# Feature Selection Algorithm Ensembling Based on Meta-Learning

***Abstract*—We propose a new approach to feature selection that is based on ensembling and meta-learning. Meta-learning is used to choose feature selection algorithm that are used to produce feature rankings, which are then aggregated into a resulting feature ranking. This approach requires a lot of additional computational time for meta-learning system construction, but it works fast and shows better feature selection quality than algorithms in aggregation.**

## I. Introduction

In the modern world, machine learning is widely used in almost any field of human activity. Feature subset selection (FSS) is one of frequently used method in data mining and machine learning [1]. It is a preprocessing step which is used to decrease training time and enhance generalization by reducing probability of overfitting via model simplification. A FSS algorithm returns a subset of the original feature set, which is the main and only difference from feature extraction algorithms. FSS algorithms effectively remove irrelevant and redundant features taking into account feature interactions [2].

FSS methods can be divided into five groups depending on how feature selection itself is interacting with model construction: wrappers, filter, embedded, hybrid, and ensembling. Wrappers are FSS methods that apply some search in the feature subset space and use a processing algorithm for estimating the quality of the selected subset. These methods are quite slow and can be applied only to relatively small datasets, as for each new subset new model has to be trained by applying the selected processing algorithm. On the contrast, filters use only intrinsic properties of features and work really fast with high-dimensional datasets. Embedded methods are part of learning models that implement FSS as the result of the model training. Hybrid methods usually combine several different FSS methods resulting in order to obtain higher model efficacy. For example, a wrapper can be applied after a filter, which leads to lower overall time investment into feature selection process. Ensembling methods work with several feature subsets selected by different methods and aggregating them.

Ensembling methods are usually applied only for filters, as filters are quite fast and easily scalable. A proper choice of FSS algorithms to be blended in an ensemble may lead to high increase in model quality and speed. With such variety of feature selection methods, it is hard to select the proper one, and some of them also require careful parameters tuning. Since feature selection is required for the data preprocessing step, it should be performed in a fast and efficient way. The problem of efficient selection of FSS algorithms may be solved under the meta-learning approach [3]. Meta-learning helps to solve the problem of algorithm selection by predicting the best algorithm (or their ranking) for a problem instance never seen before using knowledge on how algorithms works on other problem instances. However, we will use it to predict proper feature selection algorithms to be included into an ensemble.

Meta-learning is a powerful technique to predict the most efficacious algorithm and ensemble learning is a powerful technique to improve algorithms efficacy in general. The algorithm we propose is based on combination of these two techniques in order to obtain a novel, powerful tool for feature selection. The main goal of this paper is to propose such an algorithm that uses meta-learning to choose FSS algorithms to be chosen for ensembling.

The rest of this paper is organized as follows. In Section II, we provide necessary terms and concepts of meta-learning and ensemble-learning. In Section III, we present our ensembling feature selection algorithm. Section IV contains experiments results and their discussion. Section V concludes this work.

## II. Background

### A. Meta-learning Approach

Meta-learning approach is used to handle with the problem following from The No Free Lunch theorems [4]: without any task-specific behavior based on prior information about given dataset, expected efficacy of all the algorithms (including coin toss) is the same. The core idea of meta-learning is to think of algorithm selection problem as a supervised learning problem [5], [6]: datasets are objects; target function maps a dataset to the algorithm, which shows the highest performance for this dataset with respect to a certain criterion. A dataset is described with its properties, which are called meta-features. Thus, the problem of algorithm selection is reduced to learn such a classifier, which can predict the best algorithm for given dataset. After the paper [7], the problem to be solved is usually the learning to rank problem: not only the best algorithm should be predicted, but the ranking of algorithms according to their performance. It allows shifting from "true-false" prediction language to a more well-grained one, which can describe how much we are mistaking in prediction.

Consider the universal space of datasets $\mathscr{D}$. Under the classical approach [8] we assume that we are given an algorithm set $A = \{a_1, \ldots, a_L\}$ and a certain quality criterion $Q : A \times \mathscr{D} \to \mathbb{R}$. We also will think that we already know the set of meta-features which describe datasets: $M = \{m_1, \ldots, m_{|M|}\}$,

$m_i : \mathscr{D} \to \mathrm{Codomain}(m_i)$. Also we are given a set of training datasets $D_{\mathrm{train}} = \{d_1, \ldots, d_{|D_{\mathrm{train}}|}\} \subset \mathscr{D}$. Our goal is to learn an algorithm, which we will refer to as the meta-learning system. This algorithm is required to return an ordered subset of $A$ for given dataset $d$. It is the classification problem in case if only one (the best) algorithm should be returned.

