Relational Ranking with Predictive Clustering Trees

Sašo Džeroski and Ljupčo Todorovski
Department of Intelligent Systems
Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana, Slovenia

Hendrik Blockeel
Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium

Abstract

A novel class of applications of predictive clustering trees is addressed, namely relational ranking. Predictive clustering trees, as implemented in TILDE, allow for predicting multiple target variables from relational data. 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. This allows us to use relational descriptions of the tasks. 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 algorithms on a given dataset has to be predicted from a given (relational) dataset description.

1. Introduction

In many cases, running an algorithm on a given task can be time consuming, especially when 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.

As an example, consider the task of optimization, e.g., finding the minimum value of a function. Given an optimization algorithm (say Levenberg-Marquardt), we might be interested in predicting the quality of the solution found (e.g., how close to the real optimum was the solution) and/or the running time of the algorithm. A description of the task would be a description of the function to be optimized (e.g., in terms of the number of different trigonometric and algebraic operators appearing in it, the size of the tree needed to encode the function, etc.).

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. We can view this 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 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 [5], 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) contain 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 values and the average kurtosis per numerical attribute). The second set of columns 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 rank-

ing on the learning algorithms a new dataset, the most similar datasets from the meta-level dataset are chosen and the performances (rankings) of the algorithms on theses datasets are averaged to obtain a prediction of the performance (ranking) on the new dataset.

In this paper, we propose to use a relational representation of tasks (datasets) in ranking instead of a propositional one. This allows us to represent the tasks in more detail, e.g., include the kurtosis values for each numerical attribute rather than include only the average kurtosis per numerical attribute. We also propose to use predictive clustering trees for ranking instead of case-based approaches. In this case, in addition to obtaining a ranking, we also obtain an explanation.

The remainder of this paper is organized as follows. Section 2 describes in more detail the task of relational ranking of learning algorithms. This includes the base-level datasets, the algorithms ranked, the performance evaluation methodology, and finally the propositional and relational descriptions of datasets. 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.

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

Acronym	Brief description
c50tree	C5.0 - decision trees based classifier
c50rules	decision rules extracted from a C5.0 tree
c50boost	boosting C5.0 decision trees
ltree	linear discriminant decision trees
ripper	decision rules based classifier
mlcnb	naive Bayes classifier (MLC++)
mlcib1	1-NN nearest neighbor classifier (MLC++)
lindiscr	linear discriminant classifier
clemMLP	multilayer perceptron ANN (Clementine)
clemRBFN	radial-basis functions ANN (Clementine)

2. Relational Ranking of Learning Algorithms

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

2.1 The machine learning algorithms

In this study, we analyze the relative performance of ten machine learning algorithms for classification tasks. The list of algorithms is presented in Table 1: these are the ten algorithms used within the METAL project [1]. This set includes one or more representatives of different classification approaches, such as decision trees and rules, naive Bayes, nearest neighbor and linear discriminant classifiers, as well as neural networks.

Table 2. Forty-two classification datasets used in our study.

abalone	adult	allbp
allhyper	allhypo	allrep
ann	byzantine	c_class_flares
car	contraceptive	dis
dna_splice	fluid	german_numb
german_symb	krkopt	letter
m_class_flares	mushrooms	musk
nettalk	nursery	optical
page	pendigits	pyrimidines
quadruped	quisclas	segment
shuttle	sick	sick_euthyroid
splice	taska_part_hhold	taska_part_related
taskb_hhold	triazines	waveform21
waveform40	x_class_flares	yeast

2.2 The datasets

The performance of these ten algorithms have been measured on a set of forty-two classification tasks (datasets) used within the METAL project.[1] The list of datasets is given in Table 2. Some of these come from the UCI Repository of Machine Learning Datasets, while others are proprietary.

2.3 Dataset descriptions

Each classification task from Table 2 is described using a set of task properties. In the StatLog project, a set of general, statistical and information theory based dataset properties has been used [5] for dataset description. This gave rise to the Data set Characterizing Tool (DCT) [7], developed further within the METAL project [8], that extends

[1] Fifty-three classification tasks are considered within the METAL meta-level learning studies. However, in our study we have used only a subset of forty-two classification tasks, where the meta-level data (both properties and performance measures) were available. We will include the whole set of METAL classification tasks in our study, as soon as meta-level data become available.

