimental environment and results are shared in Section 3. Concluding remarks and possible future works are given in Section 4.

2. Model

The model requires a combination of several steps, including a multiobjective selection algorithm, application of machine learning techniques, and ensemble methods. After introducing these steps separately in the following subsections, we will provide a general look to describe how these steps are combined in this study.

2.1. Multiobjective selection algorithm

We employ the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) in our search for the most promising feature subset. As its name refers, NSGA-II is a variation of the genetic algorithm that is specialized for multiobjective optimization.

The NSGA-II algorithm is initialized with generating the initial population as similar to a regular genetic algorithm. Each individual in the population is represented with a chromosome, an example of which is given in Figure 1. In this study, all chromosomes in the initial population are generated randomly. Each chromosome has the length that is equal to the total number of features in the dataset. Each gene in a chromosome can either be 1 or 0, indicating that its associated feature is selected or not, respectively. Therefore, the individuals are representations of the feature subsets throughout the study.

After the initial population is generated, each individual in the population is evaluated one after another. Individual evaluation is described in detail in Section 2.4. Next, the individuals in the population are sorted with the non-dominated sorting algorithm which splits the individuals into fronts as follows: The first front consists of the individuals that are

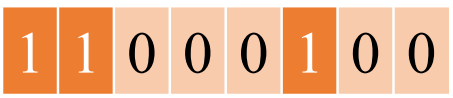


Figure 1: A sample chromosome of a dataset having 8 features.

not dominated by any other individual, the second front consists of the individuals that are only dominated by the individuals in the first front, the third front consists of the individuals that are only dominated by the individuals in the first two fronts, and so on. Hence, any individual in the second front is worse than any individual in the first front, but it is better than any individual in the third or later fronts. Crowding distance helps to determine the better individual in case a comparison between individuals within the same front is required. It is calculated for each front separately. At this point, the algorithm is ready for starting its search for the most promising subset.

First, we apply binary tournament to two randomly selected candidate individuals, and determine the better individual as the first parent. The second parent is determined with the same approach. Then, we apply half uniform crossover to mate the selected parents and generate two child individuals from the couple. Every child individual may mutate according to a mutation probability. Finally, both child individuals are added to the population. The process of mating parents to create new child individuals continues until the number of child individuals is equal to the number of parents. At this point, the population has twice the size of the initial population. After evaluating all child individuals, the individuals in the population are sorted using the non-dominated sorting algorithm, and the better half is selected as the new population. This whole process is called a generation.

The NSGA-II algorithm executes for a predetermined number of generations. Eventually, it reports the individuals in the first front, the non-dominated individuals, as the solution set.

In this study, population size is selected as 40, and the NSGA-II algorithm executes for 60 generations. The crossover probability is 1.0, and the mutation probability is 0.02.

2.2. Applied machine learning techniques

We employ five machine learning techniques in this study: Logistic Regression (LR), Support Vector Machines (SVM), Extreme Learning Machine (ELM), Naïve Bayes (NB), and Decision Tree (DT).

To summarize, LR performs classification using a probabilistic model. SVM constructs

a hyperplane that maximizes the distance between the classes. ELM is a feedforward neural network with a single hidden layer. NB is the simplest probabilistic classifier that applies Bayes theorem assuming independence between features. Finally, DT builds a tree where inner nodes represent features, branches hold distinct values of related features, and leaf nodes denote classes.

We use Python implementations of these machine learning techniques. The scikit-learn library [1] provides implementations for LR, SVM, NB, and DT. For ELM, the hpelm library [2] is utilized. We note that the main purpose of the study is not improving the classification performance of a specific feature selection problem, but comparing the performances of different classification ensemble methods with each other. Therefore, parameter tuning becomes an optional step, and it is not implemented in this study. The default parameters for the SVM, NB, and DT classifiers remained unchanged. The C and *solver* parameters of the LR classifier are set to $1e5$ and *liblinear*, respectively. For the ELM classifier, the number of hidden neurons is set to 20, and the sigmoid activation function is used.

2.3. Classifier ensemble methods

We implement six classifier ensemble methods in this study, as described below. These methods are used for determining the ensemble's prediction, as described in Section 2.4.

Greedy (Grd): As its name refers, this method uses a greedy approach, i.e. the classifier having the highest validation accuracy for the individual determines the ensemble's predictions. For example, if ELM has the highest validation accuracy for an individual, then the ELM classifier's predictions on the test instances are set as the ensemble's predictions. In the meanwhile, another classifier could achieve a higher validation accuracy for another individual. Hence, that classifier would determine the ensemble's predictions for that individual.

Averaging (Avg): Normally, the averaging method is used in regression problems due to its nature. For each instance, the prediction of each regressor is averaged and set as

[1] scikit-learn library: `https://scikit-learn.org/`

[2] hpelm library: `https://pypi.org/project/hpelm/`

the ensemble's prediction. In classification problems, however, this approach is flawed. For example, if two classifiers predict class 0 for an instance, and the remaining three classifiers predict class 1; their average would be 0.6, which does not relate to any specific class. For this reason, we use a variation of this method to make it applicable to classification problems. In this study, this method simply calculates the average test accuracy values obtained by each classifier and uses it as one of the objective values, i.e. accuracy, in the optimization phase. As soon as the NSGA-II algorithm produces the final population, we calculate the classification performance as similar to the *Grd* method, i.e. the classifier having the highest validation accuracy determines the ensemble's prediction on the test data.

Weighted averaging (Wavg): This is a variation of the *Avg* method, where the objective value is selected as the weighted average of classifier test accuracies, rather than the average. Weights for each classifier are calculated by their classification performance on the validation set. Similar to the *Avg* method, this method uses the *Grd* approach to calculate the classification performance of the individual, as soon as the NSGA-II algorithm terminates.