Despite many algorithms can be used on meta-level to solve this problem, such as neural networks [9], decision rules or ranking trees [10], one of the most popular one to be used is well-known kNN algorithm. The intuition that stands behind is the following: if an algorithm performs well on dataset $d$, then it will perform well on dataset $d'$ similar to $d$. It leads us to additional assumption that distance $\rho_M(d, d')$ between the datasets $d$ and $d'$ can be evaluated. The distance depends on meta-feature set $M$. For given dataset $d$ and algorithm $a$ we can predict $Q(a, d)$ if we know $\rho_M(d, d')$ and $Q(a, d')$ for all $d'$ being neighbors of $d$. Therefore, we can predict algorithm rank for given dataset.

The meta-learning system consists of the two basic parts: meta-information storage (database) and computational engine. The database stores meta-features for each training dataset and results of algorithm runs on this dataset. This database is constructed during the training step. The second part of the system serves to search datasets, which are the most similar to given dataset with respect to the chosen distance $\rho_M$. Also it predicts algorithm performance on the given dataset. The recommendation system ranks algorithms with respect to the evaluated similarities to the given datasets.

### B. Meta-learning for Feature Subset Selection Algorithms

Many different FSS algorithms exist. This diversity can be explained by the fact that the processed data can be gathered from completely different areas. Moreover, there is no FFS algorithm that will be the best on all kinds of data [11]. In contrary to classifier selection problem, to which meta-learning approach was applied many times [**?**], only a small number of papers are devoted to application of meta-learning to FSS problem [12], [13].

Suppose we are given a dataset $d$ and we have $L$ FSS algorithms $A = \{a_1, \ldots, a_L\}$. Then let $S_l$ be the list of features obtained by algorithm $a_l$ execution on $d$. Let $S_A = \cup_l \{S_l\}$ denote a set of features selected by algorithms from $A$, and $R_l(t)$ be a function that returns position of element $t$ in list $S_l$, i.e. if $S_l = \left(t_1, \ldots, t_{|S_l|}\right)$, then $R_l(t_j) = j$, and for each $t \notin S_l$ set $R_l(t) = |S_l| + 1$. A performance measure for comparing performance of different feature subset selection algorithms proposed in [12] is called EARR (Extended Adjusted Ratio of Ratios):

$$\mathrm{EARR}(a_i, a_j; d) = \frac{\mathrm{acc}_i / \mathrm{acc}_j}{1 + \alpha \log t_i/t_j + \beta \log n_i/n_j},$$

where $\alpha$ and $\beta$ are user-defined parameters representing relative importance of desired performance time and degree of data compression, and $\mathrm{acc}_i$, $t_i$ and $n_i$ are efficacy measure, performance time and number of selected feature for $i$th FSS algorithm on $d$ respectively. The efficacy measure of FSS algorithm is accuracy of a chosen classifier executed on the dataset preprocessed with this algorithm.

The meta-learning system predicts three FSS algorithms that are expected to be the best. The research results show that this system prediction contains the best algorithm with respect to the chosen criterion correctly in from 90.43% to 96.52% of cases depending on the classifier being used. 115 different well-known datasets were used for testing.

### C. Ensemble Learning

Ensemble learning is the approach, the core idea of which is to combine different algorithms solving the same problem in order to receive a better algorithm solving this problem [14], [15]. Nowadays, ensemble learning has many real-world applications including object detection and tracking, scene segmentation and analysis, image recognition, information retrieval, bioinformatics, data mining, etc. [14]. In the most cases, proper choice of algorithms ensemble improves problem solution quality [16], [17].

Ensemble learning is known to be applicable in FSS problem. As an example, in paper [17], where Tuv et al. suggest and analyze a new efficient feature subset selection algorithm using tree-based ensembles to generate a compact subset of non-redundant features. This algorithm shows high efficacy on different datasets. Filchenkov et. al. suggest approach to learn ensemble of ranking filters by combining their feature importance measures [18]. The approach proposed by Bólon-Canedo et al. [19] can be understood as FSS ensemble learning, despite they learn ensembles not of FSS algorithms, but classifiers which are one classifier model learnt on datasets preprocessed with the FSS algorithms.