Table 3. Data set properties.

Whole dataset	
num_of_attrs	num_of_sym_attrs
num_of_num_attrs	num_of_examples
num_of_classes	missing_values
lines_with_missing_values	mean_skewness
mean_kurtosis	num_of_attrs_with_outliers
M_stat	M_stat_DF
M_stat_ChiSq	M_stat_ChiSq_alpha
SD_ratio	fract
cancor	wilks_lambda
Bartlett_stat	Bartlett_stat_DF
Bartlett_stat_ChiSq	Bartlett_stat_ChiSq_alpha
class_entropy	entropy_attributes
joint_entropy	equivalent_num_of_attrs
noise_signal_ratio	perc_sym_attrs
perc_num_attrs	examples_per_attr
classes_per_attr	rel_num_of_attrs_with_outliers
rel_equivalent_num_of_attrs	log_num_of_examples

Per attribute	Aggregates
perc_missing_values	AVG MIN MAX
skewness	AVG MIN MAX
kurtosis	AVG MIN MAX
multi_correl	AVG MIN MAX
gini_index	AVG MIN MAX
relevance	AVG MIN MAX
g_function	AVG MIN MAX
class_freq	MAX

the set of StatLog properties. We included most of the DCT properties in the dataset descriptions used in this study. The complete list of dataset properties is presented in Table 3.

There are two groups of DCT properties. The first group contains properties of the entire dataset (first column in Table 3), while the second group contains properties of individual attributes in the dataset (second column in Table 3). In addition, the probability distribution of the class is also included.

The general DCT properties include simple facts about the dataset, such as number of examples, (nominal and numeric) attributes and class values, but also more complicated statistical and information theory based measures of the whole dataset. Furthermore, six measures are used to characterize individual attributes. Three of them are statistical measures for numerical attributes and three of them are information theory based measures for discrete attributes.

Properties of the individual attributes can not be used directly in propositional meta-learning, where the dataset description is a fixed-length vector of dataset properties. For this purpose, each property of the individual attributes is aggregated using the average, minimum or maximum function. The relational framework for meta-learning allows for a more complex representation of data sets [11]. In this study, we include all the DCT properties from Table 3, both global properties of the entire dataset (the general and aggregated ones) and properties of individual attributes.

2.4 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 accuracy $ACC(a, d)$, i.e., the percentage of correctly classified examples. To estimate this predictive accuracy on test examples, unseen during the training of the classifier, a standard ten-fold cross validation method has been used. Another performance measure of a learning algorithm a is its running time $T(a, d)$ on a given dataset d. A third performance measure that combines the predictive accuracy with the running time of a machine learning algorithm named "*adjusted ratio of ratios*" has been proposed in [10]:

$$ARR(a_p, d) = \sum_{a_q \in A, a_q \neq a_p} ARR(a_p, a_q, d),$$

$$ARR(a_p, a_q, d) = \frac{\frac{ACC(a_p,d)}{ACC(a_q,d)}}{1 + \frac{\log\left(\frac{T(a_p,d)}{T(a_q,d)}\right)}{K_T}}$$

where A is the set of learning algorithms under study, and K_T is a user-defined value that determines the relative importance of the running time. The K_T parameter is approximated by $K_T = 1/X\%$, where X is the accuracy one is willing to trade for a 10 times speedup or slowdown. However, due to the lack of the data about running time of the algorithms, we used the ARR measure with the setting $K_T = \inf$ which eliminates the influence of time.

3. Relational Ranking with Predictive Clustering Trees

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

3.1 Predictive Clustering Trees

A variety of algorithms for predictive modeling exists. Among the better known are algorithms that induce decision trees [6, 9]. Compared to other well-known techniques

such as neural networks [2], decision trees have the advantage of being more interpretable: they clearly explicitate the factors that influence the outcome most strongly.

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 (called TIC) can be found in [3]. We used the implementation of TIC as available in the first-order learner TILDE that is included in the ACE tool [4]. This implementation allows for relational tests to be used in the nodes of predictive clustering trees through the use of declarative bias.

