Landmarking for Meta-Learning using RapidMiner

Sarah Daniel Abdelmessih[1], Faisal Shafait[2], Matthias Reif[2], and Markus Goldstein[2]

[1]Department of Computer Science
German University in Cairo, Egypt
[2]Competence Center Multimedia Analysis and Data Mining
German Research Center for Artificial Intelligence (DFKI GmbH)
D-67663 Kaiserslautern, Germany
{faisal.shafait, matthias.reif, markus.goldstein}@dfki.de

***Abstract*— In machine learning, picking the optimal classifier for a given problem is a challenging task. A recent research field called meta-learning automates this procedure by using a meta-classifier in order to predict the best classifier for a given dataset. Using regression techniques, even a ranking of preferred learning algorithms can be determined. However, all methods are based on a prior extraction of meta-features from datasets. Landmarking is a recent method of computing meta-features, which uses the accuracies of some simple classifiers as characteristics of a dataset. Considered as the first meta-learning step in RapidMiner, a new operator called landmarking has been developed. Evaluations based on 90 datasets, mainly from the UCI repository, show that the landmarking features from the proposed operator are useful for predicting classifiers' accuracies based on regression.**

I. Introduction

Predicting the performance of classifiers, ranking learning algorithms and a dynamic selection of a suitable learning algorithm for a given problem are recent research topics in machine learning [1], [2], [3]. As derived from the *No Free Lunch Theorem*, no learning algorithm can be specified as outperforming on the set of all real-world problems [4]. This implies that a learning algorithm has reasonable performance on a set of problems that is defined as its *area of expertise* [3]. In this context, *meta-learning* was developed: it relates algorithms to their area of expertise using specific problem characteristics. The idea of *meta-learning* is to learn about classifiers or learning algorithms, in terms of the kind of data, for which they actually perform well [5], [6]. Using dataset characteristics, which are called *meta-features* [7], [6], one predicts the performance results of individual learning algorithms. These features are divided into several categories [7]:

- simple/general features (number of attributes, number of categorical attributes, number of samples, ...).
- statistical features (canonical discriminant correlations, skew, kurtosis, ...) [8].
- information theoretic features (class entropy, signal-to-noise ratio, ...).

In addition, a new feature category called *landmarking* was proposed [9]. These *landmarking features* are classification accuracies of some simple but fast computable learning algorithms, mostly related to more complex classifiers.

Figure 1 gives a general schematic overview of a meta-learning system: it shows, that for a number of datasets, multiple classifiers are evaluated. From these datasets *landmarking features* are extracted as explained in Section II. All results are stored in a central *case base* as later explained in Section IV and a meta-learning model is trained. Later, if an unknown dataset q needs to be processed, *landmarking features* are extracted and using the meta-learning model, a classifier is recommended.

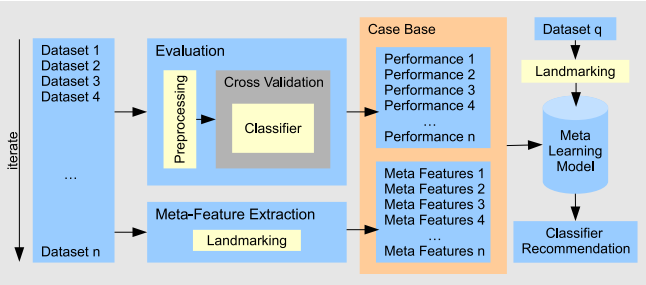

Fig. 1. Schematic view of meta-learning: on the left, the case base is created, on the right an unknown dataset is processed and a suitable classifier is recommended.

The PaREn project (Pattern Recognition Engineering)[1] aims to facilitate pattern recognition and machine learning for non-expert users. As a part of PaREn, a *landmarking operator* was developed as a first *meta-learning* step in RapidMiner [10], which is an open source system for *data mining*. In the following sections an overview of the developed *landmarking operator* and its evaluation is provided.