In this work, we will follow the approach, presented in paper [20], which is based on aggregating resulting feature ranks, obtaining a new feature rank which is used to select the best ones. We can suggest that each FSS method returns a ranked feature set. Thus, we need to imply rank aggregation algorithms in order to obtain a single ranking.

### D. Rank Aggregation

The rank aggregation problem is to combine several different ranks on the same set of candidates, or alternatives, in order to obtain a "better" ordering. In this section we describe popular ranking aggregation methods that we will use in our research.

*1) Borda Methods:* Consider the following group of heuristic rank aggregation methods, which are called Borda methods [21]. These methods are simple and intuitive, the main idea is to aggregate ranks by calculating a weight of each candidate and using different convolution functions such as arithmetic average, geometric mean, and other to find the resulting weight of each candidate. The resulting weights are used to find position of candidate in resulting rank.

In other words, weight functions computation is based on position of each candidate in each of input ordering:

$$w(t) = w\left(R_1(t), \ldots, R_L(t)\right), \ \forall t \in T.$$

Then all the candidates are ordered by values of $w$ function:

$$R^*(t_i) < R^*(t_j) \Leftrightarrow w(t_i) < w(t_j),\ \forall t_i, t_j \in T.$$

Several Borda algorithms exist defined by different weights functions:

- Method **Borda-ARM**, in which $w$ is the arithmetic average function:

$$w(x_1, \ldots, x_m) = \frac{1}{m} \sum_{i=1}^{m} x_m.$$

- Method **Borda-MED**, in which $w$ is the median function:

$$w(x_1, \ldots, x_m) = \text{median}(x_1, \ldots, x_m).$$

- Method **Borda-GEM**, in which $w$ is the geometric mean function:

$$w(x_1, \ldots, x_m) = \left( \prod_{i=1}^{m} |x_m| \right)^{\frac{1}{m}}.$$

- Method **Borda-L2N**, in which $w$ is the $L_2$-norm function:

$$w(x_1, \ldots, x_m) = \sqrt{\sum_{i=1}^{m} |x_m|^2}.$$

*2) Markov Chain Methods:* Markov chain methods provide more elegant but more complex solutions than Borda methods do. One significant difference between these two method families is that Borda methods use all the ranking ordering, while Markov chain methods use only pairwise ranking.

Several different methods to fill transition matrix exist. This matrix is used to find stationary distribution $P$ describing the Markov chain. In other words, we search a vector $\pi$, such that $\pi \times P = \pi$. The transition probability may be modified to guarantee the existence of a unique stationary distribution in the following way, as suggested by DeConde et al. [22]:

$$P'(u \to v) = (1 - a)P(u \to v) + a/L,$$

where $a$ is a tuning parameter usually set to be in interval $[0.001, \ldots, 0.15]$. In this work, we set parameter $a$ a to be equal to 0.05.

Several ways to construct Markov chain exist, in this work we use the following methods [23], [22]:

- **Method MC1**: for all $u, v \in S, u \neq v$ define transition probability as the following:

$$P(u \to v) = \begin{cases} 1/|S|, & \text{if } \exists l : R_l(u) > R_l(v); \\ 0, & \text{otherwise.} \end{cases}$$

  Then define diagonal elements as the following:

$$P(u \to v) = 1 - \sum v \neq uP(u \to v).$$

- **Method MC2**: for all $u, v \in S, u \neq v$ define transition probability as the following:

$$P(u \to v) = \begin{cases} 1/|S|, & \text{if } R_l(u) > R_l(v) \text{ for most} \\ & \text{of the input lists;} \\ 0, & \text{otherwise.} \end{cases}$$

  Then define diagonal elements the same as in MC1.

- **Method MC3**: for all $v, u \in S, u \neq v$ define transition probability as the following:

$$P(u \to v) = \sum_{i=1}^{L} I(R_i(u) > R_i(v))/L|T|.$$

  where $I$ is the indicator function that is equal to 1 if the condition inside the parentheses is satisfied; otherwise it is equal to zero.

Then we calculate stationary distribution $\pi$ of the Markov chain defined by $P$. This vector $\pi$ will be directly used for ranking FSS algorithms: then greater the value then higher the algorithm in rank, i.e. algorithm with index $\arg\max_i \pi_i$ will be on the top and algorithm with index $\arg\min_i \pi_i$ will be on the bottom of rank.