3.2 Ranking via Predicting Errors

The instance-based approaches to ranking predict rankings of algorithms on a dataset by predicting the accuracies of the algorithms on the dataset, then creating a ranking from these. An instance here consists of a description of a dataset, plus the performance of 10 different algorithms on that dataset (this performance can be measured as accuracies or ARR values). Based on these 10 target values, an example can be positioned in a 10-dimensional space.

In its standard mode of operation, TILDE 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 TILDE 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 4 algorithms which on 2 datasets score the following accuracies:

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

Clearly the relative performance of the 4 algorithms is exactly the 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 accuracies 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 correlation:

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

where D_i is the difference between actual and predicted rank of the i'th algorithm and n is the number of learning algorithms, it is clear that a linear relationship between variance and expected Spearman correlation exists. Indeed, note that

$$d(\mathbf{x}, \bar{\mathbf{x}})^2 = \sum_{i=1}^{n} {D_i}^2$$

on the condition that the "predicted rank" in each leaf of the tree is indeed the number found for the algorithm.

The latter condition is a problem. The predictive clustering tree might predict, in a specific case,

$$(6.7, 6.0, 6.4, 3.65, 6.1, 5.65, 3.5, 5.65, 3.7, 7.65)$$

If we would use these numbers as predictions, minimizing intra-cluster variance would be equivalent to maximizing expected correlation. However, if we rank algorithms based on these numbers, i.e. use the ranks of the numbers

$$(9, 6, 8, 2, 7, 4.5, 1, 4.5, 3, 10)$$

instead of the original numbers, then the equivalence does not hold anymore, and minimizing intra-cluster variance should be seen as an approximation to maximizing Spearman correlation.

4. Experiments

Our experiments investigate the performance of relational ranking with predictive clustering trees on the dataset

described in Section 2. Following the discussion from Section 3.3, we transformed the target accuracies and ARR values into ranks. Doing this, we noticed that both performance measures result into the same ranking of the learning algorithms, thus from now on, we do not make distinction between these two performance measures. The remainder of this section first describes the experimental setup, and in particular the language biases used, and the pruning performed. 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

The TILDE system has a number of settings that influence its behavior. We have performed several experiments, using default values for all settings except the following settings, which were varied (an explanation follows):

- Language bias: None, Prop, Rel, Both
- Ftest settings: 0.001, 0.005, 0.01, 0.05, 0.1, 1.0

The four language bias settings correspond to using only propositional information (i.e., properties of the whole dataset and aggregations of the properties of the individual attributes/classes), only relational information (i.e, properties of individual attributes/classes only), both, or no information at all. The latter bias was included to measure the performance of "default" models that consists of just a leaf (these predict the average of the values encountered in the training set). For the propositional data, the language bias consists of tests $A < c$ with A a meta-attribute (all meta-attributes are numeric) and c some value for it (any value from A's domain was allowed). The number of A, c combinations (i.e., the number of possible tests that can be constructed at a single node) is over 1400.

For the relational data, a language was constructed that essentially allows to check properties of an individual attribute of a dataset, e.g., check if the skewness of some numeric attribute of that dataset is above a certain value. Note that checking for the existence of an attribute with skewness > 1 (for instance) is equivalent to checking whether the maximum of all skewness values is > 1. As such maxima (and minima) are included in the propositional descriptions of the datasets, this in itself does not yield more expressiveness. With the relational version it is however also possible to check for the existence of a single attribute in a dataset that has several properties, e.g., "is there an attribute with skewness > 1 and kurtosis < 0"; this kind of tests cannot be constructed in our propositional representation. With the relational representation, the number of tests considered at a specific node of the tree varies from 566 to over 1000.

The Ftest setting in TILDE is a stopping heuristic based on the classical statistical F-test. The values indicate significance levels; lower values cause the tree to be smaller. We have not exhaustively searched the space of all possible parameter settings, but instead explored it more or less intuitively, in a kind of hill-climbing fashion. More specifically, we first performed the following experiment: "With language Rel and varying Ftest from 0.001 to 1.0, estimate the performance of ranking trees using leave-one-out". The results suggested that best performance is obtained at high Ftest values, i.e., with the least-pruned trees.

