Datasets Meta-Feature Description for Recommending Feature Selection Algorithm

Andrey Filchenkov, Arseniy Pendryak
ITMO University
St. Petersburg, Russia
afilchenkov@niuitmo.ru, pendryak@rain.ifmo.ru

***Abstract*—Meta-learning is an approach for solving the algorithm selection problem, which is how to choose the best algorithm for a certain task. This task corresponds to a dataset in machine learning and data mining. The main challenge in meta-learning is to engineer a meta-feature description for datasets. In the paper we apply meta-learning for feature selection. We found a meta-feature set which showed the best result in predicting proper feature selection algorithms. We also suggested a novel approach to engineer meta-features for data preprocessing algorithms, which is based on estimating the best parametrization of processing algorithms on small subsamples.**

I. Introduction

Curse of dimensionality [1] arises in many fields of artificial intelligence, especially in data mining. One of the commonly used approaches to handle this problem is application of dimensionality reduction techniques. Decreasing the number of variables describing data leads to increasing the speed and sometimes the efficacy of applied algorithms. Dimensionality reduction techniques are divided into [2] feature (or variable) selection and feature extraction. Feature selection algorithms are constrained to return only subsets of the original feature set, while feature extraction algorithms can return any feature set, which causes high time consumption in comparison to feature selection. Another advantage of feature selection is the explicit satisfaction of the interpretability request: in many practical domains it is required to estimate the influence of each variable on the answer and algorithm performance. For instance, this is essential in bioinformatics and computational linguistics. In this paper we will focus only on feature selection algorithms.

Selection of the proper feature selection algorithm is considered to be complicated [3] because it is usually hard to compare different algorithms and because too many feature selection algorithms are available [4], [5]. This problem is not specific to the algorithm selection domain, but arises in almost every field of artificial intelligence and computer science. Rice is known to be the first to provide a theoretical framework for algorithm selection problem [6]. Wolpert and Macready's well-known No Free Lunch Theorems as well as related results [7], [8], [9], [10] prove the impossibility to create the universal algorithm for a certain problem, in particular, feature selection problem. Nowadays, algorithm selection is an expert problem for most tasks.

Meta-learning is an approach providing a framework for reasonable selection of an algorithm (and sometimes its parametrization) for a certain task from given set of algorithms [11]. The main idea of meta-learning is reducing the algorithm selection problem to a machine learning supervised problem: tasks are described with special meta-features representing their properties (for example, is dataset sparse or high-dimensional). It can be understood as a formalization of heuristics which are applied or could be applied by a data analyst in order to select the most appropriate algorithm. Meta-learning can be considered as a formalized case of transfer learning, because it selects an algorithm grounded on the knowledge of previously estimated algorithms behavior. This is implemented with supervised learning paradigm: all tasks are described with meta-features, algorithm performance is evaluated on tasks from a training set, and then a prediction of the best algorithm for a new dataset is made. Meta-learning is applied in many artificial intelligence domains, such as evolutionary algorithms [12], [13], constraint satisfaction problems [14], [15], discrete problems [16], [17], and mostly to machine learning problems. The overwhelming majority of papers on meta-learning were dedicated to classifier selection. This can be simply explained by the fact that classification is the most prevailing problem in data mining. Also there are two main restrictions which prevent generalization of meta-learning algorithms and systems to other problems: the issue of quality measure selection and problem of creating a meta-feature description.

The described disproportion can be illustrated this way: there are more than 100 articles on classification algorithm selection, but rarely more than 10 articles on clustering [18], [19], regression [20], [21] or forecasting [22], [23]. The only work on feature selection algorithm recommendation (we will use word recommendation instead of selection to avoid multiple usage of the latter word) is the paper by Wang et al [24]. In this paper authors used only 13 meta-features taken from classification algorithm selection problems. They provide neither exploration, nor explanation of why they used this meta-feature space. Therefore, we can state that the problem of meta-feature description for feature selection algorithm recommendation has, to the best of our knowledge, never been considered.

The goal of this work is to find out which meta-features can be used for feature selection algorithms recommendation. The contribution of this paper is two-fold. First, we introduce a new approach for meta-features engineering, which is shown to be useful for feature selection algorithm recommendation. Second, we conduct an almost complete analysis of popular meta-features and suggest the best optimal meta-feature sets for different cases.

The remainder of this paper is organized as follows. In Section II we describe the principles of meta-learning

ISBN 978-952-68397-0-7 (paperback), ISBN 978-952-68397-1-4 (PDF)

systems both in the general case and for feature selection algorithms. Section III contains description of existing and novel meta-features. In Section IV the described meta-features are analyzed and optimal subsets are selected. These results are discussed in Section V, and Section VI summarizes and concludes the paper.