Majority voting (Mv): This method considers the predictions of the classifiers as votes for classes and the class receiving the majority of the votes is selected as the ensemble's prediction. For example, if two classifiers predict class 0 for a test instance, and the other three classifiers predict class 1, then the prediction of the ensemble is determined as class 1.

Weighted majority voting (Wmv): Similar to the relationship between *Avg* and *Wavg* methods, the *Wmv* method is a variation of the *Mv*. Here, each classifier's vote has a weight that is equal to its classification performance on the validation set. Again, the prediction of the *Wmv* ensemble is determined as the class having more votes. For example, assuming that only one classifier with a weight of 0.95 votes for class 0 for a test instance, whereas all remaining classifiers with a cumulative weight of 0.80 vote for class 1, then the prediction ensemble is set as class 0.

Blending (Bln): In this method, class predictions of each classifier are combined into features, e.g. LR prediction becomes the first feature, SVM prediction the second, and so on. These features are then used for training a new meta-classifier. In order not to bias the classification performance, the predictions for the validation set are used for training the

meta-classifier. Test instances are used only for testing. Due to its execution speed, DT is selected as the meta-classifier.

2.4. Model implementation

The implemented model is depicted in Figure 2. Briefly, the NSGA-II algorithm searches for the most promising feature subset regarding both objectives through generations. The inner-working mechanism of the NSGA-II algorithm is given in Section 2.1, in detail. At each generation, the fitness values of each individual in the population is evaluated. For each individual, the data is filtered in such a way that only the individual's selected features are included. Then, the filtered data is sent to the classifiers. The data consists of training, validation, and test sets. The five classifiers (LR, SVM, ELM, NB, and DT) execute in parallel and train on the training set. Consecutively, they predict the instances in the validation and test sets. The predictions are combined with respect to the selected classifier ensemble method (described in Section 2.3), and the classification performance of the individual is calculated accordingly.

3. Experiments and results

The experiments are executed on a high-performance cluster machine, consisting of a total of 64 CPUs (each having a 2.8 GHz clock rate) and 256 GB memory. The code is implemented in Python 3.6.

In this study, we held the experiments on 12 datasets. Eleven of the datasets were retrieved from the well-known UCI Machine Learning repository [3], whereas the last one (Financial) was obtained from a study by Pacheco et al. [25]. Details of these datasets are provided in Table 1.

Four of the datasets (CT, NU, C4, and WF) consist of more than two actual classes. Those datasets were subsampled into two classes by eliminating the classes having a fewer number of instances.

[3]UCI Machine Learning repository: `https://archive.ics.uci.edu/ml/datasets.php`

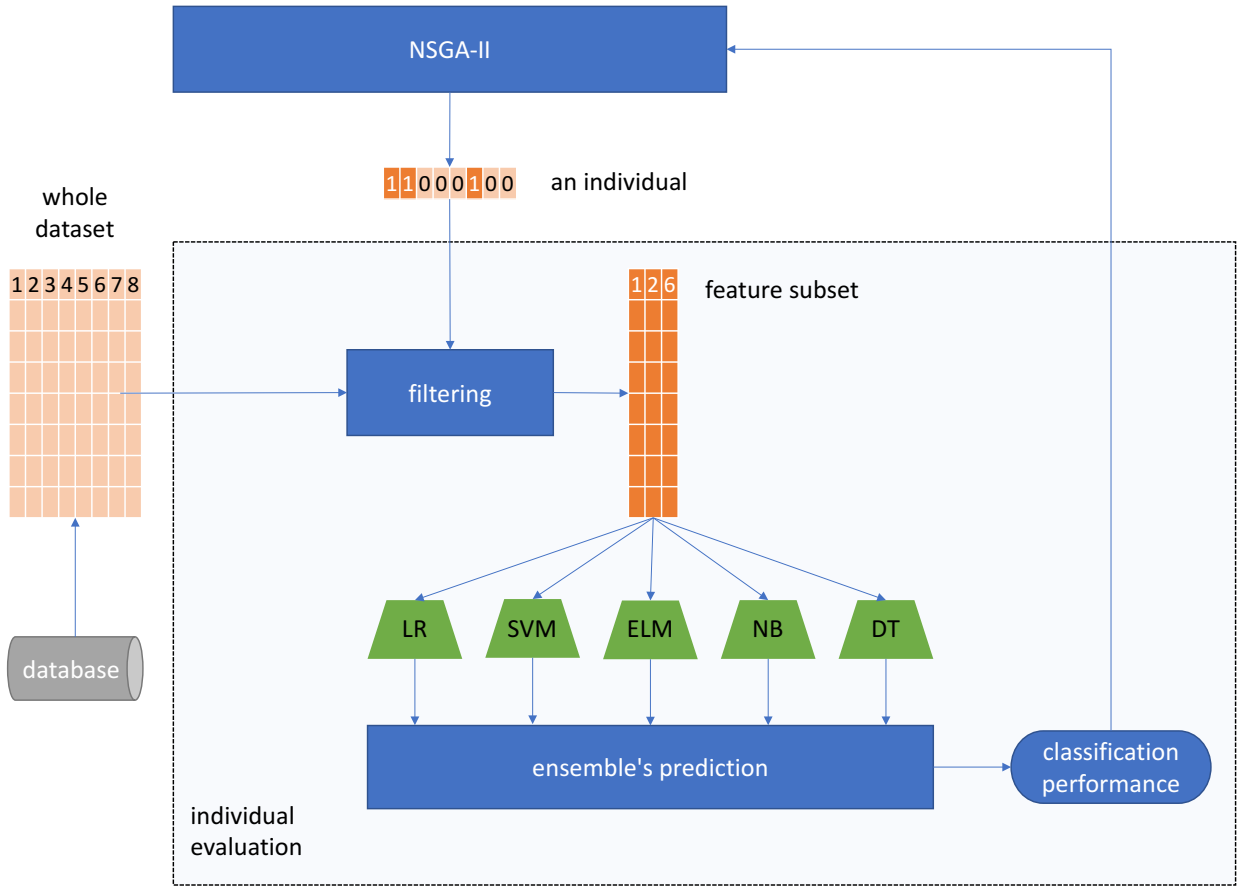


Figure 2: The model used in this study.

