Ranking with Predictive Clustering Trees

Ljupčo Todorovski[1], Hendrik Blockeel[2], and Sašo Džeroski[1]

[1] Department of Intelligent Systems, Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana, Slovenia
Ljupco.Todorovski@ijs.si, Saso.Dzeroski@ijs.si

[2] Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
Hendrik.Blockeel@cs.kuleuven.ac.be

Abstract. A novel class of applications of predictive clustering trees is addressed, namely ranking. Predictive clustering trees, as implemented in CLUS, allow for predicting multiple target variables. This approach makes sense especially if the target variables are not independent of each other. This is typically the case in ranking, where the (relative) performance of several approaches on the same task has to be predicted from a given description of the task. We propose to use predictive clustering trees for ranking. As compared to existing ranking approaches which are instance-based, our approach also allows for an explanation of the predicted rankings. We illustrate our approach on the task of ranking machine learning algorithms, where the (relative) performance of the learning algorithms on a dataset has to be predicted from a given dataset description.

1 Introduction

In many cases, running an algorithm on a given task can be time consuming, especially when the algorithm is complex and complex tasks are involved. It is therefore desirable to be able to predict the performance of a given algorithm on a given task from a description (set of properties of the task) and without actually running the algorithm. The term "performance of an algorithm" is often used to denote the quality of the solution provided, the running time of the algorithm or some combination of the two.

When several algorithms are available to solve the same type of task, the problem of choosing an appropriate algorithm for the particular task at hand arises. An appropriate algorithm would be an algorithm with a good performance on the given task. Being able to predict the performance of the algorithms, without actually starting them on a given task, will make the problem of choosing easier and less time consuming. We can view performance prediction as a multi-target prediction problem, where the same input (the task description) is used to predict several related targets (the performances of the different algorithms). In this context, it is the relative performance of the different algorithms that matters, and not so much the absolute performance of each of them. We are thus interested in obtaining an ordering of the algorithms (called also ranking) in terms of their expected relative performance.

Within the area of machine learning, many learning algorithms have been developed, especially for classification tasks. A classification task is specified by giving a table of data and indicating the target column: the pair is often referred to as a dataset. The task of predicting the performance of learning algorithms from dataset properties has been addressed within the StatLog project [6], while the task of ranking learning algorithms has been one of the major topics of study of the METAL project [1]. Both are treated as learning problems, where the results of applying selected learning algorithms on selected datasets (base-level learning) constitute a dataset for meta-level learning.

A typical meta-level dataset for ranking thus consists of two parts. The first set of columns (attributes) contains a description of the task at hand. In the case of ranking learning algorithms, it typically contains general and statistical properties of datasets (such as the number of examples and class value and the average kurtosis per numerical attribute). The second set of columns (class values) contains the performance figures for the learning algorithms on the given datasets (e.g., the classification error of C5.0, RIPPER, etc.).

Many different variants of ranking have been studied within the METAL project. A prototypical ranker uses a case-based (nearest neighbor) approach. To produce a ranking of the learning algorithms on a new dataset, the most similar datasets from the meta-level dataset are chosen and the performances (rankings) of the algorithms on these datasets are averaged to obtain a prediction of the performance (ranking) on the new dataset [11]. In an alternative approach to ranking, proposed in [2], regression methods are used to estimate the (absolute) performance of each of the learning algorithms on a given task. These individual predictions are then used to obtain the ranking of the algorithms. In this paper, instead of using regression methods for predicting performances of individual algorithms, we propose the use of predictive clustering trees for ranking. In this case, a single predictive clustering tree has the ability to predict performances of all the learning algorithms at once. Thus, in addition to obtaining a ranking, we also obtain an explanation for it.

The remainder of this paper is organized as follows. Section 2 describes in more detail the task of ranking of learning algorithms. Section 3 briefly describes predictive clustering trees and describes the particular formulation of the multi-target (relative) performance prediction used in our experiments. Section 4 describes the experimental setup and the results of evaluating our approach to ranking learning algorithms. Finally, Section 5 concludes with a summary and possible directions for future work.

2 Ranking of Learning Algorithms

This section describes in more detail the task of ranking of learning algorithms. This includes the machine learning algorithms ranked, the base-level datasets, the descriptions of the datasets, and the performance evaluation methodology.