## III. BFSSAEL Algorithm

### A. Core Idea

A meta-learning system usually returns not only the algorithm predicted to be the best one, but the top $L$ algorithms, from which user should choose the best one (or a wrapping method should try each of them). Quality of such a system is usually estimated with respect to how good is the best of $L$ returned algorithms.

The main idea of this work is to apply ensemble technique for all the best recommended algorithms. As we expect, this will built efficacious algorithm not for a certain dataset, but for its locus (e.g. "locally general").

In order to combine algorithms properly, we should adjust aggregated feature list size. This problem arises because selected ranks do not always have the same length and can consist of different sets of the features. It leads to problem of adjusting length of the resulting aggregate rank.

In this paper, we use three following subsets of feature list $S^*$, aggregated of lists

$$S^*_{\min} = \{t \in S^* | R^*(t) < le \min_i |S_i|\};$$

$$S^*_{\text{average}} = \{t \in S^* | R^*(t) < le \text{average}_i |S_i|\};$$

$$S^*_{\max} = \{t \in S^* | R^*(t) < le \max_i |S_i|\}.$$

For determining the best of them, we evaluate the classifier performance on each of these feature lists and compare them with respect a performance measure. This allows us to assume that all the aggregation methods return already adjusted list of features.

### B. AEARR Metric

In order to compare different FSS algorithms, we need to modify EARR metric. We use the meta-learning system as a black box, therefore its parameter $\alpha$ influences only on list of the feature subset selection algorithms. This list will be used as an input for the proposed system, therefore we may assume that parameter $\alpha$ is user-defined and is a part of that black box.

It worth to note that EARR metric uses classification accuracy as the FSS algorithm efficacy measure. Accuracy is

known to be inefficient in cases, in which classes have notable difference in sizes. In this paper, we suggest to use $F_1$-measure instead.

We introduce a new metric for comparing algorithms on dataset $d$, which we call AEARR (Aggregated Extended Adjusted Ratio of Ratios). It is defined as follows:

$$\mathrm{AEARR}(a_i, a_j; d) = \frac{F_i/F_j}{1 + \beta \log n_i/n_j},$$

where $f_i$ is value of $F_1$ of a classifier executed on dataset $d$ preprocessed with algorithm $a_i$, and $n_i$ is the number of features this algorithm selects during preprocessing.

To obtain the absolute rating, we order algorithms with respect to the following measure:

$$\mathrm{AEARR}(a_i; d) = \frac{1}{L-1} \sum_{j, j \neq i} \mathrm{AEARR}(a_i, a_j; d).$$

### C. Algorithm

*1) Algorithm Scheme:* Assume we have already learnt the meta-learning system for FSS algorithm recommendation. The system performs the following steps for a given dataset $d$:

1) Run the meta-learning system and find the top $k$ algorithms for $D$.
2) Extract $k$ feature rank lists from dataset using $k$ algorithms from the previous step.
3) Run each of $g$ rank aggregation methods on the $k$ rank lists obtained in the previous step.
4) Evaluate classifier performance and calculate AEARR metric for each of $g$ feature lists.
5) Return feature set with the largest value of AEARR metric as the answer.

Algorithm scheme is shown in Fig. 1.

*2) Algorithm parameters:* The algorithm has several user-defined parameters for adjusting the system to solve a particular problem:

- classifier is used for feature selection algorithm performance evaluation;
- parameter $k$ is the number of top algorithms, returned by the recommendation system;
- parameter $\alpha$ is sensitivity to feature subset selection time;
- parameter $\beta$ is sensitivity to number of selected features.

It is worth to note that all the parameters have significant impact on the system. Besides, parameters $k$ and $\alpha$ make huge effect on evaluation time, while parameter $\beta$ shows significant influence only on number of features.

*3) Usage of Weights for Increasing Classification Quality:* All the rank aggregation methods mentioned above may use weights for increasing quality. In this section, we describe two ways to implement it. The first way is based on the fact that the meta-learning system ranks the algorithms assigning weights to them. We can use these weights in rank aggregation. The other way is to evaluate classifiers on each of the recommended ranks, and then use AEARR metric as weights for ranks aggregation.