II. Meta-learning system for feature selection algorithm recommendation

Modern meta-learning systems implement the very similar architecture to METAL [25] (with the exception of more complicated systems such as multi-agent one, described in [26]). This architecture is task-independent, therefore it is applicable not only for classification task. The feature subset selection algorithm recommendation system [24] also shares this architecture.The system consists of two main parts that implement different steps of an offline machine learning algorithm: learning and application. These parts are learning and recommendation correspondingly.

Let $\mathscr{D}$ denote the universal set of datasets, $D = \{d_1, \dots, d_{|D|}\} \subset \mathscr{D}$ denote training datasets, $A = \{a_1, \dots, a_{|A|}\}$ denote the algorithm set, $F = \{f_1, \dots, f_{|F|}\}$ denote the meta-features, which are functions defined on $\mathscr{D}$, $f_i : \mathscr{D} \to \text{Codomain}(f_i)$, and let $F(d) = \{f_1(d), \dots, f_{|F|}(d)\}$ denote the meta-feature description of dataset d.

A. Learning part

The most popular algorithm on meta-level is kNN. It is based on the assumption that each algorithm performs in a similar way on similar datasets. This well-known algorithm simply finds k datasets in D that are the nearest to a new dataset d_{new}. kNN is a distant-based algorithm, therefore a distance function is required to be defined on $\mathscr{D}$. In this work we will use the popular L_1 metric, which defines the distance between two datasets as follows:

$$\text{dist}(d_i, d_j) = \|F(d_i) - F(d_j)\|_1 = \sum_{t=1}^{|F|} |f_t(d_i) - f_t(d_j)|.$$

The very common step in data analysis is feature normalization. In particular, it is the always applied for meta-features. We apply the following normalization:

$$f^{norm} = \frac{f - \min(f)}{\max(f) - \min(f)},$$

where $\min = \min\limits_{d \in D}(f(d))$ and $\max = \max\limits_{d \in D}(f(d))$ are the minimum and maximum values of meta-feature f value. After this normalization all the meta-features values are in the interval $[0, 1]$.

kNN is a lazy algorithm, therefore no explicit learning step is performed. The system only calculates all values in features–object matrix (which is meta-features–datasets in this case) $\mathbf{P}$. Each cell $P_{i,j}$ of this matrix contains a performance evaluation of algorithm a_i on dataset d_j with respect to a certain performance measure P :

$$\mathbf{P} = (P(A_i, D_i))_{i=1,j=1}^{i=|A|,j=|D|}.$$

Algorithm performance measures are defined by the type of task. We discuss the performance measure P for feature selection in subsection II-C.

B. Recommender part

The recommender part of the system is the one that is used for processing a new dataset d_{new} and return list of algorithm recommendations. METAL was the first meta-learning system whose output was not a single algorithm, but rather a ranked list of algorithms.

Let $kN(d) = (d_{(1)}, \dots, d_{(k)})$ denote the k nearest datasets to dataset d. Expected performance measure of algorithm A_i on d_{new} is estimated as

$$P_{pred}(a_i, d_{new}) = \sum_{d_j \in kN(d_{new})} \omega_j \cdot P(a_i, d_j),$$

where $\omega_j = \frac{d_j^{-1}}{\sum_{t=1}^{k} d_t^{-1}}, d_j = \text{dist}(d_{new}, d_j)$.

Algorithms from A are sorted with respect to P_{pred}. This sorted list of algorithms is the recommendation. The system returns the top N algorithms, where N usually equals three. If $N = 1$, the algorithm is a classifier, which returns a single algorithm expected to be the best as the answer.

C. Performance measure

A feature selection algorithm performance can be estimated with respect to the three measures: runtime, number of selected features, and efficacy of a classifier C executed on selected features. The classifier efficacy reveals importance of the selected features for the further processing; the runtime shows how fast is the feature selection algorithm; the number of selected features represents the degree of data compression.

Let C be a classifier. Let t_i^j denote the runtime of algorithm a_i on dataset d_j, n_i^j denote the number of features in dataset $f_i(d_j)$ which is the result of preprocessing dataset d_j with algorithm a_i, and acc_i^j denote the accuracy of C applied to dataset d_j preprocessed with algorithm a_i : $acc_i^j = \text{Accuracy}(C(a_i(d_j)))$. Then $EARR$ (extended adjusted ratio of ratios) of a_i and a_j on d_k is defined [24] as

$$EARR_{a_i,a_j}^{d_k} = \frac{acc_i^k / acc_j^k}{1 + \alpha \cdot \log(t_i^k / t_j^k) + \beta \cdot \log(n_i^k / n_j^k)},$$

where α and β are user-defined parameters representing degrees of importance of the runtime and the number of features. With a certain α and β assignment, two algorithms can be compared with respect to $EARR$: if $EARR_{a_i,a_j}^{d_k} > EARR_{a_j,a_i}^{d_k}$ then a_i performs better than a_j on dataset d_k. Now $EARR$ is defined for pairwise comparison and it is simply generalized:

$$EARR_{a_i}^{d_k} = \frac{1}{|A| - 1} \sum_{j=1, j \neq i}^{|A|} EARR_{a_i,a_j}^{d_k}.$$

Now $EARR$ shows how good an algorithm is on a certain dataset.