Table 4. Performances of PCTs induced using three different biases (propositional, relational and both) compared to the default performance (of a single-node PCT).

	$F = 0.05$		$F = 0.1$		$F = 1.0$	
	RE	r_s	*RE*	r_s	*RE*	r_s
propositional	1.16	0.46	1.13	0.49	1.20	0.46
relational	1.06	0.47	0.94	0.49	0.87	0.53
both	1.12	0.47	1.10	0.49	1.16	0.48
default	1.0	0.51	1.0	0.51	1.0	0.51

4.2 Experimental Results

We next performed a second experiment: "For all language biases, and for large Ftest values (0.05, 0.1, 1.0), estimate the performance of ranking trees using leave-one-out." The results are reported in Table 4. RE is the relative error as estimated by TILDE using leave-one-out. r_s is the average Spearman rank correlation between the predicted ranking and the actual ranking of a left-out instance. The table makes clear that the best results are obtained for relational data with an F-test value of 1.0: here the RE is lowest, and the r_s is highest.

There are a few interesting observations to make.

- Results in general are not very good, with most RE's over 1.0 (i.e. worse than default prediction, squared-error-wise) and, unsurprisingly in this light, most r_s below the default r_s of 0.51. It is somewhat strange that a learner would almost consistently construct theories worse than default; we suspect that the very small datasets and the pessimistic bias of cross-validation causes these results to look somewhat worse than they really are.
- The best result is obtained for the relational representation. A closer look at the induced trees reveals that

Table 5. An example ranking tree. In each leaf node the ranking of ten machinge larning algorithms is predicted. The following labels are used to dentote learning algorithms (See also Table 1): c5t = c50tree, c5r = c50rules, c5b = c50boost, lt = ltree, rip = ripper, nb = mlcnb, ib = mlcib1, ld = lindiscr, mlp = clemMLP, rbfn = clemRBFN. The '<' sign is used to dentoe the relation 'performs worse than'.

```
err_ranks(A,B,C,D,E,F,G,H,I,J,K)
classvalue_freq(A,L,M),M < 0.165 ?
+--yes:attr_skew_all(A,N,O),O>3.64 ?
|     +--yes:classvalue_freq(A,P,Q),Q>0.318 ?
|     |     +--yes:classvalue_freq(A,L,R),R < 0.097 ?
|     |     |     +--yes:attr_gfunction(A,S,T),T> -0.437 ?
|     |     |     |     +--yes:attr_relevance(A,S,U),safe(U>0.235) ?
|     |     |     |     |     +--yes:ld < nb < rbfn < ib < mlp < rip < lt < c5r < c5b < c5t
|     |     |     |     |     +--no: nb < ld < rbfn < mlp < lt < ib < rip < c5r < c5t < c5b
|     |     |     |     +--no: classvalue_freq(A,L,V),V < 0.003 ?
|     |     |     |            +--yes:rbfn < c5b < mlp < ld < nb < lt < ib < c5r < c5t < rip
|     |     |     |            +--no: rbfn < c5b < ld < nb < mlp < ib < rip < lt < c5t < c5r
|     |     |     +--no: mlp < rbfn < nb < ld < ib < rip < c5b < c5r < c5t < lt
|     |     +--no: attr_skew_all(A,N,W),W < 5.217 ?
|     |            +--yes:mlp < c5b < nb < rip < rbfn < c5t < ld < c5r < lt < ib
|     |            +--no: nb < ib < rbfn < mlp < c5t < lt < ld < c5r < rip < c5b
|     +--no: classvalue_freq(A,X,Y),Y>0.786 ?
|            +--yes:nb < ld < ib < c5b < rip < lt < mlp < c5r < c5t < rbfn
|            +--no: attr_skew_all(A,Z,A1),A1 <  -0.612 ?
|                   +--yes:ld < rip < c5t < nb < c5r < rbfn < lt < mlp < c5b < ib
|                   +--no: attr_relevance(A,B1,C1),safe(C1 < 0.062) ?
|                          +--yes:rbfn < ld < nb < rip < lt < c5t < mlp < c5r < ib < c5b
|                          +--no: c5b < ld < rbfn < nb < mlp < rip < lt < c5r < c5t < ib
+--no: attr_skew_all(A,D1,E1),E1>2.095 ?
      +--yes:attr_skew_all(A,F1,G1),G1 <  -0.26 ?
      |     +--yes:rbfn < mlp < nb < rip < ld < ib < c5b < c5r < c5t < lt
      |     +--no: nb < rbfn < ld < ib < c5t < lt < rip < c5r < mlp < c5b
      +--no: attr_skew_all(A,H1,I1),I1> -1.303 ?
             +--yes:attr_kurt_all(A,H1,J1),safe(J1 < 1.631) ?
             |     +--yes:ib < c5t < mlp < c5r < rip < rbfn < lt < c5b < nb < ld
             |     +--no: ib < c5t < rip < c5r < nb < c5b < rbfn < lt < ld < mlp
             +--no: mlp < rbfn < ib < ld < lt < c5r < c5t < rip < nb < c5b
```