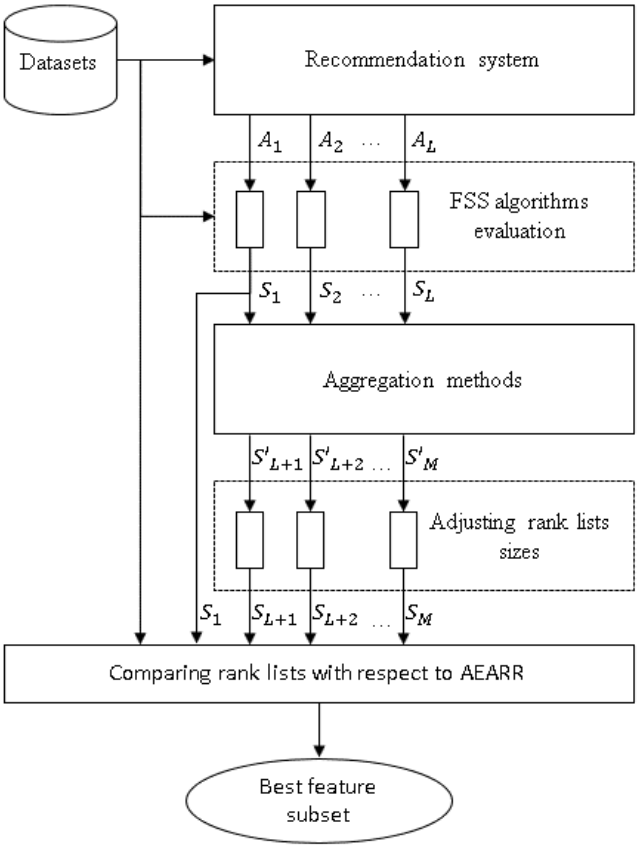


Fig. 1. Algorithm architecture with detailed evaluation engine.

## IV. Experiments

In this section, we describe results of algorithm executions on different datasets. The system is implemented on Python and Java, using WEKA [24] machine learning library.

We must note that the parameter $\alpha$ corresponding to sensitivity of FSS time is used only in the meta-learning system, therefore in the experiments we set $\alpha$ equal to 0. We will not affect the fullness of research by doing that, because this parameter has influence only on recommended algorithms and the system compares features relatively with respect to value of AEARR metric.

### A. Experiment setup

75 well-known datasets from different domains are taking for system testing. Bayesian Network is used as a classifier to measure feature selection algorithm efficacy. We used 5-fold cross-validation to evaluate model performance.

### B. Experiment results

In this subsection, numerical results of the experiments are presented. The values are the average for all the datasets.

*1) Result with different values of $\beta$:* Plot in Fig. 2 shows that the proposed algorithm is more efficient in average than the recommended algorithms for all values of parameter $\beta$.

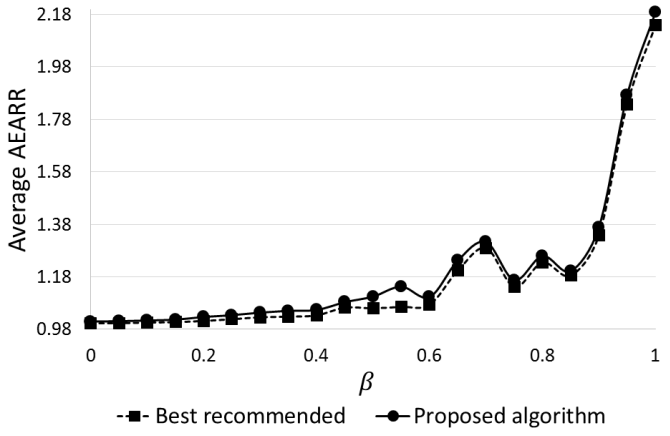


Fig. 2. Result depends on parameter $\beta$ with $L = 4$.

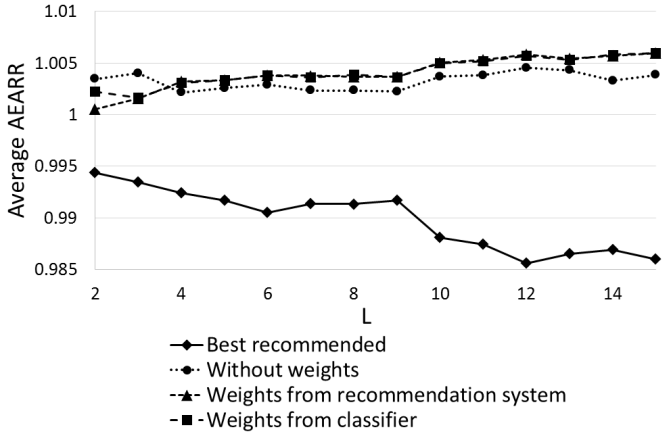


Fig. 3. Result with respect to different value of parameter $L$ with $\beta = 4$.