$EARR$ has two disadvantages.

The first disadvantage is the use of Accuracy: we use F-measure instead which is more universal, because it can deal with unbalanced classes [27]. We define $EFRR$ (extended with F-measure adjusted ratio of ratios):

$$EFRR^{d_k}_{a_i,a_j} = \frac{f_i^k / f_j^k}{1 + \alpha \cdot \log(t_i^k / t_j^k) + \beta \cdot \log(n_i^k / n_j^k)},$$

where f_i^j denotes the F_1-measure of C applied to dataset d_j preprocessed with algorithm a_i.

The second disadvantage is that both Accuracy and F-measure are absolute, but not comparative measures, therefore algorithms which show slightly worse performance than the best one will cause the same error as an algorithm showing really poor performance. Let $EFRR_{rec}$ denote the performance measure of the recommended algorithm and $EFRR_{opt}$ denote performance of the best algorithm on dataset d. Then we measure the performance quality as the ratio between two values:

$$RPR(a_{rec}, d) = \frac{EFRR_{rec}}{EFRR_{opt}}.$$

Consider the following example. Suppose the system recommends an algorithm with $EFRR_{rec}$ equals 0.70. If $EFRR_{opt}$ equals 0.71, then RPR is equal 0.986 and the classifier performs comparatively well, it is slightly worse than the best one. But if $EFRR_{opt}$ equals 0.95, then RPR is equal to 0.737 and that means that the system returned a not very effective algorithm. If $RPR = 1$, then the system recommended the best algorithm. In both the considered cases value of $EFRR_{rec}$ was the same. This shows that RPR is better in these terms than $EFRR$.

III. META-FEATURES SET

Since most meta-learning systems are dedicated to the classification task, most of the proposed meta-features describe datasets with known class labels. In this research we are focusing on feature selection for classification, therefore the datasets are the same as in classification task. The first question must be asked is if the meta-features for classification algorithm prediction can be used for feature selection. Feature selection is a preprocessing step for applying supervised or unsupervised algorithms. That is why we expect the meta-features used for classification will suit for feature selection for classification.

A. *Standard meta-features*

Basic meta-features are divided into [28], [29]:

- general: the number of samples in dataset, the number of features, the number of classes, etc.
- statistical: standard deviation, correlation coefficient, asymmetry coefficient, etc.
- information-theoretic: mean feature entropy, mutual information of class and attribute, noise ratio, etc.

Another group is landmark features which are performance measures of a classifier trained on a small subsample of the dataset [30].

Yet another group of meta-features is proposed in [31]. These meta-features are based on decision tree structure, which is learnt on a dataset. In [31] the following characteristics are considered: nodes, leaves, branches, tree levels. These properties are supposed to describe the tree structure well, therefore they are expected to describe the dataset well. The following meta-features are engineered: tree width and tree height, the number of leaves and number of nodes, maximum, minimum and mean values of branch length, the number of nodes on each level, number of inner nodes corresponding to a certain single-feature rule, and variance of all these values.

B. *Model-based meta-features*

Decision tree based meta-features are grounded upon the assumption that decision tree structure is a model of data [32], therefore the features describing this model are expected to describe the data as well. In this paper we suggest an intensively generalized approach grounded on the same suggestion: each classifier synthesizes a model of data, therefore every model description can be used for data description. For decision trees this model is represented with a tree, for Bayesian networks — with an acyclic graph with conditional distributions, for SVMs — with set of hyperplanes, etc. We will use as meta-features the numerical parameters with which these models can be described.

We must also note that all these algorithm models are parametric. This means that there exists the best parametrization for a certain task. For instance, the best number of nearest neighbors is a numerical value, which also characterizes the data. This is the second type of meta-features we suggest in this paper.

C. *Meta-feature set*

The meta-feature set consists of 79 meta-features: standard meta-features (including meta-features in [24]) and model-based meta-features. The last group contains meta-features built with decision tree, kNN, and perceptron. These three classifiers are based on different paradigms: ensemble of logical classifiers, distance-based classifier and linear classifier correspondently.

As the decision tree we use C4.5, both with and without pruning. Therefore, for each decision tree based meta-features we evaluated both pruned and unpruned instances.

The meta-feature set contains the following meta-features:

1) general:

- number of instances (NumberOfInstances);
- number of features (NumberOfFeatures);
- number of classes (NumberOfClasses);
- dataset dimensionality (DataSetDimensionality).

2) statistical (all values are mean by all non-class attributes):

- standard deviation (MeanStandardDeviation);
- variation coefficient (MeanCoefficientOfVariation);
- correlation coefficient (MeanLinearCorrelationCoefficient);
- skewness (MeanSkewness);

- kurtosis (MeanKurtosis).