II. Landmarking

Every problem or dataset has certain characteristics that relates it to an area of expertise for which a specific learning algorithm exists. In this context, *landmarking features* are defined as dataset characteristics representing the performance of some simple learners on this dataset. These simple learners are called *landmarkers*.
The basic hypothesis behind *landmarking* is that the simple

[1]This project is initialized by the *German Research Center for Artificial Intelligence (DFKI)* and funded by the *Federal Ministry of Education and Research*. For more details see `http://madm.dfki.de/paren/`.

landmarkers are somehow related to more advanced and complex learners. This means that *landmarkers* or combinations thereof are able to estimate the performance of more sophisticated algorithms for a given problem. This raises the question how to choose the learners that are ideal as landmarkers. They have to satisfy the following conditions [9]:

- The algorithm has to be simple, which requires its execution time to be short, implying minimal computational complexity of the learner [11].
- The landmarkers have to differ in their bias, mechanism, property measurements, or area of expertise [3].
- Every landmarker has to be simpler than the targeted advanced learner. Otherwise the landmarker will be useless, since the targeted learning algorithms could be evaluated directly avoiding a potentially error prone prediction step and saving time [11].

The landmarkers used in the implemented RapidMiner landmarking operator are [9], [12], [13]:

- **Naive Bayes Learner** is a probabilistic classifier, based on *Bayes' Theorem*:

$$p(X|Y) = \frac{p(Y|X) \cdot p(X)}{p(Y)} \tag{1}$$

  where $p(X)$ is the prior probability and $p(X|Y)$ is the posterior probability. It is called *naive*, because it assumes independence of all attributes to each other.
- **Linear Discriminant Learner** is a type of discriminant analysis, which is understood as the grouping and separation of categories according to specific features. Linear discriminant is basically finding a linear combination of features that separates the classes best. The resulting separation model is a line, a plane, or a hyperplane, depending on the number of features combined.
- **One Nearest Neighbor Learner** is a classifier based on instance-based learning. A test point is assigned to the class of the nearest point within the training set.
- **Decision Node Learner** is a classifier based on the information gain of attributes. The information gain indicates how informative an attribute is with respect to the classification task using its entropy. The higher the variability of the attribute values, the higher its information gain. This learner selects the attribute with the highest information gain. Then, it creates a single node decision tree consisting of the chosen attribute as a split node.
- **Randomly Chosen Node Learner** is a classifier that results also in a single decision node, based on a randomly chosen attribute.
- **Worst Node Learner** is a classifier that calculates the highest information gain for a split for all the attributes. Then it chooses the attribute with the lowest information gain among all attributes to model a single node decision tree.
- **Average Node Learner** calculates the average performance of single node decision trees, where each node corresponds to one attribute.

The accuracies of the above defined algorithms are used in the following as landmarking features for the task of meta-learning.

III. Landmarking Operator in RapidMiner

The developed landmarking operator is the starting point for meta-learning using RapidMiner. It can be easily extended and used to build meta-learning processes, such as classifier recommenders or automatic algorithm selection. In this section, an overview of the design of the landmarking operator is provided.

A. RapidMiner Terminology

Before going into details of the landmarking operator, some RapidMiner terminology has to be defined:

- **Operator** is the super-class of any operator implemented in RapidMiner. To be precise, an operator is an object that accepts an array of input objects, does some operations or processes based on its defined role, and then returns an array of output objects. These inputs and outputs of an operator are delivered or received through *ports*, that can be connected to other operators' ports. Operators can have parameters as settings for its operations or processes. The processes or operations of an operator are performed in a method called *doWork*, which has to be implemented by every operator. This method is invoked when the operator is executed in order to start its processes. Moreover, an operator can be encapsulated or embedded in another operator as a subprocess.
- **ExampleSet** is an interface that represents datasets. It can be passed to operators as input or delivered by an operator as output. This ensures that example sets can be preprocessed or processed by an operator.

