Ensemble of Decision Rules for Solving Binary Classification Problems in the Presence of Missing Values

Jerzy Błaszczyński[1], Krzysztof Dembczyński[1], Wojciech Kotłowski[1], Roman Słowiński[1,2], and Marcin Szeląg[1]

[1] Institute of Computing Science, Poznań University of Technology, 60-965 Poznań, Poland
{kdembczynski, jblaszczynski, wkotlowski, rslowinski, mszelag}@cs.put.poznan.pl
[2] Institute for Systems Research, Polish Academy of Sciences, 01-447 Warsaw, Poland

Abstract. In this paper, we consider an algorithm that generates an ensemble of decision rules. A single rule is treated as a specific subsidiary, base classifier in the ensemble that indicates only one of the decision classes. Experimental results have shown that the ensemble of decision rules is as efficient as other machine learning methods. In this paper we concentrate on a common problem appearing in real-life data that is a presence of missing attributes values. To deal with this problem, we experimented with different approaches inspired by rough set approach to knowledge discovery. Results of those experiments are presented and discussed in the paper.

1 Introduction

Decision rule is a logical expression in the form: *if* [*conditions*], *then* [*decision*]. If an object satisfies conditions of the rule, then it is assigned to the recommended class. Otherwise the object remains unassigned. Decision rules were common in the early machine learning approaches [1, 6], widely considered in the rough set approaches to knowledge discovery (see for example, [20, 15, 23, 24]), and within Logical Analysis of Data [3] where they are called patterns. The algorithm described here follows a specific approach to decision rule generation. It treats a single rule as a subsidiary, base classifier in the ensemble that indicates only one of the decision classes.

Ensemble methods became a very popular approach to classification problems. These methods consist in forming an ensemble of classifiers that are simple learning and classification procedures often referred to as base (or weak) learners. The ensemble members (i.e., base learners or classifiers) are applied to a classification task and their individual outputs are then aggregated to one output of the whole ensemble. The aggregation is computed as a linear combination of outputs. The most popular methods that are used as base learners are decision trees, for example C4.5 [21] or CART [5], and decision stumps (that are

one level decision trees). There are several approaches to construction of the ensemble, the most popular are bagging [4] and boosting [22, 9]. These algorithms have proven to be effective in reducing classification error of a base learner. In other words, a committee of low performance learners creates a powerful and quite simple solution for the classification problem. That is why these methods are often treated as off-the-shelf methods-of-choice.

The ensemble of decision rules, in our approach, is constructed using a variation of forward stagewise additive modeling [10]. Similar technique is also used by Friedman and Popescu [13]. However, one can observe substantial differences between their algorithm and the one presented in this paper. In Friedman and Popescu's algorithm, the decision trees are used as the base classifiers, and then each node (interior and terminal) of each resulting tree produces a rule. It is given by the conjuction of conditions associated with all of the edges on the path from the root to that node. Rule ensemble is then fitted by gradient directed regularization [12]. The algorithm presented here generates directly a single rule, in each iteration of forward stagewise additive modeling. This simpler way is as efficient as other main machine learning methods [2]. Usually, it is enough to generate around 50 rules to achieve satisfying accuracy and the rules, moreover, are easy in interpretation. Our algorithm is also similar to SLIPPER introduced by Cohen and Singer [7]. The difference is that SLIPPER uses AdaBoost [22] schema to produce an ensemble of decision rules. Let us notice that AdaBoost is a specific case of the forward stagewise additive modeling, so the latter is more general approach [10].

In this paper, we concentrate on a common problem of data analysis. The real-life data has often missing attributes values. There are several approaches to deal with this problem. One might discard objects having missing values, but this could lead to serious depletion of the training data. Another possibility is to impute (i.e., fill in) missing values, with the mean, median or mode over nonmissing values of objects on a given attribute. The approach taken here is inspired by rough set theory.

