Towards Automatic Generation Of Metafeatures

Fábio Pinto, Carlos Soares and João Mendes-Moreira

INESC TEC/Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias, s/n
Porto, Portugal 4200-465
fhpinto@inesctec.pt csoares@fe.up.pt jmoreira@fe.up.pt

Abstract. The selection of metafeatures for metalearning (MtL) is often an *ad hoc* process. The lack of a proper motivation for the choice of a metafeature rather than others is questionable and may originate a loss of valuable information for a given problem (e.g., use of *class entropy* and not *attribute entropy*). We present a framework to systematically generate metafeatures in the context of MtL. This framework decomposes a metafeature into three components: meta-function, object and post-processing. The automatic generation of metafeatures is triggered by the selection of a meta-function used to systematically generate metafeatures from all possible combinations of object and post-processing alternatives. We executed experiments by addressing the problem of algorithm selection in classification datasets. Results show that the sets of systematic metafeatures generated from our framework are more informative than the non-systematic ones and the set regarded as state-of-the-art.

Keywords: Metalearning, Systematic Metafeatures, Algorithm Selection, Classification

1 Introduction

A central task in the data mining process is the selection and training of a learning algorithm on a dataset. Given the number of learning algorithms available, this task can become very time consuming, especially if the data analyst does not have the necessary experience to focus on the most promising ones. Therefore, there is a need for systems that automate this process and guide the data analyst in the search for a learning algorithm that better suits a given dataset [1]. Such systems must reduce the amount of time for model development without significant loss of model performance when compared to the best learning algorithm. Metalearning (MtL) is one approach that can be used to address this need. Brazdil et al. defined MtL as the study of principled methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes [2].

Although the MtL literature proposes many metafeatures of different types for a wide range of problems (e.g. statistics and landmarks), most of those metafeatures are developed in a *ad hoc* way. For instance, some papers report the use of the entropy function applied to the target atribute in classification problems, i.e. the *class entropy* metafeature, but only a few use metafeatures based on the application of the same function to independent attributes, i.e. *attribute entropy* [3]. Very often, there is no justification for such options. We claim that the literature lacks an unifying framework to categorize and develop metafeatures. Therefore, this paper proposes one such framework to support the systematic generation of metafeatures for MtL problems.

Our proposal is a framework that decomposes a metafeature into three fundamental components: meta-function, object and post-processing functions. A meta-function (e.g. entropy) is applied to an object (e.g. set of independent variables) and the result is post-processed (e.g. average value), resulting in a metafeature (e.g. average attribute entropy). This decomposition enables the systematic generation of sets of metafeatures by applying the meta-function to all possible objects and process the result with all the possible post-processing functions.

In the experiments described in this paper, we use three meta-functions to generate systematic metafeatures: entropy, mutual information and correlation. We compare our approach with state-of-the-art metafeatures: the set of simple, statistical and information-theoretic metafeatures proposed by Brazdil et al. [3], landmarkers [4] and the pairwise meta-rules proposed by Sun and Pfahringer [5]. We address the problem of selecting the best algorithm for a classification dataset.

This paper is organized as follows. Section 2 describes the state-of-the-art in MtL regarding applications and metafeatures. In Section 3 we present the framework that supports systematic generation of metafeatures and we use it to decompose several metafeatures proposed in the literature. In Section 4 we present the results of the experiments that we carried out. Finally, Section 5 presents the conclusions and indicates some directions for future work.

2 Metalearning

Figure 1 illustrates a common MtL framework for algorithm recommendation, step by step. The process starts with a collection of datasets and learning algorithms. For each of those datasets, we extract metafeatures that describe their characteristics (A). Then, each algorithm is tested on each dataset and its performance is estimated (B). The metafeatures and the estimates of performance are stored as metadata. The process continues by applying a learning algorithm (a meta-learner) that induces a meta-model that relates the values of the metafeatures with the best algorithm for each dataset (C). Given the metafeatures of a new dataset (D), this meta-model is used to recommend one or more algorithms for that dataset (E).