Table 1. Eight machine learning algorithms for classification tasks used in our study.

Acronym	Brief description
c50tree (`c50t`)	C5.0 - decision trees based classifier
c50rules (`c50r`)	decision rules extracted from a C5.0 tree
c50boost (`c50b`)	boosting C5.0 decision trees
ltree (`lt`)	linear discriminant decision trees
ripper (`rip`)	decision rules based classifier
mlcnb (`nb`)	naive Bayes classifier (MLC++)
mlib1 (`nn`)	1-NN nearest neighbor classifier (MLC++)
lindiscr (`ld`)	linear discriminant classifier

2.1 The machine learning algorithms and datasets

In this study, we analyze the relative performance of eight machine learning algorithms for classification tasks. The same set of classifiers and algorithms has been used in the related study of estimating the predictive performance of individual classifiers [2]. The set of algorithms is presented in Table 1: this is a subset of the set of ten algorithms used within the METAL project [1]. Representatives of different classification approaches are included in this set, such as decision trees, decision rules, naive Bayes, nearest neighbor and linear discriminant classifiers.

Table 2. Sixty-five classification datasets used in our study.

abalone, acetylation, agaricus-lepiota, allbp, allhyper, allhypo, allrep, australian, balance-scale, bands, breast-cancer-wisconsin, breast-cancer-wisconsin_nominal, bupa, car, contraceptive, crx, dermatology, dis, ecoli, flag_language, flag_religion, flare_c, flare_c_er, flare_m, flare_m_er, flare_x, flare_x_er fluid, german_numb, glass, glass2, heart, hepatitis, hypothyroid, ionosphere, iris, kp, led24, led7, lymphography, monk1, monk2, monk3-full, mushrooms, new-thyroid, parity5_5, pima-indians-diabetes, processed.cleveland_2, processed.cleveland_4, processed.hungarian_2, processed.hungarian_4, processed.switzerland_2, processed.switzerland_4, quisclas, sick-euthyroid, soybean-large, tic-tac-toe, titanic, tumor-LOI, vote, vowel, waveform40, wdbc, wpbc, yeast

The performance of these eight algorithms has been measured on a set of sixty-five classification tasks (datasets) from the UCI repository [3] and from the METAL project. The list of datasets is given in Table 2.

2.2 Dataset descriptions

Finding a dataset characterization method that would provide a solid basis for prediction of performance of learning algorithms is probably most important aspect of meta-learning.[1] Several different dataset descriptions have been used for meta-learning.

[1] Note that there is an important constraint on the complexity dataset characterization method. The dataset description should be generated faster than evaluating the

One approach to dataset characterization, proposed within the StatLog project [6], is to use a set of general, statistical and information theory based measures of the dataset. The general properties include properties such as number of examples, classes and (symbolic and numeric) attributes in the dataset. Statistical properties are used to characterize numeric attributes in the dataset and they include measures such as average skewness and kurtosis of numeric attributes. Characteristics of discrete attributes are measured with information theory based measures such as average entropy and average mutual information between discrete attributes and the class.

The StatLog approach gave rise to the development of the Data set Characterizing Tool (DCT) [9] within the METAL project. The set of DCT properties extends the initial set of StatLog properties. Table 3 presents the set of DCT properties used in this study.

Table 3. DCT dataset properties.

DCT		
nr_examples	nr_num_attributes	nr_classes
nr_sym_attributes	missvalues_total	missvalues_relative
lines_with_missvalues_total	lines_with_missvalues_relative	countattr_count_all_value
ndiscrimfunct	fract	cancor
meanskew	meankurtosis	classentropy
entropyattributes	mutualinformation	equivalent_nr_of_attrs
noisesignalratio	avgattr_multicorrel	minattr_multicorrel
maxattr_multicorrel	sdratio	avgattr_gini_sym
minattr_gini_sym	maxattr_gini_sym	avgattr_relevance
minattr_relevance	maxattr_relevance	numattrswithoutliers
avgattr_gfunction	minattr_gfunction	maxattr_gfunction

The DCT properties include also properties of the individual attributes in the dataset, such as kurtosis of each numerical attribute or entropy of each symbolic attribute. These properties cannot be directly used in propositional meta-learning, where the dataset description is a fixed-length vector of properties. In order to use the DCT properties of the individual attributes, we have to aggregate them using average, minimum or maximum function.