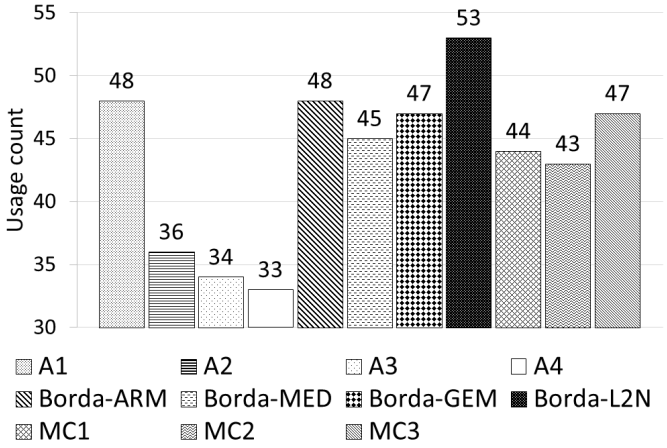


Fig. 4. Number of times when algorithm was the recommended or works equally with the recommended one.

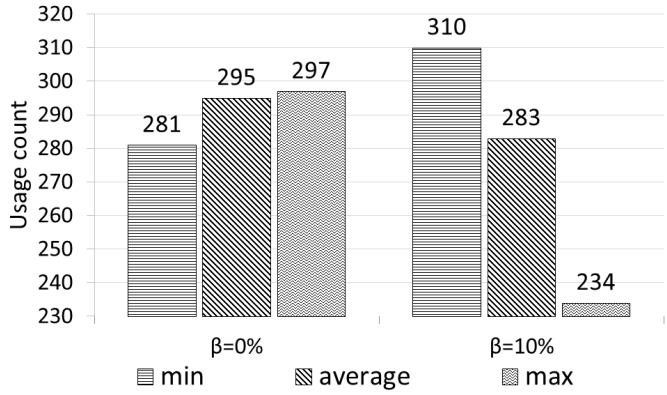


Fig. 5. Number of usage in dependence on number of features and parameter.

More detailed information about this run is given in attachment [25].

*2) Results with different values of L:* Plot in Fig. 3 shows that the proposed algorithm is more efficient than the best of two algorithms recommended by the system. In addition, it can be noted that it is reasonable to use weights from the meta-learning system, because in this case system shows better results. However, there is only a slight difference between the two approaches used to assign weights with $L \geq 2$.

More detailed information about this run is given in attachment [26].

If parameter $L$ equals to 4, then the algorithm works efficiently enough. If parameter $L$ is greater than 4, then no big boost in efficacy of the algorithm can be seen. Therefore, the number of algorithms to use in ensemble is four, because it allows finding a compromise between efficient and computing time.

The experiments allow to find some patterns. Generally, the Borda method that uses $L_2$-norm shows the best results. In addition, Borda methods that use arithmetic average and geometric mean and method MC3 show high results.

### C. Discussion

As could be seen from results, feature lists that was created with rank aggregation methods show better results than received from base algorithms, that was selected by recommendation system with different parameter $\beta$.

Increasing the number of used algorithms increases algorithm efficiently. It was founded experimentally that four algorithms in ensemble is optimal. This choice caused by necessity to find balance between quality and computation complexity.

Using weights in this implementation of algorithms shows small increment of quality.

Computational time directly depends on features subset selection algorithms time, and also depends on number of such algorithms $L$.

## V. Conclusion

In this paper, we presented new approach to ensemble feature selection. We used meta-learning in order to select the best possible filtering feature selection methods for some dataset and then used their aggregation for final feature selection in regards to AEARR metric. Different experiments shown

that this approach leads to better feature selection results in comparison to aggregated methods.