As in any other ML task, the success of the application depends on the ability to include informative (meta)features in the data. The literature clearly groups metafeatures into three types: 1) simple, statistical and information-theoretic 2) model-based and 3) landmarkers [2]. In the first group we can find

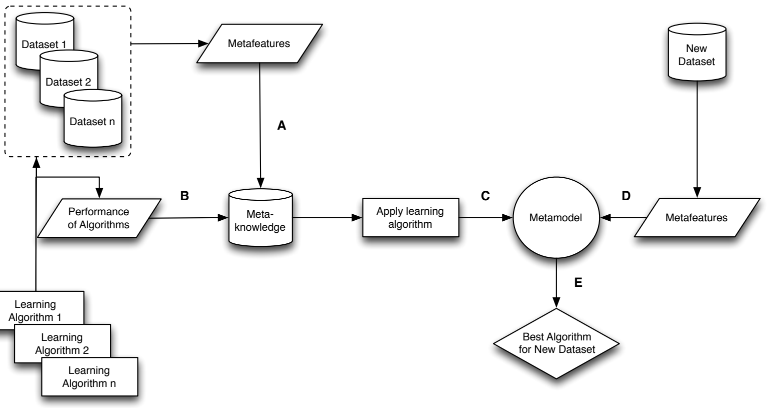

Fig. 1. Metalearning framework for algorithm recommendation.

the *number of examples* of the dataset, *correlation between numeric attributes* or *class entropy*, to name a few. Application of these kinds of metafeatures provides not only informative metafeatures but also interpretable knowledge about the problems [3]. The model-based ones [6] capture some characteristics of a model generated by applying a learning algorithm to a dataset, e.g., the *number of leaf nodes of a decision tree*. Finally, landmarkers [4] are generated by making a quick performance estimate of a simple learning algorithm in the dataset. For instance, the predictive performance of a Decision Stump.

The main focus of MtL research has been the problem of algorithm recommendation and is most commonly applied to classification problems. Brazdil et al. [3] proposed an approach that provides recommendations in the form of rankings of learning algorithms. They used simple, statistical and information-theoretic metafeatures. Sun and Pfahringer [5] extended the work of Brazdil et al. with two main contributions: the pairwise meta-rules (PMR), a higher-level type of metafeatures generated by comparing the performance of individual base learners in a one-against-one manner; and a new meta-learner for ranking algorithms. Besides PMR, they characterized datasets mainly with landmarkers.

However, MtL has also been used in other applications: time series forecasting [7], parameter tuning [8], data streams [9, 10] and others [2]. This lead to a large set of metafeatures proposed in the literature for very different problems. It is common to find discrepancies between the use of a function such as entropy or mutual information to measure exclusively a specific object. Often, it is mandatory to adapt the set of metafeatures to the problem domain. For instance, metafeatures that characterize the target feature in regression cannot be used directly in classification. We believe that it would be useful to decompose all these metafeatures into a common framework. Furthermore, such framework must also help the MtL user to systematically develop new metafeatures.

3 Systematic Generation Of Metafeatures

In this section, we propose a framework to enable a systematized and standardized development of metafeatures for MtL problems. The framework decomposes a metafeature into three fundamental components: meta-function, object and post-processing. Figure 2 illustrates the framework.

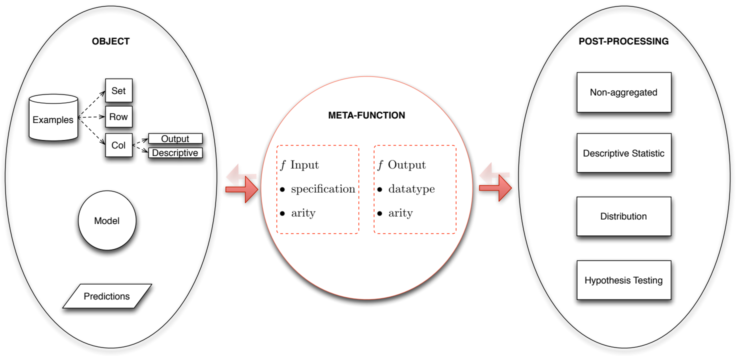

Fig. 2. Framework for systematic development of metafeatures.