B. Landmarking Operator Functionality

As shown in Figure 2, the landmarking operator has two ports: an input port that should receive an *ExampleSet*, and an output port. The delivered output contains the landmarking results in the form of an *ExampleSet*, having the landmarkers as attributes, and their accuracies as attribute values. Some parameter settings are provided to make the operator adequate to problem requirements. The user has control over:

- the landmarkers to be evaluated.
- whether the landmarkers should be encapsulated into a *cross-validation* process or not. If this parameter is not enabled, the input dataset is considered as training and test set at the same time.
- the *preprocessing* of the input *ExampleSet* by *Normalization*.

By default all the landmarkers described in Section II are enabled and the normalization property is set to true with a range [0, 1]. However, since several classifiers have special properties or requirements, some landmarkers have to preprocess the input dataset or have to use the classifier in a certain way. For example, it is illogical to evaluate

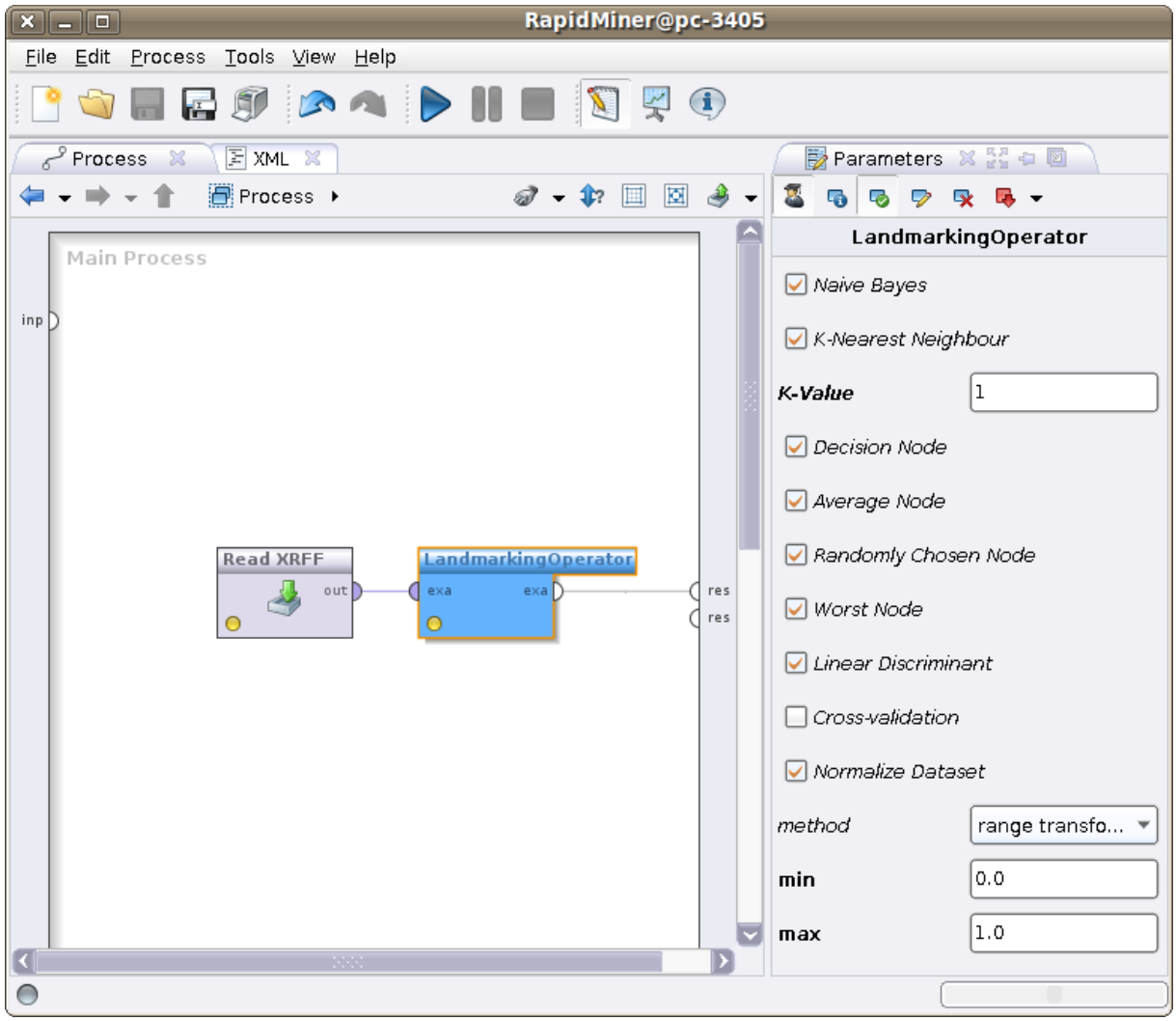

Fig. 2. The landmarking operator in RapidMiner. In the middle of the figure, the process tab is depicted. In this tab the landmarking operator is connected to an ReadXRFF operator to get an input *ExampleSet*. The parameters of the operator are represented in the Parameters tab on the right. These are the default settings of the parameters of the landmarking operator.

the One Nearest Neighbor landmarker without using cross-validation, as the prediction will be always correct by picking the point itself as the nearest neighbor, resulting in a 100% accuracy. Another specially handled landmarker is the Linear Discriminant Learner: It is set to be never validated using cross-validation, because of the inability of the RapidMiner operator to run always correctly in this case. Furthermore, some preprocessing is applied on its input *ExampleSet*. The preprocessing steps employed are:

1) Removing mappings of label values that actually do not occur in the examples of the dataset, as RapidMiner's linear discriminant operator can not handle this case.
2) Converting all *polynominal* attributes to *binominal*, then to *numerical* attributes, as the linear discriminant analysis operator does not support nominal attributes.

If cross-validation is applied on an *ExampleSet* with a total number of examples smaller than the chosen number of folds, a *leave-one-out* cross-validation is performed instead automatically.

C. Landmarking Operator Architecture

The developed landmarking operator consists of three important components, namely *AbstractLandmarker*, *LandmarkingOperator*, and *LandmarkingResults*. **AbstractLandmarker** is the super-class of all landmarkers, as illustrated in Figure 3. The main method of this class is *learnExampleSet*, which evaluates the performance of the model of the passed *Operator* on the input *ExampleSet*, according to the parameter settings described in Section III-B. Any landmarker has to be a sub-class of AbstractLandmarker, having its own operator and specific preprocessing or parameter settings. Figure 3 shows the relation between the landmarkers and RapidMiner operators. All decision node landmarkers use RapidMiner's *DecisionStumpLearner* to create their decision node. This is achieved by reducing the attribute set of their input dataset according to their definition and using the information gain of the attributes. The other landmarkers are simply mapped to their corresponding operators in RapidMiner, as depicted in Figure 3.