To improve results of this paper some more feature selection algorithms properties could be used. For example, An interesting idea is employing information about algorithm similarity. Its application to classifiers ensemble is out of question, due to classifiers diversity is the primary source for their ensemble efficacy. Application of this idea to feature selection domain is more questionable, and one of the further work direction is learning feature selection algorithm ensemble using their similarity [27].

## ACKNOWLEDGMENT

## REFERENCES


[1] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[2] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[3] A. Zabashta, I. Smetannikov, and A. Filchenkov, "Rank aggregation algorithm selection meets feature selection," *Imperial Journal of Interdisciplinary Research*, vol. 9729, pp. 740–755, 2016.

[4] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 67–82, 1997.

[5] P. Brazdil, C. G. Carrier, C. Soares, and R. Vilalta, *Metalearning: applications to data mining*. Springer Science & Business Media, 2008.

[6] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002.

[7] P. B. Brazdil, C. Soares, and J. P. Da Costa, "Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results," *Machine Learning*, vol. 50, no. 3, pp. 251–277, 2003.

[8] L. A. Rendell, R. Sheshu, and D. K. Tcheng, "Layered concept-learning and dynamically variable bias management." in *IJCAI*, 1987, pp. 308–314.

[9] H. Bensusan, C. G. Giraud-Carrier, and C. J. Kennedy, "A higher-order approach to meta-learning." *ILP Work-in-progress reports*, vol. 35, 2000.

[10] Q. Sun and B. Pfahringer, "Pairwise meta-rules for better meta-learning-based algorithm ranking," *Machine learning*, vol. 93, no. 1, pp. 141–161, 2013.

[11] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowledge and information systems*, vol. 34, no. 3, pp. 483–519, 2013.

[12] G. Wang, Q. Song, H. Sun, X. Zhang, B. Xu, and Y. Zhou, "A feature subset selection algorithm automatic recommendation method," *Journal of Artificial Intelligence Research*, 2013.

[13] A. Filchenkov and A. Pendryak, "Datasets meta-feature description for recommending feature selection algorithm," in *AINL-ISMW FRUCT*, 2015, pp. 11–18.

[14] R. Polikar, C. Zhang, and Y. Ma, "Ensemble machine learning: Methods and applications," 2012.

[15] T. G. Dietterich, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.

[16] O. DeMasi, J. Meza, and D. H. Bailey, "Dimension reduction using rule ensemble machine learning methods: A numerical study of three ensemble methods," *arXiv preprint arXiv:1108.6094*, 2011.

[17] E. Tuv, A. Borisov, G. Runger, and K. Torkkola, "Feature selection with ensembles, artificial variables, and redundancy elimination," *The Journal of Machine Learning Research*, vol. 10, pp. 1341–1366, 2009.

[18] A. Filchenkov, V. Dolganov, and I. Smetannikov, "Pca-based algorithm for constructing ensembles of feature ranking filters," in *ESANN*, 2015, pp. 202–206.

[19] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "An ensemble of filters and classifiers for microarray data classification," *Pattern Recognition*, vol. 45, no. 1, pp. 531–539, 2012.

[20] L.-Y. Chuang, C.-H. Yang, K.-C. Wu, and C.-H. Yang, "A hybrid feature selection method for dna microarray data," *Computers in biology and medicine*, vol. 41, no. 4, pp. 228–237, 2011.

[21] S. Lin, "Rank aggregation methods," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 5, pp. 555–570, 2010.

[22] R. P. DeConde, S. Hawley, S. Falcon, N. Clegg, B. Knudsen, and R. Etzioni, "Combining results of microarray experiments: a rank aggregation approach," *Statistical Applications in Genetics and Molecular Biology*, vol. 5, no. 1, 2006.

[23] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 613–622.

[24] M. L. G. at the University of Waikato, "Weka 3: Data Mining Software in Java," http://www.cs.waikato.ac.nz/ml/weka/, 2017, [Online; accessed 16-May-2017].

[25] anonymised, "Data table 1," https://www.dropbox.com/s/xzvda5xu4791nql/Attachment%201.pdf?dl=0, 2017.

[26] "Data table 2," https://www.dropbox.com/s/z3v2n1p0o2qc47z/Attachment%202.pdf?dl=0, 2017.

[27] N. Dessì and B. Pes, "Similarity of feature selection methods: An empirical study across data intensive classification tasks," *Expert Systems with Applications*, vol. 42, no. 10, pp. 4632–4642, 2015.