The key component of the framework is the meta-function, *f*. This component is selected by the user according to his/her knowledge of the MtL problem. Although we acknowledge that this choice may be based on an *ad hoc* decision, the interest of a meta-function for a MtL problem should be well motivated. For example, entropy is a concept used in several Machine Learning algorithms, including decision trees [11]. Therefore, it is expected to be useful to better understand the learning behavior of those algorithms. Furthermore, this decision is made at a more abstract level than the typical design of metafeatures and is, thus, easier. For instance, it is indisputable that the concept of entropy is important for learning decision trees and that it is likely that some of the metafeatures that can be based on this function contain useful information on the behavior of tree learning algorithms. It is less clear that *class entropy* is useful and *attribute entropy* is not, or vice-versa. Finally, given the choice of a meta-function, the methodology generates metafeatures that characterize all components of the data with which it is compatible. This makes sure that the metafeatures that contain useful information, if any, will be generated.

The formal definition of f is given in Definition 1.

Definition 1 *A meta-function f is defined as*

$$f : X \rightarrow Y. \quad (1)$$

X is formed according to the data specification and the arity suitable to f input and the data specification and the arity of the O elements that are available

for the problem. Given O as a type of object about a learning problem, namely, a set of examples, a row, an output column, a descriptive column, a model or a set of predictions, X is any set composed of those six object; and Y is the set of all possible results of applying f to X.

The selected meta-function has intrinsic characteristics that affect the metafeatures that can be generated from it. Particularly, the *data specification* and the *arity* of meta-function's input, and the *datatype* and *arity* of the meta-function's output. *Arity* corresponds to the number of inputs of a function. For instance, the arities of entropy and mutual information are, respectively, 1 and 2. Regarding the *data specification*, we use the data mining ontology proposed by Panov et al. [12] to guide the characterization. We take into account the role of the element in the context of the learning problem, i.e., if the column corresponds to *descriptive* data or *output* data. For instance, entropy is a meta-function that allows *discrete descriptive data*, *boolean descriptive data*, *discrete output data* and *boolean output data* as input. The arity of the objects X need to be equal to 1 in order to suit the meta-function's input. These characteristics are used to identify which of the available objects X can be used to generate metafeatures.

Let us present some examples of objects. The *examples* type can be detailed into a *set* of examples (e.g., all the examples of the dataset, a bootstrap sample of the examples, a specific subset, etc), a single *row* or a *column*. The *column* can also be detailed into a column that represents an *output* variable or a column that represents a *descriptive* one. For the *model* type, we refer to all the information that can be measured regarding its induction and final output variable (e.g., the value of a parameter or a characteristic of the model - number of leaf nodes of a decision tree or the number of support vectors of a SVM). Finally, the *predictions* type considers all the information that can be extracted from the output of a predictive model when used to predict. For instance, to compute landmarkers, it is necessary two types of objects: an *output* column and a set of *predictions*. We give more detailed examples in Section 3.1.

The post-processing function p concerns the aggregation of the meta-function output, Y. Formally, p is defined in Definition 2.

Definition 2 *A post-processing function p is defined as*

$$p : Y \rightarrow MF \tag{2}$$

where MF is the set of all possible metafeatures. The datatype and arity of Y defines implicitly the post-processing function p that can be used to form a metafeature.

For the meta-function's output Y, we need to characterize its *datatype* (real, discrete, boolean, etc) and *arity*. Same thing for each post-processing function p. If the characteristics of Y match with the ones from p, a metafeature is formed. For instance, if the meta-function entropy is applied to n discrete descriptive attributes, the *datatype* of Y is real and the *arity* is n. This allows to generate metafeatures using post-processing functions such as *mean*, *maximum value*, *standard deviation*, *histogram bins*, etc.

Our framework splits p into four groups: *non-aggregated*, *descriptive statistic*, *distribution* and *hypothesis testing.* The *non-aggregated* alternative uses the meta-function output in its raw state directly as metafeature(s). In some MtL problems it might be useful not to aggregate the information. This is particularly frequent in MtL applications such as time series or data streams where the data has the same morphology [9] or when the MtL algorithm is relational [13]. For instance, instead of computing the mean of the correlation between pairs of numerical attributes, one could use the correlation between all pairs of numerical attributes. It can also be the case that Y does not need aggregation and, therefore, the non-aggregated post-processing function is applied.