The problem of missing values were studied in many places within rough set theory (see, for example [14, 16–18]). According to Kryszkiewicz [18], generated rules should remain true when all or some missing values will be replaced by arbitrary values. Such an approach is compatible with knowledge discovery from incomplete information systems. In fact, considered here classification problem is quite different. We tried to adapt this approach to the ensemble of decision rules. In this case, in classification procedure, it is reasonable that only universal selector on a given attribute covers objects with missing values on this attribute. It is so, because, the true value of the object on the attribute is unknown, and we are certain that only universal selector will cover it. The goal of this paper is to compare the approach mentioned-above with other approaches to missing values.

The paper is organized as follows. In Section 2, the problem is formulated. Section 3 presents the algorithm for construction of an ensemble of decision rules. In Section 4 approaches taken to deal with problem of missing values are

presented. Section 5 contains experimental results. The last section concludes the paper.

2 Problem Statement

Let us define the classification problem in a similar way as in [11, 13]. The aim is to predict the unknown value of an attribute y (sometimes called *output*, *response variable* or *decision attribute*) of an object using the known joint values of other attributes (sometimes called *predictors*, *condition attributes* or *independent variables*) $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where some of values may be missing. We consider binary classification problem, and we assume that $y \in \{-1, 1\}$. In other words, all objects for which $y = -1$ constitute decision class Cl_{-1}, and all object for which $y = 1$ constitute decision class Cl_1. The goal of a learning task is to find a function $F(\mathbf{x})$ using a set of training examples $\{y_i, \mathbf{x}_i\}_1^N$ that classifies accurately objects to these classes. The optimal classification procedure is given by:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y\mathbf{x}} L(y, F(\mathbf{x}))$$

where the expected value $E_{y\mathbf{x}}$ is over joint distribution of all variables $(y, \mathbf{x})$ for the data to be predicted. $L(y, F(\mathbf{x}))$ is a loss or cost for predicting $F(\mathbf{x})$ when the actual value is y. The typical loss in classification tasks is:

$$L(y, F(\mathbf{x})) = \begin{cases} 0 & y = F(\mathbf{x}), \\ 1 & y \neq F(\mathbf{x}). \end{cases} \tag{1}$$

The learning procedure tries to construct $F(\mathbf{x})$ to be the best possible approximation of $F^*(\mathbf{x})$.

3 Ensembles of Decision Rules

Decision rule is a logical expression in the form: if [*conditions*], *then* [*decision*]. [*conditions*] are represented by a complex $\Phi = \phi_1^{\propto} \wedge \phi_2^{\propto} \wedge \ldots \wedge \phi_t^{\propto}$, where $\phi^{\propto}$ is a selector and t is a number of selectors in the complex, also referred to as a length of the rule. Selector $\phi^{\propto}$ is defined as $x_j \propto v_j$, where v_j is a single value or a subset of values from the domain of the j-th attribute; and $\propto$ is specified as $=, \in, \geq$ or $\leq$, depending on the characteristic of the j-attribute. In other words, complex Φ is a set of selectors that allows to select a subset of objects. Objects covered by complex Φ are denoted by $cov(\Phi)$ and referred to as cover of a complex Φ. [*decision*] indicates one of the decision classes and is denoted by $d(\mathbf{x}) = -1$ or $d(\mathbf{x}) = 1$. Let as denote a rule by $r(\mathbf{x}, \mathbf{c})$, where $\mathbf{c}$ represents complex and decision of the rule, $\mathbf{c} = (\Phi, \quad d(\mathbf{x}))$. Then, the output of the rule may be defined as follows:

$$r(\mathbf{x}, \mathbf{c}) = \begin{cases} d(\mathbf{x}) & \mathbf{x} \in cov(\Phi), \\ 0 & \mathbf{x} \notin cov(\Phi). \end{cases} \tag{2}$$