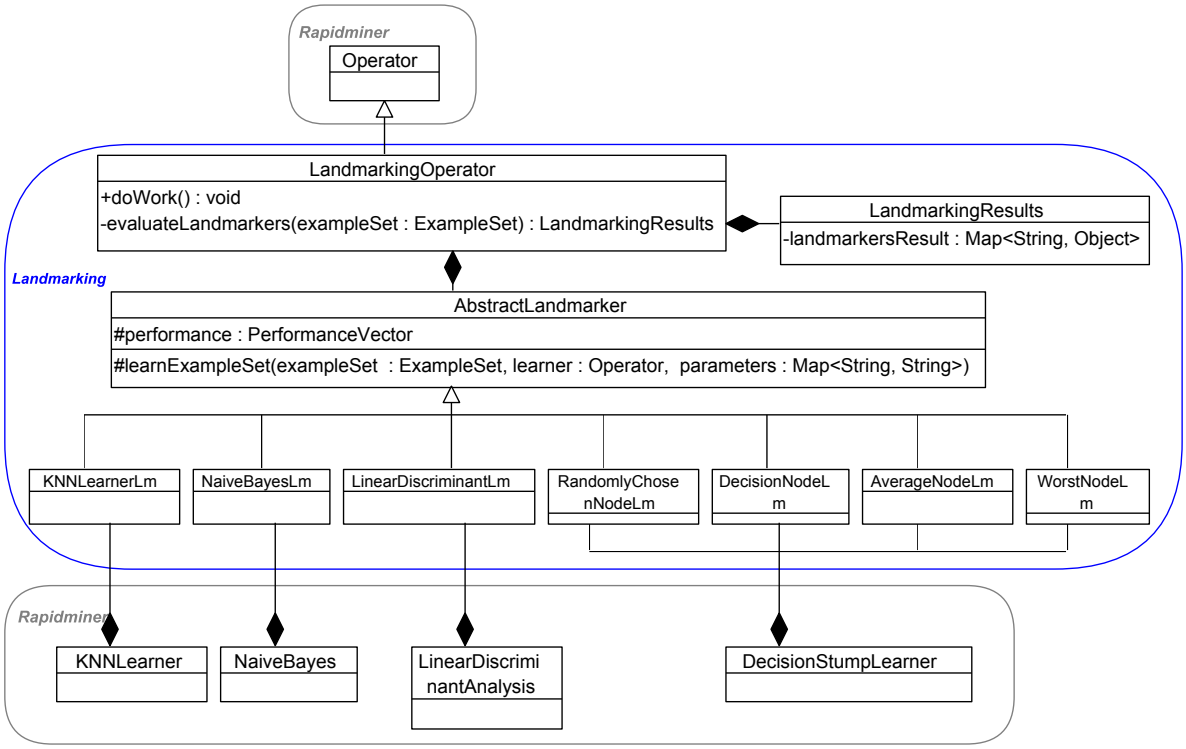

Fig. 3. Class diagram for RapidMiner's landmarking operator

To integrate the landmarkers in an operator in RapidMiner, the *LandmarkingOperator* has been developed as a sub-class of *Operator*. The interaction point between the operator and the landmarkers is the *doWork* method of the landmarking operator. By calling the major method *evaluateLandmarkers*, a map containing the parameters described in Section III-B as key is created. For any additional parameters that should be passed to a landmarker, a modification of the map can be easily done in the LandmarkingOperator class. Then, these parameters are passed to the landmarkers and each landmarker process is managed separately. At the end a *LandmarkingResults* object is returned. The result of the evaluated landmarkers is controlled by *LandmarkingResults* class. The result object contains a map, *landmarkersResult*, that is filled after the evaluation of a landmarker with its name and accuracy. The LandmarkingOperator parses the LandmarkingResults object and delivers it as an *ExampleSet* to the output port of the operator. If one is interested in results other than the accuracy (e.g., absolute error, RMSE, ...), a key and a value can be easily added to the result map.

IV. Performance Evaluation

The main goal of the evaluation is to determine whether landmarking features are useful for meta-learning or not compared to other meta-features. Accordingly, landmarking features are evaluated by predicting the accuracies of some classifiers.

A. Experiment Setup

To evaluate the landmarkers, 90 datasets from the UCI machine learning repository [14] and other sources are used. In order to have reliable accuracy estimates, all chosen datasets have more than 100 samples. RapidMiner processes were used to manage the experiments. To extract the landmarking features from the datasets, the landmarking operator (described in Section III) was used. The parameters of the operator were set according to the description provided in [9]. Accordingly, the extraction of landmarking features was performed using a 10-fold cross-validation on four landmarkers: One Nearest Neighbor[2], Decision Node, Randomly Chosen Node, and Worst Node. In addition, the datasets were normalized before evaluating the landmarkers. The target algorithms, for which the accuracy was predicted, are Naive Bayes, k-Nearest Neighbors, Multilayer Perceptron, OneR, RandomForest-Weka[3], Decision Tree, and LibSVM. They origin from different algorithm categories and were chosen because of their prominence. A grid search parameter optimization was applied on important parameters of the target classifiers. For each of the classifiers, the accuracy was computed for all the datasets. These accuracies, along with the dataset names were stored in a dataset considered as a *case base* as illustrated in Figure 1. In order to estimate the accuracies of the chosen classifiers and the confidence of their prediction, *regression*[4] was applied [8]. The *root*