Kalousis and Theoharis [8] have proposed an alternative approach to dataset characterization. They use histograms for fine grained aggregation of the DCT properties of the individual attributes. Histograms, used as an aggregation method, preserve more information about the DCT properties of the individual attributes compared to the simple aggregation functions of average, minimum and maximum used in the DCT approach. For detailed description of how aggregations based on histograms are calculated see [8]. In this paper, we used

performance of the learning algorithms on the dataset. Otherwise, the task of meta-level leering would be meaningless. However, analysis of computational complexity of different dataset description approaches is beyond the scope of this paper, it be found in [7].

the same set of histograms as the one used in [2]. This set includes histograms for four DCT properties of individual attributes and twelve DCT properties of the whole dataset. We refer to the histogram approach to dataset description as HISTO.

Finally, in the landmarking approach to dataset description [10], the performances of a set of simple and fast learning algorithms, named landmarkers, are estimated and used as dataset properties. In the original study on using landmarkers for meta-learning, a set of seven landmarkers was proposed. This set includes simple classifiers, such as different versions of a decision node classifier (i.e., a decision tree with a single decision node), naive Bayes, linear discriminant and 1-nearest neighbor. However, three of the landmarkers are already included in the list of classifiers from Table 1 for which we predict the performance. Therefore, in the present study, we use the set of the remaining four landmarkers and we will refer to this approach to dataset description as LAND.

2.3 The performance of a learning algorithm

When building a dataset for meta-learning, we also need an estimate of the performance of the learning algorithms on a given classification task. Most often, the performance of a learning algorithm a on a given classification task d is measured by the predictive error $ERR(a,d)$, i.e., the percentage of incorrectly classified examples. To estimate the predictive error on test examples, unseen during the training of the classifier, a standard ten-fold cross validation method has been used.

2.4 The performance of ranking

The performance of ranking is measured by comparing the ranking predicted by the ranking method with the true ranking of the learning algorithms on a given dataset. We used a standard measure of similarity of two rankings, Spearman's rank correlation coefficient [11]:

$$r_s = 1 - 6\frac{(\sum_{i=1}^{n} {D_i}^2)}{n^3 - n}, \tag{1}$$

where D_i is the difference between actual and predicted rank of the i'th algorithm and n is the number of learning algorithms. Again, to estimate the performance of ranker on test datasets, unseen during the training of the ranker, a standard ten-fold cross validation method has been used.

3 Ranking with Predictive Clustering Trees

This section first briefly describes predictive clustering trees. It then discusses how they could be used to predict the errors of different learning algorithm on a given dataset simultaneously. It finally proposes to use the ranks calculated from the errors as the target variables, rather than the errors themselves.

3.1 Predictive Clustering Trees

Decision trees are most often used in the context of classification or single-target regression; i.e., they represent a model in which the value of a single variable is predicted. However, as a decision tree naturally identifies partitions of the data (course-grained at the top of the tree, fine-grained at the bottom), one can also consider a tree as a hierarchy of clusters. A good cluster hierarchy is one in which individuals that are in the same cluster are also similar with respect to a number of observable properties.

This leads to a simple method for building trees that allow the prediction of multiple target attributes at once. If we can define a distance measure on tuples of target variable values, we can build decision trees for multi-target prediction. The standard TDIDT algorithm can be used: as a heuristic for selecting tests to include in the tree, we use the minimization of intra-cluster variance (and maximization of inter-cluster variance) in the created clustering.

A detailed description of the algorithm can be found in [4]. An implementation is publicly available in the first-order learner Tilde that is included in the ACE tool [5]; however for this paper we have used Clus, a downgrade of Tilde that works only on propositional data.

3.2 Ranking via Predicting Errors

The instance-based approaches to ranking predict rankings of algorithms on a dataset by predicting the errors of the algorithms on the dataset, then creating a ranking from these [11]. An instance here consists of a description of a dataset, plus the performance of eight different algorithms on that dataset. Based on these eight target values, an example can be positioned in an eight-dimensional space.