the essentially relational aspects of the representation are used, but not very often, i.e., most of the trees could in principle also be found from propositional data. Our explanation for this is that the large number of attributes in the propositional descriptions confuses the tree learner. Somewhat unintuitively, the relational bias is actually stronger (there are fewer splits possible): consequently, the relational descriptions are more concise and can focus on more relevant properties.

We have taken a closer look at the tree in Figure 5, derived from the entire dataset with the optimal settings. The tree suggests, for instance, that the most important property of a dataset with respect to the behavior of learning algorithms is whether that dataset contains an infrequent class ("infrequent" defined here as having a frequency below 0.165). Also, the skewness of attribute distributions is identified as highly relevant. Finally, the tree uses relational information, for example, one leaf includes the condition: "there is an attribute with skewness above -1.303 and kurtosis below 1.631."

5. Summary and Further Work

We have used predictive clustering trees to rank (predict the relative performance of) machine learning algorithms for classification. A relational description of datasets is used, which allows to specify dataset properties in more detail: for example, properties of individual attributes can be used rather than bulk properties averaged across all attributes of a dataset. The relational ranking trees perform

better than propositional ranking trees and also better than the default ranking when a smaller amount of tree pruning is applied. As compared to existing ranking approaches which are propositional and instance-based, our approach also allows for an explanation of the predicted rankings.

An immediate direction for further work is to repeat the experimental evaluation on the full METAL ranking dataset (53 meta-level data points) once it becomes available. Given the size of the meta-level dataset, any additional point matters. This would also allow for a direct comparison to the propositional instance-based approaches to ranking.

Other directions for further work include the definition of a relational distance measure on datasets and the use of relational instance-based learning for relational ranking. Investigating the use of kernels for relational data would also be an interesting direction to pursue. Both approaches work well for small datasets of high dimensionality.

If we can deal with small datasets of high dimensionality, it makes sense to also consider additional dataset properties. Dataset properties based on landmarking have been shown to predict performance well and to be useful for ranking. Also, features based on the shape of decision trees induced from a datasets could be interesting in this respect.

Finally, the relational ranking methodology proposed in the paper can be also used and evaluated on other ranking tasks. A possible application would be the ranking of optimization algorithms on the basis of descriptions of optimization problems.

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] C. M. Bishop. *Neural Networks for Pattern Recognition*. University Press, Oxford, 1999.

[3] H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63, 1998. `http://www.cs.kuleuven.ac.be/~ml/PS/ML98-56.ps`.

[4] 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*, 16: 135–166, 2002.

[5] 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, 1994.

[6] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.

[7] R. Engels and C. Theusinger. Using a Data Metric for Offering Preprocessing Advice in Data Mining Applications. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, pages 430–434, 1998.

[8] G. Lindner and R. Studer. Ast: Support for algorithm selection with a cbr approach. In *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work*, pages 38–47, 1999.

[9] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in machine learning. Morgan Kaufmann, 1993.

[10] C. Soares and P. B. Brazdil. Zoomed ranking: Selection of classification algorithms based on relevant performance information. In *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 126–135. Springer, 2000.

[11] L. Todorovski and S. Džeroski. Experiments in meta-level learning with ilp. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pages 98–106. Springer, 1999.