[2]Note that in [9] the *elite* One Nearest Neighbour was used instead.
[3]Operator in the Weka plug-in
[4]A paper about regression for meta-learning will be published soon

mean squared error (RMSE) is used as the confidence of the prediction. In other words, it gives feedback on how close the predicted and computed accuracies are.
The experiment was divided in two stages:

1) Evaluation of the landmarkers on all datasets was performed using the presented landmarking operator with its default settings as outlined in Section III. The resultant dataset, *lm-case base*, included the evaluated landmarkers accuracy along with the dataset name as attributes. These attributes were joined with the original case base dataset.
2) To predict the accuracy values of the classifiers, a regression model was trained on the landmarking features stored in the *lm-case base* dataset using a *leave-one-out cross-validation*. *LibSVM* of type *ϵ-SVR* and *radial basis function* kernel was used for regression. Then the RMSE was calculated. To increase the reliability of the regression model, the following steps were performed:
   - Missing attributes and labels were filtered from the lm-case base dataset. The classifiers' computed accuracies were considered as dataset labels. Any example having a missing label was excluded from the dataset, as it would not improve the regression model. Furthermore, examples having missing meta-features (represented as dataset attributes) were removed.
   - Estimation of the LibSVM parameters **C** and γ was attained by optimization. *Grid search* and *leave-one-out cross-validation* were used for optimization.

B. Results and Discussion

This experiment was also applied on other meta-features, for instance, statistical, information theoretic, and simple features, in comparison to landmarking features[5]. The RMSE results of the different experiments were compared, taking into consideration the variations in the number of datasets included in the experiments, that resulted from the filtering explained previously. As an overall result, the RMSE range ([5.1%, 8.3%]) of the landmarking features experiment can be currently considered as the most confident among its counterparts, as it can be deduced from Table I. Therefore, a more detailed discussion about the experiment of landmarking features is provided below.
Out of 90 datasets used for the experiment only 86 datasets were included in the lm-case base dataset, due to the failure of the evaluation of the four datasets: *trains*, *profb*, *splice*, and *mfeat-pixel*. For the trains and profb datasets, the problem occurred when applying the preprocessing operators. For the mfeat-pixel dataset, the landmarking operator was preprocessed successfully only if one landmarker was evaluated at a time. Otherwise, it resulted in an *OutOfMemoryError*. This might have occurred due to the large number of categorical attributes (about 240) it had. The execution of the linear discriminant analysis operator on the *splice* dataset was not successful neither as an encapsulated operator in the landmarking operator nor independently as a RapidMiner operator; although the preprocessing steps described in Section IV-A were applied on the dataset in both cases.
From the results shown in Table I, the following information can be deduced:

- By filtering the missing labels[6] from the lm-case base, the number of datasets taken into consideration differed from one classifier to the other.
- Random Forest has the lowest RMSE, indicating the highest confidence of prediction. A possible reason may be the inclusion of three decision node landmarkers, as the Random Forest classifier is based on Decision Trees.
- Nearest Neighbor has the second lowest RMSE. This could be a result of the inclusion of One Nearest Neighbour landmarker.
- For OneR, the predicted accuracies were less assertive, having an RMSE of 8.3%. This was not expected, as both decision node landmarkers and OneR are based on RapidMiner's *DecisionStumpLearner*, which should enhance the predictions for OneR.

As future work it would be interesting to find out which landmarking meta-features are related to which classifiers.