In order to eliminate bias when measuring the classification performance, it is common to apply k-fold cross-validation. k-fold cross-validation splits the data into k folds and uses each fold to test a model trained by all remaining folds. Overall classification performance is the average classification performance of each fold. However, applying k-fold cross-validation on a dataset having a high number of instances is costly in terms of training time. As a result, we opted-out k-fold cross-validation; rather, we used an approximation to it. In this study, we employed a similar approach that was already used in other studies [7, 25–27]; in fact, we used the same datasets that were used in [27]. In this approach, 10 training sets were randomly selected, but with respect to the instance proportions per class. For each training set, 10 test sets were randomly selected after excluding the instances that were already selected for the training set. In this study, we also need a validation set, which was

Table 1: Details of the datasets used in the experiments.

Dataset	**Dataset ID**	**# of features**	**Actual # of classes**	**# of instances**	**Training set size (x10)**	**Validation set size (x10)**	**Cumulative test set size (x10)**
Covertype	CT	54	7	581,012	600	200	1800
Mushrooms	MR	22	2	8124	1300	200	1800
Spambase	SB	57	2	4601	600	200	1800
Nursery	NU	8	5	12,960	400	200	1800
Connect-4 Opening	C4	42	3	67,557	1200	200	1800
Waveform	WF	40	3	5000	400	200	1800
Financial	FI	93	2	17,108	1000	200	1800
Pima Indian Diabetes	PM	8	2	768	268	200	1800
Breast Cancer	BC	9	2	699	199	100	900
Ionosphere	IO	34	2	351	101	50	450
Wisconsin Breast Cancer	WBC	30	2	569	169	80	720
Musk (Version 2)	MU	168	2	6598	400	200	1800

inexistent in the dataset. Therefore, we used the first set of the 10 test sets as the validation set. Sizes of the training, validation, and cumulative test sets are also provided in Table 1.

After training, validation, and test sets were determined, the values in the datasets were standardized. In this process, only the training sets were used when formulating a translation for the values. The translation was applied to all the instances in the training, validation, and test sets.

We present the experiment results in four subsections. In the following subsection, Section 3.1, we present the performance results of ensemble methods along with the performance results of single classifiers. Also, we provide a detailed comparative analysis with statistical tests. Then, in the next subsection, Section 3.2, we compare the ensemble methods with other well-known ensemble algorithms, i.e. Random Forest, AdaBoost, and Gradient Boosting. In the third subsection, Section 3.3, we compare the performance of the *Grd* method with the performances obtained in our previous studies. Finally, in Section 3.4, we compare the maximum accuracy obtained in the *Grd* method with the state-of-the-art algorithms.

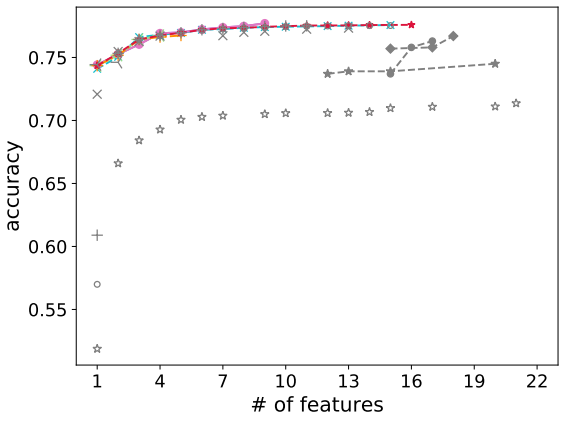
3.1. Comparison with single classifiers

The non-dominated solution set of each classifier and classifier ensemble method is plotted for each dataset in Figure 3. The only exception is the NU dataset, for which all classifiers and classifier ensemble methods could find 1.0 accuracy having only 1 feature selected. The solutions that are closer to the top left corner are better than others since the accuracy is higher (top), and the number of features is lower (left). In some of the figures, *Bln* and *Grd* methods stand out among other classifier ensemble methods, yet, we cannot draw an immediate conclusion. On the other hand, even though it is not easy to discriminate a classifier ensemble method from another one by just looking at these figures, it is clear that classifier ensembles perform better than classifiers when executed alone.

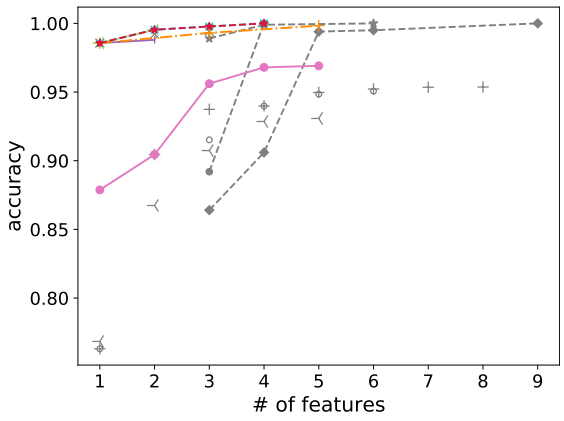
We used statistical tests to provide a more formal comparison of the methods. Friedman test indicates significant difference between the results of all single classifiers and classifier ensemble methods in terms of accuracy ($\chi^2(10) = 54.605, p < 0.001$), feature size ($\chi^2(10) = 33.238, p < 0.001$), number of solutions ($\chi^2(10) = 37.238, p < 0.001$), and execution time ($\chi^2(10) = 104.561, p < 0.001$). A post hoc Wilcoxon test is applied to make pairwise comparisons between each method. The comparison results are provided in Figure 4. In the figure, a light-red color means that the difference is not significant, whereas a green color indicates a significant difference. The saturation of the green color is a representative of significance level, i.e. a darker green indicates higher significance. The mean value for each method is provided as a numerical value at the diagonal.