Algorithm 1: Ensemble of decision rules

input : set of training examples $(y_i, \mathbf{x}_i)$, $i = 1, ..., N$,
M – number of decision rules to be generated.
output: ensemble of decision rules $\{r_m(\mathbf{x})\}_1^M$.
$F_0(\mathbf{x}) := \arg\min_{\alpha \in \{-1,1\}} \sum_i^N L(y_i, \alpha)$; or $F_0(\mathbf{x}) := 0$; //default rule
$F_0(\mathbf{x}) := \nu \cdot F_0(\mathbf{x})$;
for $m = 1$ *to* M **do**
  $\mathbf{c} := \arg\min_{\mathbf{c}} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i, \mathbf{c}))$;
  $r_m(\mathbf{x}) = r(\mathbf{x}, \mathbf{c})$;
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot r_m(\mathbf{x})$;
end
$ensemble = \{r_m(\mathbf{x})\}_1^M$;

The loss of a single decision rule takes a specific form:

$$L(y, r(\mathbf{x}, \mathbf{c})) = \begin{cases} 0 & y \cdot r(\mathbf{x}, \mathbf{c}) = 1, \\ 1 & y \cdot r(\mathbf{x}, \mathbf{c}) = -1, \\ l & r(\mathbf{x}, \mathbf{c}) = 0, \end{cases} \tag{3}$$

where $0 \geq l \geq 1$ is a penalty for specificity of the rule. It means, the lower the value of l then the smaller the number of objects covered by the rule from the opposite class.

Forward stagewise additive modeling [10] is a general schema that constructs an ensemble. This schema may be suited to the problem of decision rules generation. We have used a variation of it that was also applied by Friedman and Popescu [13]. In their approach, however, base classifiers are decision trees, from which decision rules are produced. Here, the rule is generated directly in each step of Algorithm 1. In this procedure, $L(y_i, F(\mathbf{x}))$ is a loss function, $r_m(\mathbf{x}, \mathbf{c})$ is a decision rule characterized by a set of parameters $\mathbf{c}$ and M is a number of rules to be generated. $S_m(\eta)$ represents a different subsample of size $\eta \leq N$ randomly drawn with or without replacement from the original training data. ν is so called "shrinkage" parameter, usually $0 \leq \nu \leq 1$. Values of ν determine the degree to which previously generated decision rules $r_k(\mathbf{x}, \mathbf{p})$, $k = 1, \ldots, m$, affect the generation of a successive one in the sequence, i.e., $r_{m+1}(\mathbf{x}, \mathbf{p})$.

In the algorithm, in each consecutive iteration m we augment the function $F_{m-1}(\mathbf{x})$ by one additional rule $r_m(\mathbf{x})$ weighted by shrinkage parameter ν. This gives a linear combinations of rules $F_m(\mathbf{x})$. The additional rule $r_m(\mathbf{x}) = r(\mathbf{x}, \mathbf{c})$ is chosen to minimize $\sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i, \mathbf{c}))$. $F_0(\mathbf{x})$ corresponds to the default rule in the ensemble generation process. It is set to $F_0(\mathbf{x}) := \arg\min_{\alpha \in \{-1,1\}} \sum_i^N L(y_i, \alpha)$ (i.e., it corresponds to the default rule indicating the majority class) or there is no default rule (then $F_0(\mathbf{x}) = 0$). The default rule is taken with the same "shrinkage" parameter ν as all other rules.

The loss of the linear combination of rules $F_m(\mathbf{x})$ takes the following form in the simplest case:

$$L(y, F_m(\mathbf{x})) = \begin{cases} 0 & y \cdot F_m(\mathbf{x}) > 0, \\ 1 & y \cdot F_m(\mathbf{x}) < 0, \\ l & y \cdot F_m(\mathbf{x}) = 0. \end{cases} \tag{4}$$

Nevertheless, the interpretation of l in the above definition is not as easy as in the case of a single rule. It depends on all other parameters used in Algorirhm 1. $L(y, F_m(\mathbf{x}))$ takes value equal to l in two cases. The first case is, when $F_0(\mathbf{x})$ is set to zero (there is no default rule) and no rule generated in m iterations cover object $\mathbf{x}$. The second case is when rules covering object $\mathbf{x}$ indicate equally two classes Cl_{-1} and Cl_1. The interpretation of l is similar to the case of a single rule, when $F_0(\mathbf{x})$ is set to zero and $\nu = 1/M$. Note that $\nu = 1/M$ means that each next rule is more important than all previously generated.