The *descriptive statistic* case is perhaps the most common approach to aggregate information and generate a metafeature. This can be accomplished by using the *mean*, *maximum*, *minimum*, *standard deviation*, *mode*, etc. However, such aggregation can cause loss of valuable information.

Another option is the *distribution* alternative. It captures a representation of the output provided by the meta-function by characterizing its distribution. This fine-grained aggregation can be achieved through the use of *histograms* with a fixed number of bins as proposed in [14]. In this case, each bin is used as metafeature, providing a description of the distribution of the meta-function's output.

Finally, the *hypothesis testing* subcomponent. Here, the output Y provided by the meta-function f is used to test an assumption. For instance, it can test whether the values of Y follow a normal distribution. The output of this test (it can be a p-value or a nominal variable) is used as metafeature.

From definitions 2 and 1, a metafeature $mf \in MF$ can be defined as

$$mf = p(f(x)) \tag{3}$$

where $x \in X$.

3.1 Decomposing Metafeatures

A first test to the validity of the proposed framework is to check if existing metafeatures could be the result of its use. We show examples from three types of metafeatures: simple, statistical and information-theoretic; model-based and landmarkers.

Figure 3 illustrates the decomposition of five metafeatures. For instance, the *absolute mean correlation between numeric attributes* is very similar to the *correlation between numeric attributes* (used in data streams applications [9]) except for the post-processing alternative. In this case, the application domain makes it feasible and potentially more informative to not aggregate the correlation values.The framework decomposes the computation of the metafeatures in detail. Furthermore, it allows the comparison between two or more metafeatures.

Still regarding Figure 3, the decomposition of the two last metafeatures shows that is possible to use the framework for more complex metafeatures. The *number*

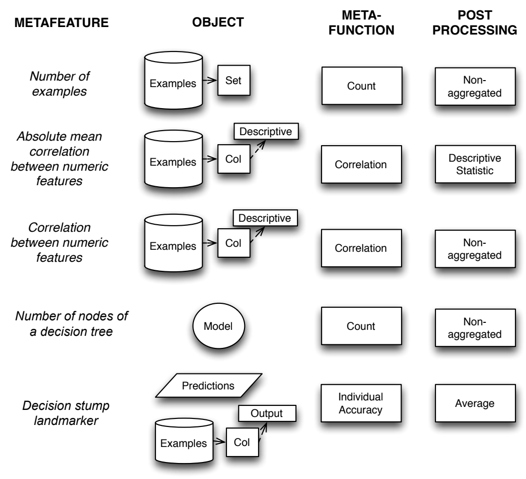

Fig. 3. Metafeatures decomposed using our framework.

of nodes of a decision tree is an example of a model-based metafeature. The object component is the decision tree model, the meta-function is count and the post-processing option is non-aggregated. Peng et al. [6] propose several model-based metafeatures (for decision trees models) of this kind. Finally, we also show an example of a landmarker. The *decision stump landmarker*, uses as object a set of predictions and the output column of the dataset. In the example given, the meta-function is individual accuracy. As post-processing function, the most common one is average. However, it could be used another, such as histogram bins. This kind of flexibility is one of the advantages of our framework.

4 Experiments

The experiments that we present in this Section aim to answer two questions.

– 1) Is the proposed framework able to develop sets of systematic metafeatures that are consistently more informative than non-systematic sets?
– 2) Is the set of systematic metafeatures computed with the proposed framework more informative than the state-of-the-art metafeatures?

For both questions, we executed experiments by addressing the problem of selecting the best algorithm in a set with a classification perspective [4, 14].

4.1 Experimental Setup

For the first question, we carried out a set of experiments with the goal of providing a proof of concept of our framework. By testing whether the systematic generation of metafeatures increases the amount of information of a set of metafeatures, we show that our framework can be useful and help MtL users to avoid an *ad hoc* selection of metafeatures. For the second question, we executed experiments in which we compare the set of metafeatures generated by our framework with a set of metafeatures regarded as state-of-the-art All the experiments were executed on 58 UCI classification datasets [15]. The selection of the datasets was done randomly. To speed up the experiments, we limited the number of instances in larger datasets to a maximum of 5000 instances with stratified random sampling.