Pairwise comparison results on the accuracy (see Figure 4a) indicate that *Bln* outperforms all ensemble and single classifier methods with a significant difference, except for the *Grd* and SVM methods. As similar to *Bln*, *Grd* achieves better accuracy values than all single classifier methods, except for SVM. In comparison with the ensemble methods, it performs better than *Avg* and *Wavg*, but its difference with *Mv* and *Wmv* is not significant. *Grd* is followed by *Wmv*, which is followed by *Mv*. Finally, *Wavg* and *Avg* perform similarly, and both perform better than DT only.

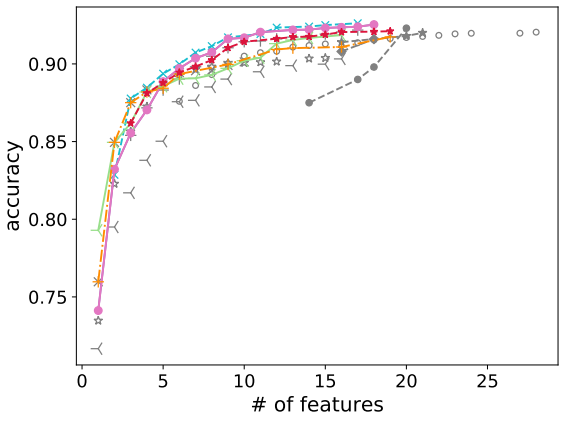
In contrast with their performance on the accuracy, *Avg* and *Wavg* can reduce the number of features effectively, with respect to the pairwise feature size comparisons (see Figure 4b).