Classification procedure is performed according to:

$$F(\mathbf{x}) = sign(a_0 + \sum_{m=1}^{M} a_m r_m(\mathbf{x}, \mathbf{c})). \tag{5}$$

In other words, it is a linear classifier in a very high dimensional space of derived decision rules that are highly nonlinear functions of the original predictor variables $\mathbf{x}$. Parameters $\{a_m\}_0^M$ can be obtained in many ways. For example, they can be set to fixed values (for example, a_0=0 and $\{a_m = 1/M\}_1^M$), computed by some optimization techniques, fitted in cross-validation experiments or estimated in a process of constructing the ensemble (like in AdaBoost [22]).

To perform our experiment, we have used simple greedy heuristic to construct a single decision rule. It consists in searching for $\mathbf{c}$ such that $L_m = \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i, \mathbf{c}))$ is minimal. At the beginning, the complex contains an universal selector (i.e., all objects are covered by the rule). In the next step, a new selector is added to the complex and a decision of the rule is set. The selector and the decision are chosen to give the minimal value of L_m. This step is repeated until L_m is minimized. Remaining settings of the algorithm are as follows. We have decided to generate default rule indicating the majority class. Besides (4), we have tried several formulas for loss function. The best results were obtained, when we used sigmoidal function:

$$L(y, F_m(\mathbf{x})) = \frac{1}{1 - exp(y \cdot F_m(\mathbf{x}))}, \tag{6}$$

and this formulation was used in the experiment (for a wide discussion on different formulas for loss function see [10]). The "shrinkage" parameter was set to $\nu = 0.5$. Each rule is generated using subsample of size $\eta = N$ drawn with replacement. The classification is performed according to (5), where $a_0 = F_0(\mathbf{x})$ and $\{a\}_1^M$ are set to 1.

4 Missing Values

Decision rule models are well-suited to problems where objects have missing values. A single decision rule involves only a part of attributes. So, the algorithm can seek for such selectors on attributes that "avoid" missing values. In this paper, we have performed the experiment in which we have compared simple, but also effective methods to deal with missing values inspired by rough set approach to knowledge discovery. We assume here that the missing value is interpreted in such a way that its true value is one of all possible values from the domain of the considered attribute. This is consistent with the *disjunctive and exclusive interpretation* of multi-valued information systems given by Düntsch, Gediga and Orłowska in [8].

There are two problems to be solved. The first one is the way, in which a single rule is generated in the presence of missing values. The second one is the way, in which an unseen object having missing values is classified by a rule. The first problem can be solved in the following ways:

(1) objects with missing values are discarded from the further analysis (this is the simplest solution; we implemented it to have comparison with further approaches),
(2) object with missing values satisfies all selectors built on attributes on which object has no value,
(3) object with missing values does not satisfy selectors built on attributes on which object has no value,
(4) object with missing values does not satisfy selectors built on attributes on which object has no value for the rules with decision concordant with a decision of the object, and satisfies all selectors built on attributes on which object has no value for the rules with decision opposite to a decision of the object.

The second problem can be solved in the following ways:

(a) an unseen object with missing values satisfies selectors built on attributes on which object has no value,
(b) an unseen object with missing values does not satisfy selectors built on attributes on which object has no value.

Approaches (2) and (a) are motivated by the fact that an object having a missing value on an attribute may be treated as indiscernible with other object on this attribute. So, any selector on this attribute is satisfied by this object. One may also take another definition of indiscernibility, where only an universal selector on the given attribute (i.e., all objects satisfy such an selector) covers an object with missing value on this attribute. This definition is taken in approaches (3) and (b).

The approach (4), in our opinion, is the most concordant with the rough set theory. According to Kryszkiewicz [18], generated rules should remain true when all or some missing values will be replaced by arbitrary values. Such an approach

Table 1. Approaches to missing values that were used.