Six classification algorithms were tested as base learners: NaiveBayes, k-NN, C5.0, CART, SVM (with RBF kernel) and Random Forest. The estimates of algorithm performance were done using 10-fold cross validation and accuracy as error measure.[1]

We tested three different meta-learners: C5.0, SVM (with RBF kernel) and Random Forests. Again, the estimates of performance of the meta-learners were done using 10-fold cross validation (with 30 repetitions) and accuracy is the performance measure. As baseline, we use the default class of the training set. For statistical validation we used the methodology proposed by Demšar [16]: Friedman rank test with Nemenyi test for post-hoc multiple comparisons.

We compare our approach with a set of metafeatures that are widely used in the field. Brazdil et al. [3] proposed the following set of metafeatures of the so called simple, statistical and information-theoretic: *number of examples*, *proportion of symbolic attributes*, *proportion of missing values*, *proportion of attributes with outliers*, *entropy of classes*, *average mutual information of class and attributes* and *canonical correlation of the most discriminating single linear combination of numeric attributes and the class distribution*. To this set of simple, statistical and information-theoretic metafeatures we also added the *absolute average correlation between numeric attributes*. Finally, we included two landmarkers: *Decision Stump sub-sampling landmarker* and a *Naive Bayes sub-sampling landmarker*. We call this set of 10 metafeatures the *Traditional* one.

In order to provide a fair comparison, we designed metafeatures based as much as possible on the concepts involved in the *Traditional* metafeatures. Therefore, the set of objects consists on the dataset and on two sets of predictions: obtained with Naive Bayes and Decision Stump. As post-processing functions p, we used: mean, weighted mean, standard deviation, variance, minimum, maximum and histogram bins. We generated four sets of systematic metafeatures by selecting three meta-functions: entropy, mutual information and correlation, resulting in 20, 36 and 19 metafeatures, respectively. The fourth set includes *all* the metafeatures from the three previous sets (75 in total). The choice of the meta-functions was not done randomly. We used meta-functions that are used in the *Traditional* set with a non-systematic approach. Our approach is to use the very same meta-functions together with our framework to generate systematic sets of metafeatures.

[1] The estimates of performance showed that NaiveBayes was better in 4 datasets, k-NN in 9, C5.0 in 23, CART in 2, SVM in 14 and Random Forest in 6.

One of the disadvantages of using our framework to generate systematic metafeatures is the curse of dimensionality. The number of metafeatures generated is usually very high. Since MtL applications dont have a large number of examples, this can be emphasized by our approach. So, we rely on two feature selection algorithms to tackle this problem: ReliefF [17] and correlation feature selection (CFS).

4.2 Systematized vs Unsystematized

The Critical Difference (CD) diagrams generated with the first set of experiments is presented in Figure 4. We set $\alpha = 0.05$ for all experiments.