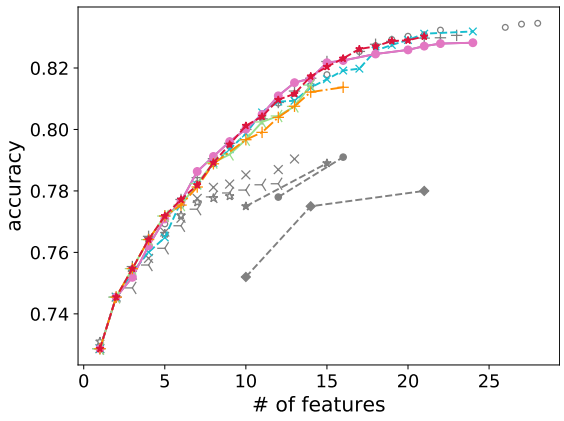
(a) CT



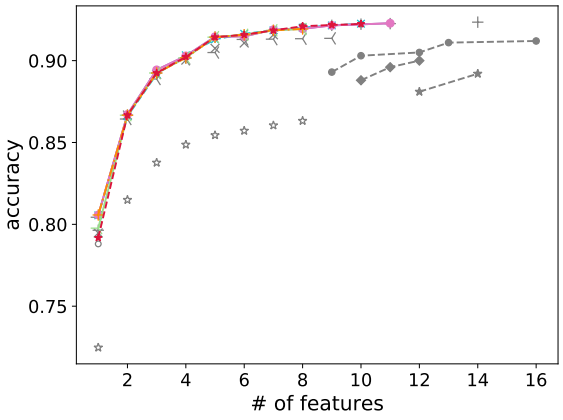
(b) MR



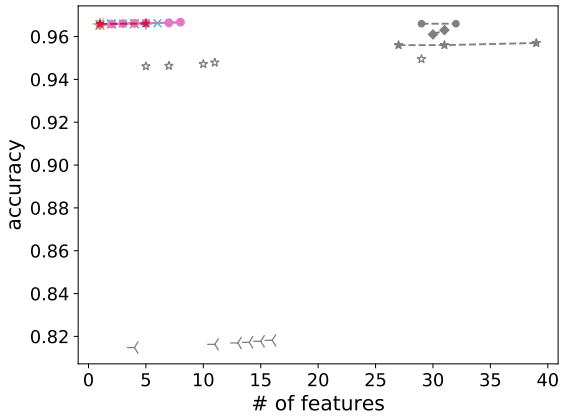
(c) SB



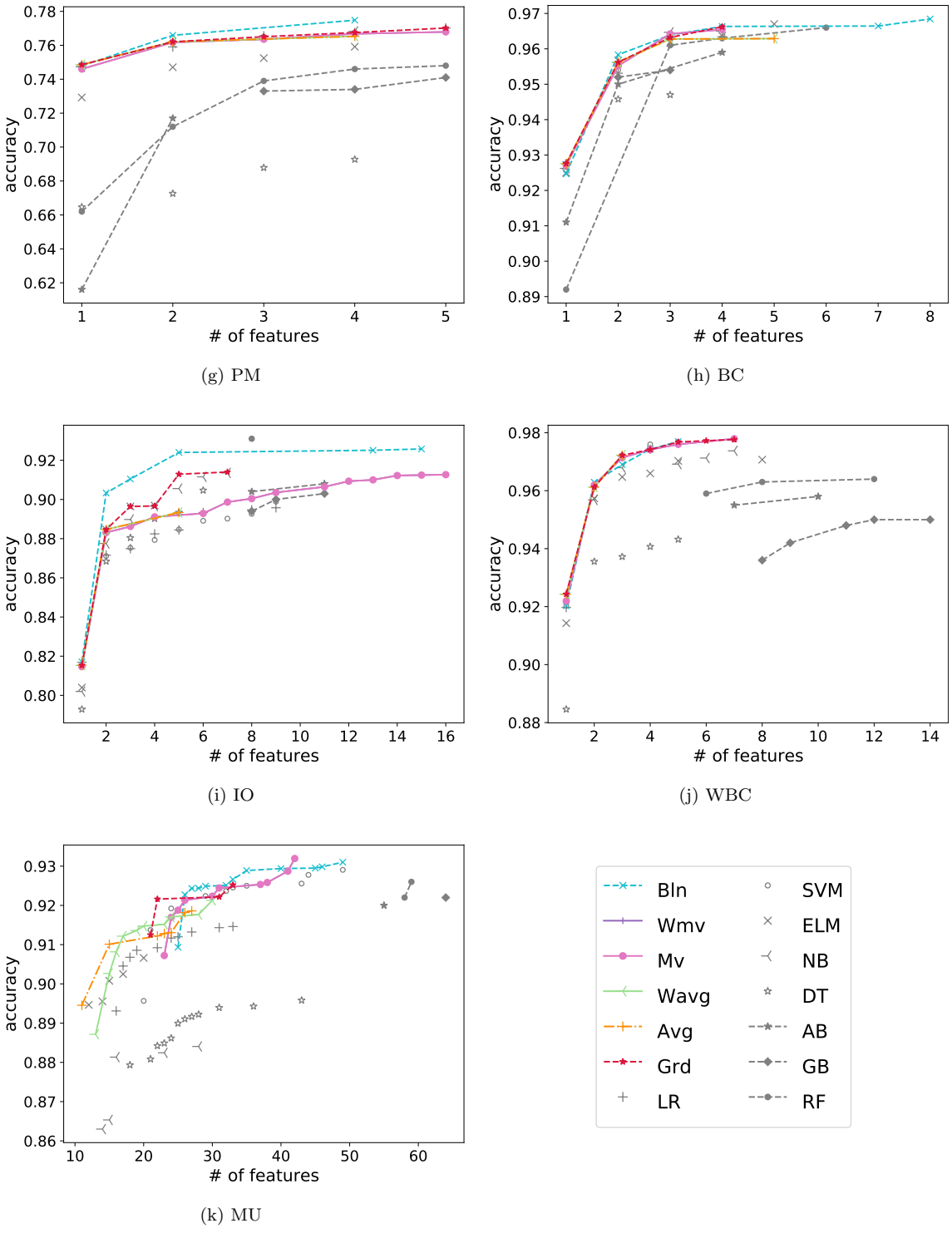
(d) C4



(e) WF



(f) FI



(g) PM

(h) BC

(i) IO

(j) WBC

(k) MU

Figure 3: The non-dominated solutions of the classifiers and ensemble methods on all datasets.

Wavg outperforms all other ensemble methods, and it is followed by *Avg*. Both also perform better than LR and SVM. The remaining ensemble methods have no significant differences with any other method, except for ELM, which significantly performs better than *Grd*, *Mv*, and *Bln*. These results are consistent with the results of the pairwise comparison of the number of solutions (see Figure 4c). Again, the difference between the number of solutions of *Avg* and *Wavg* with other ensemble methods is significant; stating that *Avg* and *Wavg* have less number of solutions in the solution set. Other differences between other ensemble methods are statistically not significant.

Finally, on pairwise comparison of the methods in terms of execution time (see Figure 4d), it is clear that all single classifiers execute faster than all ensemble methods. The only exception is SVM, which executes significantly faster than *Mv* and *Wmv* only. Among the ensemble methods, *Wmv* is the slowest method. It is followed by *Mv* and *Bln*, in respective order. *Grd*, *Avg*, and *Wavg* methods perform similarly: all faster than *Bln*, but with no significant difference in-between.

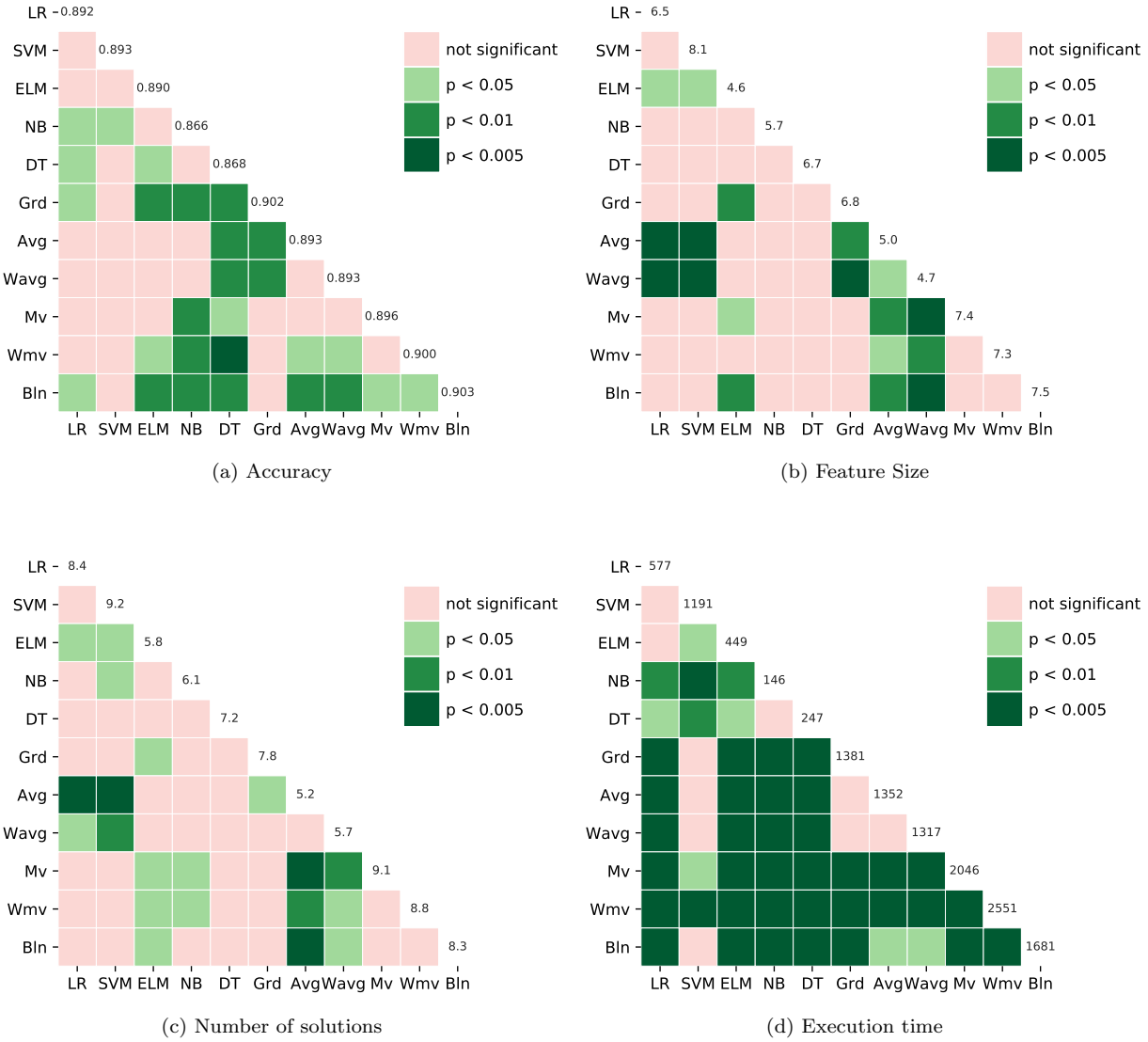
3.2. Comparison with existing ensemble algorithms