Training phase	Testing phase	Abbrev.	Training phase	Testing phase	Abbrev.
(1)	(a)	1-a	(3)	(a)	3-a
(1)	(b)	1-b	(3)	(b)	3-b
(2)	(a)	2-a	(4)	(a)	4-a
(2)	(b)	2-b	(4)	(b)	4-b

is compatible with knowledge discovery from incomplete information systems. In fact, considered here classification problem is quite different. Nevertheless, the linear combination of rules from the ensembles implies a partition of the space of all joint predictor variable values $\mathbf{x}$ to some regions that might be interpreted as granules of knowledge considered in rough set approaches. These granules contain mainly object from one class. An object with missing value may either belongs to such a granule or not depending on the true value of the attribute with the missing value. It is also possible that the implied granule include a whole domain of an attribute. Then objects with missing values on this attribute may be contained in this granule. So, if a rule contains a selector built on attribute on which an object has a missing value, it is reasonable to penalize, when this object belongs to opposite class than the class indicated by the rule, and do not reward when this object belongs to the class indicated also by the rule. It is implemented by assuming that object with missing values does not satisfy selectors built on attributes on which object has no value for the rules with decision concordant with a decision of the object, and satisfies all selectors built on attributes on which object has no value for the rules with decision opposite to a decision of the object. Then, the loss of the rule increases when covering an object with missing value from the opposite class, and do not decrease when covering an object with missing value from the class concordant with the decision of the rule.

In classification procedure, it is reasonable to proceed according to (b), i.e., only universal selectors on a given attribute covers an object with missing value on this attribute. It is so, because, the true value of the object on the attribute is unknown, and we are certain that only universal selector will cover it.

5 Experimental Results

We designed an experiment to compare performance of the ensemble of decision rules with different approaches to missing values. To conduct this experiment we implemented the method in Weka package [25]. We have decided to compare all combinations of approaches described in Section 4.

For purpose of the experiment, we used four data sets taken from UCI [19] repository of machine learning data sets. These sets contain two-class classification problems that have large number of objects with missing values. The data

Table 2. Number of attributes and objects for data sets included in the experiment.

Data set	Attributes	Obj. class -1	Obj. class 1	Obj. with missing values
colic	23	232	136	361
vote	17	267	168	203
labor	17	20	37	56
hepatitis	20	32	123	75

sets that we chosen are presented in Table 2. Hepatitis problem is highly unbalanced, but we have not used any technique to deal with this problem. Labor data contains relatively small number of objects. We have decided to generate 15, 30, and 60 rules and compare the performance of the algorithm for different number of rules. To estimate classifiers error rates, we used 10-fold cross-validation that we repeated 30 times. Results of the experiment are presented in Tables 3–4. In those tables we show performance of classifiers on factors (given as percents): correctly classified examples (C), true positive in class +1 (TP +1), precision in class +1 (P +1), true positive in class -1 (TP -1), precision in class -1 (P -1).

It is easy to observe that discarding objects with missing values in the case of colic and labor data sets is an unacceptable technique. However, for vote and hepatitis problem, where the number of objects with missing values is relatively smaller, the difference to more advance techniques is not so visible. For colic problem technique 2-b seems to be the worst. One may see a slight superiority of the 4-b approach. The higher number of rules in this case do not improve the results significantly. It is hard to indicate the best technique in the case of vote data. Also, the performance is good regardless for the number of rules and chosen approach to missing values. We expect that there are some easy to discover general patterns in this problem. The best performance for hepatitis problem is achieved by the 3-a technique (particularly, the true positive ratio for class 1 has an acceptable high level). The number of rules in this case plays an important role. For 60 rules the true positive ratio is the highest. In our opinion, it is caused by unbalanced number of objects in decision classes. For the labor problem that contains relatively small number of objects, one may see that the higher the number of rules than the performance is better. The best results for labor are obtained by 2-a and 3-b techniques.