3) information-theoretic:

- average normalized features entropy (MeanNormalizedFeatureEntropy);
- normalized class attribute entropy (NormalizedClassEntropy);
- maximal mutual information between the attribute and the class (MaxMutualInformation);
- average mutual information between the attribute and the class (MeanMutualInformation);
- ratio of signal noise (NoiseSignalRatio);
- number of equivalent features (EquivalentNumberOfFeatures).

4) decision tree based:

- tree height (TreeHeight);
- tree width (TreeWidth);
- number of inner vertices (TreeNodeNumber);
- number of leaves (TreeLeavesNumber);
- minimum branch length (TreeMinBranch);
- maximum branch length (TreeMaxBranch);
- average branch length (TreeMeanBranch);
- standard deviation of the tree branch length (TreeDevBranch);
- maximum number of vertices on the same tree level (TreeMaxLevel);
- average number of vertices on the same tree level (TreeMeanLevel);
- standard deviation of the vertices on the same tree level (TreeDevLevel);
- minimum number of inner features that are matching to the same attribute (TreeMinAttr);
- maximum number of inner features that are matching to the same attribute (TreeMaxAttr);
- average number of inner features that are matching to the same attribute (TreeMeanAttr);
- standard deviation of the number of inner features that are matching to the same attribute (TreeDevAttr);
- minimum number of leaves that are matching to the same class (TreeMinClass);
- maximum number of leaves that are matching to the same class (TreeMaxClass);
- average number of leaves that are matching to the same class (TreeMeanClass);
- standard deviation of the number of leaves that are matching to the same class (TreeDevClass).

5) based on the perceptron structure (sum of weights is the sum of all weights of the perceptron; number of objects in the set):

- weight sum on the full dataset (FullPerceptronWeightSum);
- minimum weight sum; subsamples sizes are $N/10$, $N/2$, and $\sqrt{N}$ (MinOneTenthPerceptronWeightSum, MinHalfPerceptronWeightSum, MinSqrtPerceptronWeightSum);
- average weights sum; subsamples sizes are $N/10$, $N/2$, and $\sqrt{N}$ (MeanOneTenthPerceptronWeightSum, MeanHalfPerceptronWeightSum, MeanSqrtPerceptronWeightSum);
- standard deviation of the weights sum; subsamples sizes are $N/10$, $N/2$, and $\sqrt{N}$ (StdDevOneTenthPerceptronWeightSum, StdDevHalfPerceptronWeightSum, StdDevSqrtPerceptronWeightSum).

6) kNN best parameter based:

- number of neighbors for the full dataset (FullBestK);
- minimum number of neighbors; subsamples sizes are $N/10$, $N/2$, and $\sqrt{N}$ (MinOneTenthBestK, MinHalfBestK, MinSqrtBestK);
- maximum number of neighbors; subsamples sizes are $N/10$, $N/2$, and $\sqrt{N}$ (MaxOneTenthBestK, MaxHalfBestK, MaxSqrtBestK);
- average number of neighbors; subsamples sizes are $N/10$, $N/2$, and $\sqrt{N}$ (MeanOneTenthBestK, MeanHalfBestK, MeanSqrtBestK);
- standard deviation of the number of neighbors; subsamples sizes are $N/10$, $N/2$, and $\sqrt{N}$ (StdDevOneTenthBestK, StdDevHalfBestK, StdDevSqrtBestK).

IV. Analysis and selection of meta-features

A. Experimental setup for meta-feature sets comparison

The training set D contains 84 datasets, which can be found in the repository http://genome.ifmo.ru/files/papers_files/FRUCT2015/Datasets.pdf. These datasets have very different nature, and the only fact which is essential is that these datasets describe real-world problems. Despite such a low interest in data nature can cause disagreement for many data science specialist, the core of meta-learning is choosing algorithm without any under assumption. We assume that meta-feature description is the only information which is required to predict algorithms for its processing. This makes meta-learning very useful in Big Data processing, because its well-known property is lack of any intuition on how data should be preprocessed and processed.

Program implementation was performed in Java language, machine learning algorithms were taken from WEKA library [33].

The algorithm set A contains 16 different feature selection algorithms which are listed in Table I. We provide a brief description of these algorithms taken from WEKA. In this library every feature selection algorithm can be represented as a pair (Evaluator, Search Method). Search method is a method to be used in feature subset space search, and each object of this space is evaluated with Evaluator. One special case of algorithms is ranking filters (Ranker search method, do not get confused with Rank search search method), where we do not evaluate subsets, but only single features. This case is supported in WEKA with attribute evaluation methods; in this work we used the probabilistic significance measure (16th algorithm). All the other feature selection algorithms in this paper are evaluated with different implementation of two evaluation measures which are dependency (for the detailed description see [34]) and consistency (for the detailed description see [35]). We use various search methods, which description can be found in [36]. The most detailed overview of WEKA algorithm is given by Witten and Frank in chapter 8 in [37]. We must note that the set of listed feature selection algorithms is the same as in [24].