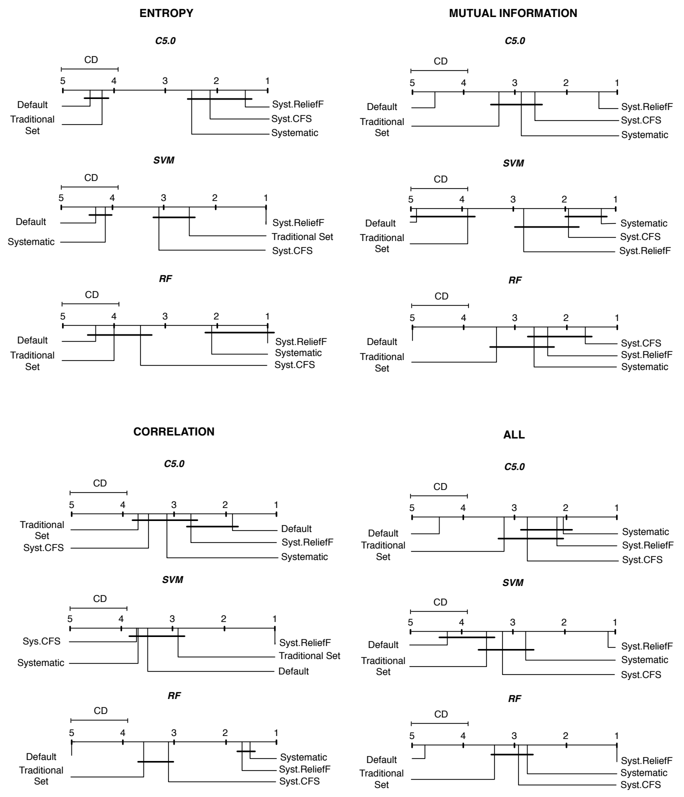

Fig. 4. Critical Difference Diagrams for the meta-functions: *entropy*, *mutual information*, *correlation* and *all.*

The *Traditional* set in those experiments changes according to the meta-function which is being compared, to maximize the fairness of the comparison, as discussed earlier. For instance, the *Traditional* set used against the systematic metafeatures of the entropy meta-function consists of the following metafeatures: *entropy of classes*, *average entropy of symbolic attributes* and the two landmarkers mentioned above. Similarly, if the meta-function for generating systematic metafeatures is correlation, the *Traditional* set consists of the *absolute average correlation between numeric attributes* and, again, the two landmarkers.

Overall, the metafeatures generated from our framework present superior performance. This result is consistent regarding the meta-function and the meta-learner. However, it is noticeable that the results obtained with the correlation function are worst than in the other cases. It is probably related to the fact that this meta-function cannot be applied to the target variable, as this is a nominal variable and the input to the function must be numerical.

The set of metafeatures generated both with the meta-functions entropy and mutual information present good results when compared with the *Traditional* set. The combination of the *Systematic* metafeatures with the feature selection algorithm ReliefF presents a very good average rank in almost all CD diagrams. This result is consistent across different meta-learners.

Although this is not shown on the CD diagrams, the average accuracy obtained with the *all* set is lower than with the *entropy* and *mutual information* set. This suggest that the curse of dimensionality does affect our methodology, as expected. Since in the *all* set we gathered all the metafeatures generated from the entropy, mutual information and correlation meta-functions, the number of metafeatures is much higher. This makes the task of the feature selection algorithms more difficult. Nevertheless, the results obtained are still better than the baseline and the *Traditional* set.

4.3 Systematized vs State-of-the-art

Sun and Pfahringer [5] proposed the pairwise meta-rules (PMR), a metafeature generation method based on rules that compare the performance of individual base learners in a one-against-one manner. Adding the PMR to the sets of systematic metafeatures the probability that the results will be affected by the curse of dimensionality. For each pair of algorithms (since we test 6 base-learners, we have 15 pairwise comparisons) the method generates on average two PMR. So, using PMR implies adding 30 new metafeatures.

The *Traditional* set used in this experiments comprises the 10 metafeatures mentioned before in the beginning of Section 4. To prevent an unfair advantage of our approach, we do not use the results of the previous experiments, in which the sets based on the entropy and mutual information meta-functions obtained the best results. Thus, we compare the all set, which includes all the metafeatures, with the state of the art approaches. We added the respective PMR to the *Traditional* and *Syst.ReliefF* sets, forming *Traditional + PMR* and *Syst.ReliefF + PMR*. We compared the four sets of metafeatures with the same meta-learners used previously. We use ReliefF as feature selection algorithm.

Figure 5 presents the results of the experiments. Comparing the sets of metafeatures, it is noticeable that those generated using our framework present a superior predictive performance in comparison with the *Traditional* sets. This difference is statistically significant. Also, the result is consistent across different meta-learners.

Regarding the addition of the PMR, the gain both on the *Traditional* or on the *Syst.ReliefF* set is not statistically significant. Overall, the sets *Syst.ReliefF* and *Syst.ReliefF + PMR* are the most informative across all meta-learners.

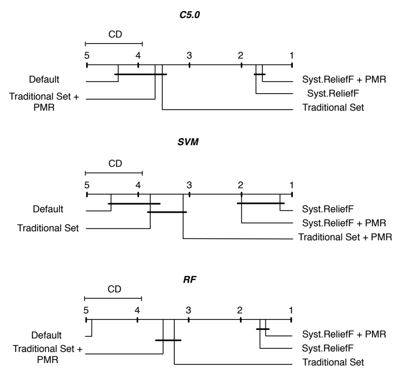

Fig. 5. Critical Difference diagrams of systematic metafeatures Vs state-of-the-art in the classification experiments.