In its standard mode of operation, Clus builds its trees so that the intra-cluster variance is minimized, where variance is defined as $\sum_{j=1}^{N} d(\mathbf{x}_j, \bar{\mathbf{x}})^2$ where $\bar{\mathbf{x}}$ is the mean vector of the cluster, $\mathbf{x}_j$ is an element of the cluster, N is the number of elements in the cluster, and d represents the euclidean distance. So, what Clus does is trying to create clusters in such a way that a given algorithm will perform similarly on all datasets in that cluster.

Note that this is different from what we want: creating clusters in which several algorithms have the same relative performance. To illustrate this, suppose we have four algorithms which on two datasets score the following errors:

$$\{(0.1, 0.2, 0.3, 0.4), (0.5, 0.6, 0.7, 0.8)\}$$

Clearly the relative performance of the four algorithms is exactly same on the three datasets, so they belong to the same cluster. However, the variance in this cluster is relatively large. Compare this to

$$\{(0.1, 0.2, 0.3, 0.4), (0.4, 0.3, 0.2, 0.1)\}$$

which has a smaller variance than the previous cluster but is clearly worse: the relative performances are opposite.

3.3 Ranking Trees

A solution for this problem is to first rank the algorithms and to predict these ranks instead of the errors themselves. In this way, we obtain ranking trees. A ranking tree has leaves in which a ranking of the performance of different algorithms is predicted.

This transformation removes fluctuations in the variance that are caused by differences in absolute rather than relative performance. Moreover, given the formula for the Spearman's rank correlation coefficient (1), it is clear that a linear relationship between variance and expected Spearman correlation exists. Indeed, note that in the case when the ranks are predicted, the variance $d(\mathbf{x}_i, \bar{\mathbf{x}})^2$ equals $\sum_{i=1}^{n} {D_i}^2$ from the formula (1). This is true under an assumption that the exact ranking number of each algorithm is predicted. This assumption is not fulfilled. Instead of predicting exact ranks, the clustering tree predicts only approximations of rank numbers, e.g.:

$$(6.0, 6.4, 3.65, 6.1, 5.65, 3.5, 5.65, 3.7).$$

Of course, by comparing these approximations we can easily obtain the following exact ranking:

$$(6, 8, 2, 7, 4.5, 1, 4.5, 3)$$

of the eight algorithm. However, the aforementioned equivalence of variance and Spearman's correlation coefficient does not hold anymore. Thus, minimizing intra-cluster variance should be seen as an approximation to maximizing Spearman's correlation coefficient. Note, however, that this approximation is far better than minimizing intra-cluster variance based on the error rates themselves.

4 Experiments

Our experiments investigate the performance of ranking with predictive clustering trees induced using the three different dataset characterization approaches presented in Section 2. Following the discussion from Section 3.3, we transformed the target error values into ranks. The remainder of this section first describes the experimental setup. It then presents the experimental results, including an example ranking tree and performance figures on the correlation between actual and predicted rankings.

4.1 Experimental Setup

Clus was run several times with the same system settings, but on different datasets that vary along two dimensions:

- Language bias: DCT, HISTO, LAND, DEF
- Targets: errors, ranks

The first three language bias settings correspond to the three dataset characterization approaches described in Section 2. DCT uses set of properties of the whole dataset and aggregations of the properties of the individual attributes. HISTO uses more sophisticated aggregation method of histograms for aggregating the properties of the individual attributes. LAND uses estimated performances of four landmarkers for dataset description. Finally, DEF uses no information at all to induce a "default" model tree that consists of a single leaf (i.e., a model that just predicts the average of the performances encountered in the training set). For the first three cases, tests in the constructed tree are always of the form $A < c$, where A is a numerical attribute from one of the DCT, HISTO or LAND datasets (note that all meta-level attributes are numeric) and c some value for it (any value from A's domain was allowed).

The target values were either the errors themselves, which allows us to compare some results directly with [2], or the ranks, which (according to our explanation in Section 3.3) we hope to yield better results w.r.t. the Spearman's rank correlation coefficient.