TABLE I. LIST OF FEATURE SELECTION ALGORITHMS

No	Evaluator	Search method	Evaluation measure
1	CFS-SFS	BestFirst + Seq. Forward Search	Dependency
2	CFS-SBS	BestFirst + Seq. Backward Search	Dependency
3	CFS-BiS	BestFirst + Bi-direction Search	Dependency
4	CFS-GS	Genetic Search	Dependency
5	CFS-LS	Linear Search	Dependency
6	CFS-RS	Rank Search	Dependency
7	CFS-SS	Scatter Search	Dependency
8	CFS-SWS	Greedy Stepwise Search	Dependency
9	CFS-TS	Tabu Search	Dependency
10	Cons-SFS	BestFirst + Seq. Forward Search	Consistency
11	Cons-BiS	BestFirst + Bi-direction Search	Consistency
12	Cons-GS	Genetic Search	Consistency
13	Cons-LS	Linear Search	Consistency
14	Cons-RS	Rank Search	Consistency
15	Cons-SWS	Greedy Stepwise Search	Consistency
16	Signific	Ranker	Prob. Significance

Since the RPR measure defined in Section II depends on a classifier, we used five different standard classifiers in order to avoid classifier-based bias. These classifiers, implemented in WEKA, are: naïve Bayesian classifier (NaiveBayes), two decision trees (C4.5 and PART), Bayesian network (BayesNet), and Nearest Neigbour Classifier (IB3). We use leave-one-out cross-validation to estimate performance of the meta-learning system.

B. *Comparison of meta-features groups*

First, we picked several groups of meta-features:

- ”original” is the set of 13 meta-features, described in [24];
- ”basic” is the set of 4 general meta-features;
- ”statistical” is the set of 5 statistical meta-features;
- ”information-theoretic” is the set of 6 information-theoretic meta-features;
- ”standard” is the set of 15 meta-features which are union of the three previous sets;
- ”pruned” is the set of 19 meta-features, evaluated for pruned decision trees;
- ”unpruned” is the set of 19 meta-features, evaluated for unpruned decision trees;
- ”tree” is the set of 38 meta-features, evaluated for decision trees (union of the previous two sets);
- ”neural” is the set of 13 meta-features, evaluated for perceptron;
- ”knn” is the set of 13 meta-features, evaluate for kNN algorithm;
- ”classifier based” is the set of 64 meta-features, evaluated for classifiers;
- ”all” is the set of all the 79 meta-features described above.

The recommender system efficacy for each meta-feature subset is shown in Table II. We can see from the table that the efficacy of the recommender system for different classification algorithms is maximal on different meta-feature set. This is the evidence of that the original meta-feature set is not the optimal one, therefore the proper choice of meta-feature subset will improve the efficacy.

TABLE II. MEAN VALUE OF RPR FOR EACH CLASSIFIER AND META-FEATURE SUBSETS

Meta-feature set	**NaiveBayes**	**C4.5**	**PART**	**BayesNet**	**IB3**
original	**0.9738**	0.9653	0.9635	**0.9714**	0.9530
general	0.9636	0.9675	0.9589	0.9650	0.9530
statistical	0.9391	0.9571	0.9532	0.9533	0.9448
information-theoretic	0.9691	0.9616	0.9529	0.9589	0.9520
standard	0.9732	0.9643	0.9634	0.9711	0.9534
pruned	0.9502	0.9569	0.9563	0.9655	0.9209
unpruned	0.9639	0.9537	0.9528	0.9576	0.9403
tree	0.9481	0.9578	0.9522	0.9645	0.9505
neural	0.9549	0.9519	0.9477	0.9631	0.9513
knn	0.9571	0.9540	0.9187	0.9515	0.9240
Classifier based	0.9575	0.9648	0.9574	0.9637	**0.9620**
All	0.9601	**0.9725**	**0.9653**	0.9642	0.9516

C. *Meta-feature subset search*

Experimental results described in Subsection IV-B show that even a small change of meta-feature subset can cause a huge change in meta-learning system efficacy. That is why we should be more accurate when choosing a meta-feature description.

Meta-features are features, therefore we apply a feature subset selection algorithm for solving this problem. The performance measure described in Section II is defined with a classifier C performance measure. That is why the only type of feature selection algorithm we can apply for solving this problem is wrappers. We chose the wrapper based on Genetic Search, because classifier performance function is not analytical.

D. *Meta-feature subset optimization for a certain classifier*

In meta-feature subset optimization for a certain classifier (MFSO-single) we fix classifier C and then find meta-feature subset F such that $Q(C(F))$ is maximal, where Q is a quality measure of classifier performance.

This approach is designed to archive high score for a certain classifier and is not so efficient if another classifier is used. This is explicitly shown in Table III. The same constraint is valid for wrapper methods.