In this section, we compare the performances of the ensemble methods against existing bagging (Random Forest) and boosting (AdaBoost and Gradient Boosting) ensemble algorithms.

Random Forest (RF) constructs multiple decision trees on various sub-samples of the dataset and combines their predictions. AdaBoost (AB) combines multiple classifiers, also called as weak learners, and iteratively updates the weights of their contributions to the result. At each iteration, it increases the weights of the learners with incorrect classifications and decreases the weights of the learners with correct ones. Gradient Boosting (GB) works iteratively, similar to AB, but it does optimization according to a loss function.

We used the scikit-learn library for the implementations of these algorithms, with 100 predictors and the default parameters. These three algorithms are executed separately, each one as the single classifier of the system. The non-dominated solution sets of these algorithms are also presented in Figure 3 to facilitate a better comparison.

Statistical tests support the incompetencies of these algorithms, as compared to the ensemble methods. Friedman test indicates significant difference between the results in terms of accuracy ($\chi^2(8) = 26.709, p < 0.001$), feature size ($\chi^2(8) = 50.773, p < 0.001$), number of solutions ($\chi^2(8) = 50.058, p < 0.001$), and execution time ($\chi^2(8) = 76.556, p < 0.001$).



(a) Accuracy

(b) Feature Size

(c) Number of solutions

(d) Execution time

Figure 4: Pairwise comparison of each classifier and ensemble methods in terms of accuracy, feature size, number of solutions, and execution time. Numbers on the diagonals represent the average values for each method.

Accordingly, we applied post hoc Wilcoxon tests to make pairwise comparisons between the methods. Pairwise comparisons indicate that the existing algorithms perform similarly in terms of maximizing the accuracy (RF=0.895, AB=0.889, and GB=0.892). The *Bln* method achieves higher accuracy values than the AB and GB algorithms. On the other hand, all the ensemble methods outperform the existing algorithms in terms of minimizing the number of features (RF=14.7, AB=14.5, and GB=15.6), the number of solutions (RF=2.8, AB=2.3, and GB=2.8), and execution time (RF=3293, AB=3660, and GB=5590). Exceptional cases

Table 2: Multiobjective comparison of the *Grd* method with previous studies.

(a) C4

F. size	**NSGA II + Grd**	**MTLBO MD + LR**	**NSGA II + LR**
1	0.729	0.729	-
2	0.745	0.746	-
3	0.755	0.755	-
4	0.764	0.764	0.765
5	0.772	0.772	0.772
6	0.777	0.777	0.778
7	0.782	0.784	0.785
8	0.789	0.791	0.791
9	0.795	0.796	0.796
10	0.801	0.797	0.800
11	0.804	0.799	0.802
12	0.810	0.799	0.807
13	0.812	-	0.810
14	0.817	-	0.812
15	0.820	-	0.815
16	0.823	0.805	0.819
17	0.826	-	0.821
18	0.827	-	-
19	0.829	-	0.822
20	0.829	-	0.822
21	0.830	-	-
22	-	-	0.825
23	-	-	0.826
24	-	-	0.826

(b) SB

F. size	**NSGA II + Grd**	**MTLBO MD + LR**	**NSGA II + LR**
1	-	0.782	-
2	-	0.835	0.835
3	0.862	0.857	0.857
4	0.881	0.871	0.871
5	0.888	0.879	0.883
6	0.894	0.891	0.890
7	0.898	0.896	0.902
8	0.903	0.905	0.906
9	0.910	0.910	0.910
10	0.914	0.911	0.914
11	-	0.913	0.915
12	0.916	-	0.916
13	0.917	0.915	0.917
14	0.918	-	0.919
15	0.919	-	0.919
16	0.921	-	0.920
17	-	-	0.920
18	0.921	-	0.920
19	0.921	-	-

(c) CT

F. size	**NSGA II + Grd**	**MTLBO MD + LR**	**NSGA II + LR**
1	0.744	0.743	0.743
2	0.753	0.753	0.753
3	0.764	0.764	0.764
4	0.767	0.767	0.767
5	0.770	0.770	0.770
6	0.772	0.772	0.772
7	0.773	0.773	0.772
8	0.774	0.773	0.773
9	0.774	0.774	0.774
10	0.775	-	0.774
11	0.775	-	-
12	0.775	-	-
13	0.775	-	0.774
14	0.776	-	0.775
16	0.776	-	-