5 Conclusion and Future Work

This paper proposes a generic framework to develop metafeatures for MtL problems. This framework is a step towards the automatic generation of metafeatures. The framework is structured in such a way that the systematic generation of metafeatures is triggered through the selection of a meta-function. Then, given the objects and post-processing functions that are available, the framework outputs a set of metafeatures generated according to the characteristics (e.g., domain of the inputs) of the selected meta-function.. The process can be repeated with several meta-functions. The selection of the meta-function is crucial and it should be chosen intuitively according to the MtL application.

Our experiments aim to answer two questions: (1) are the systematic sets of metafeatures better than the non-systematic ones? (2) are the systematic sets generated with the framework better than the state-of-the-art? In the first set of experiments, we found that the systematic metafeatures generated are consistently more informative than the non-systematic ones. In the second set of experiments, we found that the systematic sets are also more informative than state-of-the-art methods such as PMR.

As for future work, we plan to use this framework to generate systematic metafeatures for different MtL problems that we have been working on, particularly, MtL for pruning of bagging ensembles and dynamic integration of models [18].

Acknowledgments

This research has received funding from the ECSEL Joint Undertaking, the framework programme for research and innovation horizon 2020 (2014-2020) under grant agreement n 662189-MANTIS-2014-1.

References

1. Serban, F., Vanschoren, J., Kietz, J.U., Bernstein, A.: A survey of intelligent assistants for data analysis. ACM Computing Surveys (CSUR) **45**(3) (2013) 31
2. Brazdil, P., Carrier, C.G., Soares, C., Vilalta, R.: Metalearning: applications to data mining. Springer (2008)
3. Brazdil, P.B., Soares, C., Da Costa, J.P.: Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. Machine Learning **50**(3) (2003) 251–277
4. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Tell me who can learn you and i can tell you who you are: Landmarking various learning algorithms. In: International Conference on Machine Learning. (2000) 743–750
5. Sun, Q., Pfahringer, B.: Pairwise meta-rules for better meta-learning-based algorithm ranking. Machine learning **93**(1) (2013) 141–161
6. Peng, Y., Flach, P.A., Soares, C., Brazdil, P.: Improved dataset characterisation for meta-learning. In: Discovery Science, Springer (2002) 141–152
7. Prudêncio, R.B., Ludermir, T.B.: Meta-learning approaches to selecting time series models. Neurocomputing **61** (2004) 121–137
8. Reif, M., Shafait, F., Dengel, A.: Meta-learning for evolutionary parameter optimization of classifiers. Machine learning **87**(3) (2012) 357–380
9. Rossi, A.L.D., De Carvalho, A.C.P.D.L.F., Soares, C., De Souza, B.F.: Metastream: A meta-learning based method for periodic algorithm selection in time-changing data. Neurocomputing **127** (2014) 52–64
10. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: Algorithm selection on data streams. In: Discovery Science, Springer (2014) 325–336
11. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1) (1986) 81–106
12. Panov, P., Soldatova, L., Džeroski, S.: Ontology of core data mining entities. Data Mining and Knowledge Discovery **28**(5-6) (2014) 1222–1265
13. Getoor, L., Mihalkova, L.: Learning statistical models from relational data. In: ACM SIGMOD International Conference on Management of data, ACM (2011) 1195–1198
14. Kalousis, A., Theoharis, T.: Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. Intelligent Data Analysis **3**(5) (1999) 319–337
15. Blake, C., Merz, C.J.: {UCI} repository of machine learning databases. (1998)
16. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7** (2006) 1–30
17. Robnik-Šikonja, M., Kononenko, I.: Theoretical and empirical analysis of relieff and rrelieff. Machine learning **53**(1-2) (2003) 23–69
18. Pinto, F., Soares, C., Mendes-Moreira, J.: Pruning bagging ensembles with metalearning. In: Multiple Classifier Systems. Springer (2015) 64–75