TABLE III. MEAN VALUES OF PRP FOR MFSO-SINGLE

	NaiveBayes	C4.5	PART	BayesNet	IB3
NaiveBayes	**0.9808**	0.9700	0.9665	0.9691	0.9600
C4.5	0.9639	**0.9923**	0.9542	0.9570	0.9422
PART	0.9631	0.9630	**0.9816**	0.9768	0.9697
BayesNet	0.9645	0.9601	0.9693	**0.9820**	0.9592
IB3	0.9581	0.9841	0.9587	0.9634	**0.9830**

Meta-feature subsets for each classifier are listed in repository http://genome.ifmo.ru/files/papers_files/FRUCT2015/Subsets.pdf. These subsets are very dissimilar to each other, despite some meta-features (such as noise signal ration and maximum mutual information) are presented in almost each meta-feature subset.

We conducted a more detailed research on meta-feature significance. We define a measure we call mean significance S_C of meta-feature f in the following way: we run a meta-feature selection algorithm T times and for every time when f was selected we add PRP of the learnt classifier C with the selected meta-feature subset to the value of the S_C:

$$S_C(f) = \frac{1}{T}\sum_{t=1}^{T} PRP(C(F_{(t)}))[f \in F_{(t)}],$$

where T is the number of runs, $F_{(t)}$ is a meta-feature subset returned by the feature selection algorithm on tth run and $[b]$, where b is a boolean formula, is defined as $[b] = 1$ if b and 0 if $\neg b$.

We take T equal 10. Top ten meta-features with the highest value of mean significance are listed in Table IV. Mean significance of all the other meta-features as well as significance for each classifier can be found in repository http://genome.ifmo.ru/files/papers_files/FRUCT2015/Significance.pdf.

TABLE IV. TEN META-FEATURES WITH THE HIGHEST VALUE OF MEAN SIGNIFICANCE

Meta-feature	**Mean significance**
MaxMutual Information	0.6527
NumberOfFeatures	0.5135
MinOneTenth PerceptronWeightSum	0.4843
MaxOneTenthBestK	0.4794
MinHalf PerceptronWeightSum	0.4667
DataSetDimensionality	0.4644
NumberOfInstances	0.4586
MinSqrt PerceptronWeightSum	0.4491
EquivalentNumber OfFeatures	0.4487
MeanMutual Information	0.4449

E. Meta-feature subset optimization for the general case

Meta-feature subset optimization for a general case (MFSO-general) is a classifier-independent method. To achieve this, we take an average value of all PRP values for each classifier. In this case the feature selection algorithm is not being adjusted to characteristics of a certain algorithm, but keeps only the features that are universal for all the classifiers we use for testing.

Only 18 meta-features are left after applying this algorithm:

- DataSetDimensionality;
- MeanSkewness;
- MeanNormalizedFeatureEntropy;
- MaxMutualInformation;
- PrunedTreeDevAttr;
- PrunedTreeMinAttr;
- PrunedTreeMinBranch;
- PrunedTreeDevClass;
- PrunedTreeMaxClass;
- UnprunedTreeLeavesNumber;
- UnprunedTreeMaxAtt;
- UnprunedTreeMaxLevel;
- UnprunedTreeMinBranch;
- UnprunedTreeDevClass;
- MinSqrtBestK;
- MinOneTenthBestK;
- MeanHalfBestK;
- MinHalfPerceptronWeightSum.

We must note that most of the selected meta-features are model-based.

Comparison of MFSO-general performance with original meta-feature set as well as the best of group choice from Table II is represented in Table V. It shows that MFSO-general outperforms all the other approaches.

TABLE V. COMPARISON ON RPR ON DIFFERENT SUBSETS WITH MFSO-GENERAL

Meta-features	NaiveBayes	C4.5	PART	BayesNet	IB3
Original	0.9738	0.9653	0.9635	0.9714	0.9530
Group choice	0.9738	0.9725	0.9653	0.9714	0.9620
MFSO general	**0.9739**	**0.9841**	**0.9720**	**0.9755**	**0.9725**

Comparison of MFSO-single performance with the other approaches to retrieve meta-feature subset are presented in Table VI. As it can be expected, all the MFSO-algorithm outperforms MFSO-general due to their adjustment to the specific classifier.

TABLE VI. COMPARISON ON RPR ON DIFFERENT SUBSETS WITH MFSO-GENERAL AND MSFO-SINGLE

Meta-features	NaiveBayes	C4.5	PART	BayesNet	IB3
Original	0.9738	0.9653	0.9635	0.9714	0.9530
Group choice	0.9738	0.9725	0.9653	0.9714	0.9620
MFSO general	0.9739	0.9841	0.9720	0.9755	0.9725
MFSO single	**0.9808**	**0.9923**	**0.9816**	**0.9820**	**0.9830**

Now we estimate average significance, which is composition of significance for each classifier. Ten meta-features with the highest average significance are listed in Table VII.

Average of all the other significance can be found in repository http://genome.ifmo.ru/files/papers_files/FRUCT2015/Av_significance.pdf.