Our evaluation is based on a ten-fold cross validation, in order to maximize comparability with the results in [2]. Unfortunately we could not use exactly the same cross validation folds.

The CLUS system has a number of parameters with which it can be tuned. One parameter that influences the results quite strongly is the so-called "ftest" parameter. CLUS uses a stopping criterion that is based on a statistical F-test (the standard way to test whether the average intra-cluster variance after a split is significantly smaller than the original variance); the "significance level" at which this test is performed is the "ftest" parameter. Values close to 0 cause CLUS to quickly stop adding nodes to the tree, yielding small trees; the value 1 yields trees of maximal size.

Preliminary experiments with CLUS and TILDE on this and similar datasets indicated that `ftest=1` yielded best results. Therefore we adopted this setting for all the experiments described here. Except for this ftest parameter, the default values were used for all parameters.

4.2 Experimental Results

Table 4 shows the mean absolute deviations of the predicted error rates from the true ones for different learning algorithms. The left-hand side (CLUS - ranks) gives results of clustering trees: these are compared to the results in the right-hand side taken from [2]. We can see that on average, predictive clustering trees score approximately equally good as the Kernel or Cubist methods on DCT and HISTO meta-level datasets. Clustering trees perform worse on the LAND dataset. This is due to the fact that we decided to use a set of landmarkers that are disjoint with the set of target classifiers. In [2] seven landmarkers have been used, three of them being the same as the target classifiers. However, having meta-level attributes (landmarkers) that are the same to the meta-level class to be predicted (target classifiers) makes the task of predicting their performance trivial. Thus, the results on the LAND dataset are hard to compare.

Note, however, that a single predictive clustering tree predicting the performance of all the learning algorithms at once has a very important advantage over the set of eight regression trees for predicting the performance of individual algorithms. A clustering tree provides a single model that can be easily interpreted.

Table 4. Mean absolute deviations (MADs) for a single predictive clustering tree (predicting all the error rates at once) induced with CLUS compared to the MADs of a set of regression trees (one for each learning algorithm) induced with Kernel and Cubist methods. The Kernel and Cubist results are taken from [2]. Note that LAND and LAND* meta-level datasets are different.

	CLUS - errors				Kernel			Cubist		
Classifier	DCT	HISTO	LAND	DEF	DCT	HISTO	LAND*	DCT	HISTO	LAND*
c50boost	0.105	0.114	0.139	0.136	0.112	0.123	0.050	0.103	0.128	0.033
c50rules	0.100	0.110	0.136	0.135	0.110	0.121	0.051	0.121	0.126	0.036
c50tree	0.101	0.109	0.137	0.139	0.110	0.123	0.054	0.114	0.130	0.044
lindiscr	0.119	0.124	0.126	0.139	0.118	0.129	0.063	0.118	0.140	0.054
ltree	0.106	0.107	0.123	0.134	0.105	0.113	0.041	0.114	0.121	0.032
mlcib1	0.120	0.124	0.144	0.155	0.120	0.138	0.081	0.150	0.149	0.067
mlcnb	0.124	0.135	0.145	0.149	0.121	0.143	0.064	0.126	0.149	0.044
ripper	0.135	0.114	0.138	0.147	0.113	0.128	0.056	0.128	0.131	0.041

While the above MAD values are useful to compare our approach with previous approaches, our ultimate criterion is the Spearman's rank correlation coefficient between the predicted ranking and the actual ranking of the methods. Spearman correlations are shown in Table 5.

Table 5. Spearman's rank correlation coefficients (SRCCs) for the predictive clustering trees (predicting error rates and rankings) approach compared to SRCCs of other ranking approaches. Results for Cubist, Kernel and Zooming are taken from [2].

	CLUS		regression trees		
	ranks	errors	Kernel	Cubist	Zooming
DEF	0.372	0.349	0.330	0.330	0.330
DCT	0.399	0.380	0.435	0.083	0.341
HISTO	0.429	0.426	0.405	0.174	0.371
LAND	0.266	0.197	-	*0.090	*0.190

A first observation is that for each meta-level dataset, ranking trees built from ranks score better than ranking trees built directly from error rates. This corresponds with our intuition, explained in Section 3.3. Furthermore, both clustering trees approaches have better scores than all the others, except for the kernel method with the DCT dataset, which has also the highest overall value.