	RMSE			
Classifier (samples)	Simple[i]	Statistical[i]	Information Theoretic[ii]	Landmarking
Decision Tree ($n = 61$)	0.116	0.111	0.109	**0.070**
LibSVM ($n = 61$)	0.115	0.096	0.109	**0.064**
Naive Bayes ($n = 68$)	0.136	0.121	0.123	**0.079**
Nearest Neighbor ($n = 68$)	0.123	0.105	0.109	**0.053**
Multilayer Perceptron ($n = 57$)	0.118	0.104	0.108	**0.074**
OneR ($n = 68$)	0.162	0.159	0.105	**0.083**
Random Forest ($n = 68$)	0.112	0.101	0.099	**0.051**

TABLE I

RMSE FOR THE PREDICTED ACCURACIES BASED ON DIFFERENT META-FEATURE EXTRACTION APPROACHES. NOTE, THAT FOR EACH CLASSIFIER THE NUMBER OF SAMPLES (DATASETS) DIFFERS DUE TO FILTERING OF MISSING ATTRIBUTES AND LABELS AS EXPLAINED IN SUBSECTION IV. EACH RMSE VALUE IS A MEASURE FOR THE CONFIDENCE OF PREDICTION OF A CLASSIFIER. IT WAS FOUND THAT LANDMARKING OUTPERFORMS ALL OTHER METHODS WITH RESPECT TO PREDICTION ACCURACY.

i) Features described in the STATLOG project [8].
ii) Features extracted using the METAL data characterization toolkit (DCT) [15]. More details about the METAL project can be found in `http://cordis.europa.eu/esprit/src/26357.htm`.

[5]results will be published soon

[6]Most values are missing due to the unreasonable amount of time they need to be computed.

This correlation could lead to a more reliable approach with confident results. For example, best node landmarker may be supportive for decision tree classifier, while another landmarking feature may reduce the reliability of the regression model for predicting its accuracy.

V. Conclusion

As a bottom line, our aim was to introduce the developed landmarking operator in RapidMiner as a meta-learning step. This operator extracts landmarking features from a given dataset by basically applying seven fast computable classifiers on it. Throughout our evaluation illustrated in Section IV, it was depicted that landmarking-features are well suited for meta-learning. The method employed in the evaluation for predicting some classifiers' accuracies was based on meta-features and regression. The RMSE was chosen to be an indicator for the confidence of prediction, as emphasized in Table I, illustrating satisfactory results. In this context, the landmarking features can be considered reasonable for developing systems involving classifier recommendation or automatic classifier selection.

References

[1] Pavel B. Brazdil, Carlos Soares, and Joaquim P. da Costa. Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, 2003.

[2] Carlos Soares and Pavel Brazdil. Zoomed ranking: Selection of classification algorithms based on relevant performance information. In *PKDD*, volume 1910, pages 126–135, Lyon, France, September 2000.

[3] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Revieww*, 18(2):77–95, 2002.

[4] David H. Wolpert. The supervised learning no-free-lunch theorems. In *In Proc. 6th Online World Conference on Soft Computing in Industrial Applications*, pages 25–42, 2001.

[5] Kate A. Smith-Miles. Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection. *ACM Computing Surveys (CSUR)*, 41(1):25, 2008.

[6] Shawkat Ali and Kate A. Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138, 2006.

[7] Ciro Castiello, Giovanna Castellano, and Anna Maria Fanelli. Meta-data: Characterization of input features for meta-learning. In *MDAI*, pages 457–468, Tsukuba, Japan, July 2005.

[8] R.D. King, C. Feng, and A. Sutherland. Statlog: Comparison of Classification Algorithms on Large Real-Worlds Problems. *Applied Artificial Intelligence*, 9(3):289–333, 1995.

[9] Bernhard Pfahringer, Hilan Bensusan, and Christophe G. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *ICML*, pages 743–750, Stanford University, June 2000.

[10] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM.

[11] Hilan Bensusan and Christophe G. Giraud-Carrier. Discovering task neighbourhoods through landmark learning performances. In *PKDD*, pages 325–330, Lyon, France, September 2000.

[12] Hilan Bensusan and Alexandros Kalousis. Estimating the predictive accuracy of a classifier. In *ECML*, volume 2167, pages 25–36, Freiburg, Germany, September 2001.

[13] Christophe Giraud-Carrier. Metalearning - a tutorial. 2008.

[14] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[15] J. Petrak. The METAL Machine Learning Experimentation Environment V3. 0. 2002.