(d) MU

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
21	0.913	0.907	-
22	0.922	0.910	-
23	-	0.912	-
24	-	0.914	-
25	-	0.916	-
26	-	0.917	-
27	-	0.918	-
28	-	0.919	-
31	0.922	-	-
33	0.925	-	-

(e) WF

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
1	0.792	0.789	0.868
2	0.867	0.868	0.893
3	0.892	0.893	0.902
4	0.902	0.902	0.915
5	0.914	0.915	0.917
6	0.916	0.917	0.919
7	0.919	0.919	0.921
8	0.921	0.921	0.922
9	0.922	0.922	0.923
10	0.922	0.923	0.923

(f) MR

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
1	0.986	0.763	0.750
2	0.995	0.905	0.867
3	0.998	0.937	0.937
4	1.000	0.940	0.937
5	-	0.949	0.945
6	-	0.950	0.946
7	-	-	-
8	-	-	0.946
9	-	-	0.949

(g) IO

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
1	0.815	0.816	-
2	0.885	0.872	0.872
3	0.896	0.876	0.876
4	0.897	0.882	0.878
5	0.913	0.886	0.886
6	-	0.886	0.889
7	0.914	0.887	-
8	-	0.890	-

(h) WBC

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
1	0.924	0.920	0.920
2	0.961	0.961	0.961
3	0.972	0.971	0.972
4	0.974	0.975	0.975
5	0.977	-	-
6	0.977	-	-
7	0.978	-	-

(i) BC

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
1	0.928	0.927	0.927
2	0.956	0.953	0.953
3	0.963	0.963	0.963
4	0.966	0.963	0.963
5	-	0.963	0.963
6	-	-	-

(j) PM

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
1	0.749	0.747	0.747
2	0.762	0.760	0.760
3	0.765	0.766	0.766
4	0.768	0.768	0.768
5	0.770	0.771	0.770

(k) FI

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
1	0.966	0.966	0.966
3	-	0.967	0.966
5	0.966	-	-

(l) NU

F. size	NSGA II + Grd	MTLBO MD + LR	NSGA II + LR
1	1.000	1.000	1.000

are as follows. The *Wavg* method has a similar number of solutions with all ensemble algorithms. Also, the *Wmv* method requires similar execution times with the AB and GB algorithms.

3.3. Comparison with previous studies

In this section, we compare the results with the results from our previous studies in Table 2. For this purpose, we present the non-dominated solution sets of the *Grd* method, due to its high accuracy and low execution time, side-by-side with the non-dominated solution sets obtained in our previous studies [7, 27]. The same dataset splits were used in all three studies. Moreover, the selection algorithm (NSGA-II) is the same as one of them [7]. Therefore, a change in the result is a direct consequence of the classifier ensemble method. Over the 12 datasets, a total of 83 solutions exist where we can directly compare the accuracy of *Grd* with at least one other method. *Grd* can find the maximum accuracy in 53 (64%) of these direct comparisons. This amount increases to 78 (94%) if we include the solutions with up to a 0.005 decrease in accuracy as compared to the maximum. As a result, *Grd* is a viable method in terms of approximating the best solution.

3.4. Comparison with state-of-the-art algorithms

We conclude this section by comparing the maximum accuracy results of the *Grd* method with the state-of-the-art. We only compare the maximum accuracy values, as those studies do not provide accuracy values per feature size. Two of these studies [28, 29] utilize genetic algorithm with ELM, whereas the remaining one [30] uses MTLBO-MD with LR as the multiobjective selection algorithm and the classifier.

The results of the matching datasets are provided in Table 3. The *Grd* method achieves the highest accuracy values on nine datasets out of 12. Its pairwise comparison with FS-ELM and HGEFS algorithms are inconclusive with respect to the small number of comparisons (seven and three, respectively). On the other hand, the Wilcoxon signed-rank test indicates a significant difference when it is compared with MTLBO-MD ($W = 1, p = 0.007$). Therefore, the *Grd* method remains as a viable option.

Table 3: Accuracy comparison of the *Grd* method with the state-of-the-art.

Dataset	*Grd*	FS-ELM [29]	MTLBO-MD [30]	HGEFS [28]
CT	**0.776**	-	0.772	-
MR	**1.000**	-	0.951	-
SB	**0.921**	0.907	0.914	-
NU	**1.000**	-	**1.000**	-
C4	**0.830**	-	0.826	-
WF	**0.922**	0.713	0.919	-
FI	**0.966**	-	**0.966**	-
PM	0.770	**0.779**	0.771	-
BC	0.966	**0.977**	0.962	-
IO	0.914	**0.934**	0.889	0.913
WBC	**0.978**	0.977	0.973	0.971
MU	**0.925**	0.780	0.911	0.881

4. Conclusion

An expert system consists of several layers, including data collection, data processing, model learning, and decision making. The purity of the data plays a vital role in its performance. In other words, an expert system performs better if the data is cleansed from redundant or irrelevant information. Clean data also decreases the training time of the model. As a result, feature selection, the task of removing redundant or irrelevant features from the data, has become a crucial preprocessing step for an expert system.

In wrapper-based feature selection, a multiobjective search algorithm searches for the best representative of features, while a classifier is used to evaluate the performances of the candidates. In the literature, it is common to report the performances of a model, evaluated by different classifiers separately. Even though the same methodology is applied, different classifiers could often produce different solutions. Hence, combining these classifiers could produce fruitful results. Combining predictions of different classifiers to predict instances is called classifier ensembling.

In this study, we compare the performances of six different classifier ensemble meth-

ods in the feature selection domain, with empirical results. The tested ensemble methods are Greedy (*Grd*), Averaging (*Avg*), Weighted averaging (*Wavg*), Majority voting (*Mv*), Weighted majority voting (*Wmv*), and Blending (*Bln*). These ensemble methods are used to combine the predictions of five classifiers, namely Logistic Regression, Support Vector Machines, Extreme Learning Machine, Naïve Bayes, and Decision Tree. We hold the experiments on 12 well-known datasets and provide comparison results on four different metrics: accuracy, number of features, number of solutions, and execution time.