These experimental results provide support for the two effects we identified earlier as possibly positively influencing the results. First, predictive clustering trees capture dependencies between different algorithms better than separate predictive models for each algorithm can. Second, when using intra-cluster variance minimization as a heuristic, it is better to first convert values into ranks.

We conclude this discussion with an example tree. Table 6 shows a predictive clustering tree induced on the DCT dataset with ranks as target values. Each leaf node in the tree predicts a ranking of the eight algorithms from Table 1. For example, first leaf node in the tree (marked with `(*)`) predicts that c50boost (`c50b`) will perform better than c50rules (`c50r`) that will perform better than ltree (`lt`) and so on. The tree indicates that the number of attributes with outliers is most influential for the ranking of the algorithms. It also indicates that the two properties of number of symbolic and numeric attributes in the dataset seem to have good predictive power. Further interpretation and analysis of the tree is possible but it is beyond the scope of this paper.

5 Summary and Further Work

We have used predictive clustering trees to rank (predict the relative performance of) classification algorithms according to the performance on a given dataset using dataset properties. Three different dataset descriptions were used. Two different tasks were considered: predicting actual performances and predicting relative performances (ranking).

On the first task of predicting the performance of classifiers, a single clustering tree predicting performances of all classifiers at once performs as well as a set of regression trees, each of them predicting performances of an individual classifier. However, the important advantage of the clustering trees approach is that it provides a single interpretable model.

On the second task of predicting ranking, the experimental results show that using ranks as target variables in clustering trees works better than using actual performances for all dataset descriptions. Ranking with a single clustering tree performs better than ranking with a set of regression trees for two out of three dataset description approaches. Finally, ranking with clustering trees outperforms also instance-based approach of Zooming.

An immediate direction for further work is to extend our ranking approach to work with relational dataset descriptions, similar to the one presented in [12]. Following the relational approach, properties of individual attributes can be included in the dataset description without being aggregated using mean, maximal and minimal values or histograms. This can be easily done, due to the fact that TILDE allows for relational tests to be used in the nodes of predictive clustering trees by using an appropriate language bias.

Other directions for further work include consideration of additional dataset properties. Dataset properties based on the shape of decision trees induced from a datasets could be interesting in this respect. Finally, the ranking methodology proposed in the paper can be also used and evaluated on other ranking tasks. A possible application would be ranking of the optimization algorithms performance on the basis of the optimization problem description.