The meta-features we have already mentioned in Subsection IV-D, which are noise signal ration and maximum mutual information, are in this top. It is worth to note that lists of mean and average significance values are similar. That allows

TABLE VII. TEN META-FEATURES WITH THE HIGHEST VALUE OF AVERAGE SIGNIFICANCE

Meta-feature	Average significance
MaxMutualInformation	0.7973
MaxOneTenthBestK	0.6124
MeanMutualInformation	0.5640
NoiseSignalRatio	0.5543
UnprunedTreeMaxClass	0.5542
MinOneTenthPerceptronWeightSum	0.5346
EquivalentNumberOfFeatures	0.4959
NumberOfFeatures	0.4958
PrunedTreeMinBranch	0.4861
UnprunedTreeMeanAttr	0.4667

us to claim that noise signal ration and maximum mutual information are more important meta-features for feature selection algorithm recommendation than correlation coefficient or class entropy.

V. DISCUSSION

A. Time-consuming evaluation of meta-features based on parameter tuning

Algorithm parameter tuning is known to take much time, therefore evaluation of the meta-feature described in Section III which are based on these parameters is costly. The time spent on new dataset meta-features extraction is usually greater than the time spent on meta-learning system prediction computation. That is why we are interested in working with fast-evaluated meta-features. In this subsection we suggest an approach to reduce the time spent on evaluation of best-parameter-based meta-features.

The fast modern approach for parameter optimization is sequential based model optimization (SMBO) [38]. Therefore, we may use only an approximation of the best parametrization which we will get after applying a number of steps in SMBO. Also meta-learning is applied for predicting certain details in SMBO, such as initial values [39]. The corresponding meta-features may be used instead of best-parameter-based features.

B. Trade-off between significance and evaluation time

Nevertheless, the approach described above is only a particular way to handle a certain type of meta-features. Development of meta-learning systems and creation of new meta-features will definitely lead to existence of highly significant, but expensive-to-evaluate meta-features. To illustrate this idea, we give an example of the most significant meta-feature, which is the best algorithm. If we can evaluate this meta-feature for a dataset, then the prediction of the best algorithm is trivial: we should only return the value of that feature. But in order to evaluate this meta-feature value, we need to spend the time equal to the exhaustive algorithm search time. Therefore, a trade-off between using significant and fast-to-evaluate meta-features should be found.

We see two solutions which can be suggested. The first solution is to measure each meta-feature both with its significance and its evaluation time. This allows to reduce meta-feature selection problem to multi-criteria optimization problem. The second solution is to allow active feature evaluation: split meta-feature set to fast-to-evaluate and other meta-features. Then evaluate meta-features from the first part. If the algorithm performance is not high enough, then try to evaluate new features which are expected to improve it.

C. When evaluation time is not that meaningful

This section is not complete without mentioning the case when meta-feature evaluation time is not critical. This case is algorithm comparison based on meta-feature space. The main idea of this approach is to find for each algorithm its area of competence, on which this algorithm is the best with respect to a fixed performance measure [40], [12]. Therefore, in order to find if a new algorithm is good or bad, we do not need it to be compared with several other algorithms. The only thing we should do is to find its area of competence and to estimate, how likely real-world datasets will appear in this area.

The space in which such areas are located is described with meta-features. The key challenge in this approach is to find a proper meta-feature description. Evaluation time is a secondary question. Therefore, best-parameter-based meta-features are applicable for algorithm comparison.

VI. CONCLUSION

In this paper we have suggested a novel approach for creating meta-features based on classifier model and the best parametization of classifier. We expect this approach to be applicable for a classifer selection as well. We have found an optimal meta-feature subset for each of the classifiers we used to evalute feature selection algorithm performance and for the general case which is based on averaging different classifiers evaluation.

For future work we plan to investigate which model-based meta-features can be evaluated in small time. Another challenge is to predict best feature selection algorithm for clustering.

This work was financially supported by the Government of Russian Federation, Grant 074-U01.

ACKNOWLEDGMENT

Authors would like to thank Igor Buzhinsky, Daniil Chivilikhin, and Veronika Minina for useful comments.

REFERENCES

[1] R. Bellman, *Adaptive control processes: a guided tour*. Princeton university press Princeton, 1961, vol. 4.

[2] S. Cateni, M. Vannucci, M. Vannocci, and V. Colla, “Variable selection and feature extraction through artificial intelligence techniques,” *Multivariate Analysis in Management, Engineering and the Science*, pp. 103–118, 2012.

[3] S. Li, R. Xia, C. Zong, and C.-R. Huang, “A framework of feature selection methods for text categorization,” in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, 2009, pp. 692–700.

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

[5] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu, “Advancing feature selection research,” *ASU feature selection repository*, pp. 1–28, 2010.

[6] J. R. Rice, “The algorithm selection problem,” 1975.

[7] D. H. Wolpert and W. G. Macready, “No free lunch theorems for search,” Technical Report SFI-TR-95-02-010, Santa Fe Institute, Tech. Rep., 1995.