Experiment results show that using classifier ensembles increases accuracy, but they execute slower than single classifiers. *Bln* method performs best in terms of accuracy, while *Grd* can achieve similar results in a timely manner. *Wavg* can reduce the number of features in almost the same amount of time with *Grd*, but with a trade-off on the accuracy.

Experiment results show that the difference between execution times of four ensemble methods and the ensemble's slowest classifier, i.e. SVM, is statistically insignificant. Therefore, these ensemble methods are scalable as much as the slowest executing classifier. In addition, we approach the problem as a binary classification problem. Theoretically, all the ensemble methods can be applied to datasets having a higher number of classes. As a result, we believe that these ensemble methods are scalable in terms of the number of features and classes.

Possible future work for this study could be the joint optimization of the classifier ensemble, i.e. rather than including all five classifiers into the ensemble, the best ensemble of classifiers could be dynamically searched along with the search for the best representative subset.

References

[1] M. Verleysen, D. François, The curse of dimensionality in data mining and time series prediction, in: International Work-Conference on Artificial Neural Networks, Springer, 2005, pp. 758–770.

[2] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of machine learning research 3 (Mar) (2003) 1157–1182.

[3] H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering, IEEE Transactions on knowledge and data engineering 17 (4) (2005) 491–502.

[4] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (1) (2014) 16–28.

[5] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, ACM Computing Surveys (CSUR) 50 (6) (2018) 94.

[6] Y. Li, T. Li, H. Liu, Recent advances in feature selection and its applications, Knowledge and Information Systems 53 (3) (2017) 551–577.

[7] A. Deniz, H. E. Kiziloz, T. Dokeroglu, A. Cosar, Robust multiobjective evolutionary feature subset selection algorithm for binary classification using machine learning techniques, Neurocomputing 241 (2017) 128–146.

[8] B. Xue, M. Zhang, W. N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Transactions on Evolutionary Computation 20 (4) (2015) 606–626.

[9] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm, in: Feature extraction, construction and selection, Springer, 1998, pp. 117–136.

[10] I.-S. Oh, J.-S. Lee, B.-R. Moon, Hybrid genetic algorithms for feature selection, IEEE Transactions on pattern analysis and machine intelligence 26 (11) (2004) 1424–1437.

[11] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, A survey on new generation metaheuristic algorithms, Computers & Industrial Engineering 137 (2019) 106040.

[12] M. Mafarja, A. Qasem, A. A. Heidari, I. Aljarah, H. Faris, S. Mirjalili, Efficient hybrid nature-inspired binary optimizers for feature selection, Cognitive Computation (2019) 1–26.

[13] S. Arora, P. Anand, Binary butterfly optimization approaches for feature selection, Expert Systems with Applications 116 (2019) 147–160.

[14] E. Emary, H. M. Zawbaa, A. E. Hassanien, Binary grey wolf optimization approaches for feature selection, Neurocomputing 172 (2016) 371–381.

[15] M. Mafarja, I. Aljarah, H. Faris, A. I. Hammouri, A.-Z. Ala'M, S. Mirjalili, Binary grasshopper optimisation algorithm approaches for feature selection problems, Expert Systems with Applications 117 (2019) 267–286.

[16] H. Rao, X. Shi, A. K. Rodrigue, J. Feng, Y. Xia, M. Elhoseny, X. Yuan, L. Gu, Feature selection based on artificial bee colony and gradient boosting decision tree, Applied Soft Computing 74 (2019) 634–642.

[17] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE transactions on evolutionary computation 6 (2) (2002) 182–197.

[18] D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, Journal of artificial intelligence research 11 (1999) 169–198.

[19] R. M. Bell, Y. Koren, C. Volinsky, All together now: A perspective on the netflix prize, Chance 23 (1) (2010) 24–29.

[20] Y. Saeys, T. Abeel, Y. Van de Peer, Robust feature selection using ensemble feature selection techniques, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 313–325.

[21] H.-F. Yu, H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. McKenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, Y.-H. Wei, et al., Feature engineering and classifier ensemble for kdd cup 2010, in: KDD Cup, 2010.

[22] K. Yang, Z. Yu, X. Wen, W. Cao, C. P. Chen, H.-S. Wong, J. You, Hybrid classifier ensemble for imbalanced data, IEEE Transactions on Neural Networks and Learning Systems 31 (4) (2019) 1387–1400.

[23] P. Yang, W. Liu, B. B. Zhou, S. Chawla, A. Y. Zomaya, Ensemble-based wrapper methods for feature selection and class imbalance learning, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2013, pp. 544–555.

[24] A. Tsymbal, S. Puuronen, D. W. Patterson, Ensemble feature selection with the simple bayesian classification, Information fusion 4 (2) (2003) 87–100.

[25] J. Pacheco, S. Casado, L. Núñez, A variable selection method based on tabu search for logistic regression models, European Journal of Operational Research 199 (2) (2009) 506–511.

[26] A. Unler, A. Murat, A discrete particle swarm optimization method for feature selection in binary classification problems, European Journal of Operational Research 206 (3) (2010) 528–539.

[27] H. E. Kiziloz, A. Deniz, T. Dokeroglu, A. Cosar, Novel multiobjective tlbo algorithms for the feature subset selection problem, Neurocomputing 306 (2018) 94–107.

[28] X. Xue, M. Yao, Z. Wu, A novel ensemble-based wrapper method for feature selection using extreme learning machine and genetic algorithm, Knowledge and Information Systems 57 (2) (2018) 389–412.

[29] E. Sevinc, A novel evolutionary algorithm for data classification problem with extreme learning machines, IEEE Access 7 (2019) 122419–122427.

[30] A. Deniz, H. E. Kiziloz, On initial population generation in feature subset selection, Expert Systems with Applications 137 (2019) 11–21.