Table 6. An example ranking tree (see Table 1 for the legend of the algorithms' acronyms. Note that symbol < in the leaves denotes "performs better than".

```
NumAttrsWithOutliers > 3
+-yes: AVGAttr_gFunction > -1.236
|    +-yes: ClassEntropy > 0.977
|    |    +-yes: ClassEntropy > 0.999
|    |    |    +-yes: c50b<c50r<lt<nn<c50t<rip<ld<nb (*)
|    |    |    +-no: c50b<nn<c50t<c50r<rip<nb<lt<ld
|    |    +-no:  Nr_sym_attributes > 6
|    |         +-yes: Nr_sym_attributes > 10
|    |         |    +-yes: ClassEntropy > 0.445
|    |         |    |    +-yes: c50b<rip<c50t<c50r<lt<nb<nn<ld
|    |         |    |    +-no:  MINAttr_Gini_sym > -0.068
|    |         |    |         +-yes: c50t<c50b<c50r<lt<rip<nn<ld<nb
|    |         |    |         +-no: c50r<c50t<c50b<rip<lt<nn<ld<nb
|    |         |    +-no: c50r<c50t<lt<rip<c50b<nn<ld<nb
|    |         +-no: lt<c50b<ld<c50t<c50r<nn<rip<nb
|    +-no:  Nr_num_attributes > 0
|         +-yes: Nr_sym_attributes > 4
|         |    +-yes: c50b<c50r<c50t<rip<lt<nb<ld<nn
|         |    +-no: c50b<lt<c50r<c50t<nn<rip<ld<nb
|         +-no: c50b<ld<nb<lt<c50t<c50r<nn<rip
+-no:  MAXAttr_gFunction > -1.064
     +-yes: Nr_examples > 303
     |    +-yes: Nr_num_attributes > 0
     |    |    +-yes: SDRatio > 1.085
     |    |    |    +-yes: c50b<c50r<lt<c50t<ld<nn<rip<nb
     |    |    |    +-no: ld<c50b<lt<c50r<nb<c50t<rip<nn
     |    |    +-no:  Nr_examples > 1,728
     |    |         +-yes: c50b<rip<c50t<nn<nb<lt<c50r<ld
     |    |         +-no:  ClassEntropy > 0.914
     |    |              +-yes: Nr_sym_attributes > 9
     |    |              |    +-yes: c50t<c50r<lt<c50b<rip<nn<ld<nb
     |    |              |    +-no: c50b<c50r<nn<c50t<lt<rip<nb<ld
     |    |              +-no: c50r<c50t<lt<rip<ld<c50b<nn<nb
     |    +-no:  Nr_classes > 3
     |         +-yes: c50r<nb<c50t<lt<rip<c50b<nn<ld
     |         +-no:  Nr_examples > 215
     |              +-yes: ld<nb<lt<c50r<c50b<rip<nn<c50t
     |              +-no: lt<nb<rip<c50r<c50b<nn<ld<c50t
     +-no:  Equivalent_nr_of_attrs > 9.738
          +-yes: MeanKurtosis > 2.891
          |    +-yes: Nr_examples > 303
          |    |    +-yes: Nr_classes > 6
          |    |    |    +-yes: ld<nb<lt<c50b<c50t<c50r<nn<rip
          |    |    |    +-no: ld<lt<c50b<c50r<c50t<nb<nn<rip
          |    |    +-no: lt<ld<nb<c50b<rip<nn<c50r<c50t
          |    +-no: c50b<lt<ld<c50r<nb<c50t<rip<nn
          +-no:  Nr_classes > 3
               +-yes: c50b<nn<c50r<c50t<nb<ld<lt<rip
               +-no: lt<ld<c50b<c50r<nb<rip<nn<c50t
```

Acknowledgments

This work was supported in part by the METAL project (ESPRIT Framework IV LTR Grant Nr 26.357). Hendrik Blockeel is a post-doctoral fellow of the Fund for Scientific Research (FWO) of Flanders. The CLUS system was implemented by Jan Struyf. Thanks to Alexandros Kalousis for providing the meta-level data.

References

1. *ESPRIT METAL Project (project number 26.357)*: A Meta-Learning Assistant for Providing User Support in Machine Learning and Data Mining. `http://www.metal-kdd.org/`.
2. H. Bensusan and A. Kalousis. Estimating the predictive accuracy of a classifier. In *Proc. of the Twelfth European Conference on Machine Learning*, pages 25–36. Springer, Berlin, 2001.
3. C. L. Blake and C. J. Merz. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
4. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proc. of the Fifteenth International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann, 1998.
5. H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. Ramon, and H. Vandecasteele. Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 2002. In press.
6. P. B. Brazdil and R. J. Henery. Analysis of results. In D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors, *Machine learning, neural and statistical classification*, pages 98–106. Ellis Horwood, Chichester, 1994.
7. A. Kalousis. Algorithm Selection via Meta-Learning. PhD Thesis. University of Geneva, Department of Computer Science, 2002.
8. A. Kalousis and T. Theoharis. NEOMON: design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis* 3(5): 319–337, 1999.
9. G. Lindner and R. Studer. AST: Support for algorithm selection with a CBR approach. In *Proc. of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work*, pages 38–47. J. Stefan Institute, Ljubljana, Slovenia, 1999.
10. B. Pfahringer, H. Bensusan and C. Giraud-Carrier. Meta-Learning by Landmarking Various Learning Algorithms. In *Proc. of the Seventeenth International Conference on Machine Learning*: 743–750. Morgan Kaufmann, San Francisco, 2000.
11. C. Soares and P. B. Brazdil. Zoomed ranking: Selection of classification algorithms based on relevant performance information. In *Proc. of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 126–135. Springer, Berlin, 2000.
12. L. Todorovski and S. Džeroski. Experiments in meta-level learning with ILP. In *Proc. of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pages 98–106. Springer, Berlin, 1999.