[8] ——, “No free lunch theorems for optimization,” *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 67–82, 1997.

[9] D. H. Wolpert, “The supervised learning no-free-lunch theorems,” in *Soft Computing and Industry*. Springer, 2002, pp. 25–42.

[10] C. Schaffer, “A conservation law for generalization performance,” in *Proceedings of the 11th international conference on machine learning*, 1994, pp. 259–265.

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

[12] K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis, “Towards objective measures of algorithm performance across instance space,” *Computers & Operations Research*, vol. 45, pp. 12–24, 2014.

[13] D. Whitley, “Blind no more: Constant timenon-random improving moves and exponentially powerful recombination,” in *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*. ACM, 2015, pp. 391–407.

[14] L. Kotthoff, I. P. Gent, and I. Miguel, “A preliminary evaluation of machine learning in algorithm selection for search problems,” in *Fourth Annual Symposium on Combinatorial Search*, 2011.

[15] F. Hutter, D. Babić, H. H. Hoos, and A. J. Hu, “Boosting verification by automatic tuning of decision procedures,” in *Formal Methods in Computer Aided Design, 2007. FMCAD’07*. IEEE, 2007, pp. 27–34.

[16] N. Musliu and M. Schwengerer, “Algorithm selection for the graph coloring problem,” in *Learning and Intelligent Optimization*. Springer, 2013, pp. 389–403.

[17] A. Zabashta, I. Smetannikov, and A. Filchenkov, “Study on meta-learning approach application in rank aggregation algorithm selection,” 2015, pp. 115–117.

[18] R. G. Soares, T. B. Ludermir, and F. A. De Carvalho, “An analysis of meta-learning techniques for ranking clustering algorithms applied to artificial data,” in *Artificial Neural Networks–ICANN 2009*. Springer, 2009, pp. 131–140.

[19] M. C. De Souto, R. B. Prudencio, R. G. Soares, R. G. De Araujo, I. G. Costa, T. B. Ludermir, A. Schliep *et al.*, “Ranking and selecting clustering algorithms using a meta-learning approach,” in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 3729–3735.

[20] R. B. Prudêncio and T. B. Ludermir, “Active meta-learning with uncertainty sampling and outlier detection,” in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 346–351.

[21] C. Soares, P. B. Brazdil, and P. Kuba, “A meta-learning method to select the kernel width in support vector regression,” *Machine learning*, vol. 54, no. 3, pp. 195–209, 2004.

[22] M. Matijaš, J. A. Suykens, and S. Krajcar, “Load forecasting using a multivariate meta-learning system,” *Expert systems with applications*, vol. 40, no. 11, pp. 4427–4437, 2013.

[23] A. Orlov, “Non-stationary time series forecasting on basis of analysis and prediction of forecasting models efficiency,” *Brain*, vol. 900, p. 100, 2013.

[24] 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.

[25] 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.

[26] O. Kazík, K. Pešková, M. Pilát, and R. Neruda, “Meta learning in multi-agent systems for data mining,” in *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02*. IEEE Computer Society, 2011, pp. 433–434.

[27] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[28] C. Castiello, G. Castellano, and A. M. Fanelli, “Meta-data: Characterization of input features for meta-learning,” in *Modeling Decisions for Artificial Intelligence*. Springer, 2005, pp. 457–468.

[29] A. Kalousis and M. Hilario, *Feature selection for meta-learning*. Springer, 2001.

[30] H. Bensusan and C. Giraud-Carrier, “Discovering task neighbourhoods through landmark learning performances,” in *Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 325–330.

[31] Y. Peng, P. A. Flach, C. Soares, and P. Brazdil, “Improved dataset characterisation for meta-learning,” in *Discovery Science*. Springer, 2002, pp. 141–152.

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

[33] “Weka 3: Data mining software in java,” http://www.cs.waikato.ac.nz/ml/weka/, accessed: 2015-09-30.

[34] “Weka cfssubseteval class javadoc and descripton with refrerences,” http://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/attributeSelection/CfsSubsetEval.html, accessed: 2015-09-30.

[35] “Weka consistencysubseteval class javadoc and descripton with refrerences,” http://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/attributeSelection/ConsistencySubsetEval.html, accessed: 2015-09-30.

[36] “Weka assearch class javadoc and descripton,” http://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/attributeSelection/ASSearch.html, accessed: 2015-09-30.

[37] I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, and S. J. Cunningham, “Weka: Practical machine learning tools and techniques with java implementations,” 1999.

[38] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[39] M. Feurer, J. T. Springenberg, and F. Hutter, “Using meta-learning to initialize bayesian optimization of hyperparameters,” in *ECAI workshop on Metalearning and Algorithm Selection (MetaSel)*, 2014, pp. 3–10.

[40] M. R. Smith, A. White, C. Giraud-Carrier, and T. Martinez, “An easy to use repository for comparing and improving machine learning algorithm usage,” *arXiv preprint arXiv:1405.7292*, 2014.