Concluding the above results, it is hard to point out the best approach to deal with missing values. For sure, it is not a good idea to discard objects with some missing values from the analysis. In some cases, it does not decrease results, but it can provide to unacceptable results. It seems that the poorest results from the more advance techniques are taken for 2-b. The increase of number of rules, certainly, does not lead to worse results. However, in some cases, it does not improve results significantly, but the cost of computations is of course higher. Unfortunately, we can not say that the technique 4-b, which is the most

Table 3. Classification results in percents [%], part 1; C indicates *Correctly classified examples*, TP +1 *True Positive* in class +1, P +1 *Precision* in class +1, TP -1 *True Positive* in class -1, P -1 *Precision* in class -1.

Classifier	colic					vote				
	C	TP +1	P +1	TP -1	P -1	C	TP +1	P +1	TP -1	P -1
15 rules, 1-a	63.04	100.00	0.00	63	0	94.89	92.52	98.66	99.11	89.22
15 rules, 1-b	63.04	100.00	0.00	63	0	95.63	94.80	97.00	98.10	92.10
15 rules, 2-a	84.92	91.78	73.22	85.39	83.97	95.57	94.13	97.84	98.56	91.32
15 rules, 2-b	79.58	95.29	52.80	77.54	86.85	95.59	94.80	96.90	98.03	92.09
15 rules, 3-a	84.86	92.03	72.63	85.16	84.29	94.97	92.78	98.42	98.95	89.57
15 rules, 3-b	84.98	92.01	72.99	85.33	84.29	95.63	94.80	97.00	98.10	92.10
15 rules, 4-a	85.57	93.90	71.35	84.84	87.30	95.49	93.92	97.96	98.65	91.05
15 rules, 4-b	85.52	93.76	71.46	84.86	87.06	95.59	94.80	96.90	98.03	92.09
30 rules, 1-a	63.04	100.00	0.00	63	0	95.26	93.75	97.66	98.45	90.76
30 rules, 1-b	63.04	100.00	0.00	63	0	95.62	94.80	96.98	98.09	92.10
30 rules, 2-a	85.08	91.67	73.83	85.67	83.91	95.51	94.70	96.82	97.97	91.95
30 rules, 2-b	79.59	94.25	54.58	78.01	84.80	95.54	94.85	96.66	97.87	92.14
30 rules, 3-a	85.14	92.25	73.02	85.37	84.71	95.57	94.52	97.26	98.22	91.75
30 rules, 3-b	85.64	92.14	74.56	86.06	84.78	95.58	94.80	96.86	98.01	92.08
30 rules, 4-a	85.51	93.62	71.67	84.94	86.82	95.39	94.72	96.46	97.74	91.95
30 rules, 4-b	85.69	93.78	71.89	85.06	87.14	95.48	94.90	96.40	97.70	92.21
60 rules, 1-a	63.04	100.00	0.00	63	0	95.57	94.70	96.96	98.05	91.99
60 rules, 1-b	63.04	100.00	0.00	63	0	95.56	94.86	96.68	97.89	92.16
60 rules, 2-a	84.64	90.86	74.02	85.65	82.64	95.63	95.35	96.04	97.47	92.87
60 rules, 2-b	80.17	93.65	57.18	78.89	84.11	95.66	95.34	96.14	97.53	92.85
60 rules, 3-a	84.72	92.25	71.87	84.84	84.50	95.61	95.29	96.08	97.50	92.78
60 rules, 3-b	85.76	91.94	75.20	86.35	84.58	95.54	95.09	96.24	97.60	92.49
60 rules, 4-a	85.34	92.5	73.11	85.44	85.12	95.90	95.77	96.10	97.51	93.44
60 rules, 4-b	85.91	93.29	73.31	85.64	86.53	95.48	94.90	96.40	97.70	92.21

concordant with rough set approach, is the one achieving the best results. Some further investigations are still required.

6 Conclusions and Future Plans

We have described a general algorithm constructing an ensemble of decision rules, for which we have experimented with different approaches to deal with missing values. These approaches were inspired by rough set approach to knowledge discovery. It seems that it is hard to point out the best approach. Let us underline that the decision rule models are well-suited to problems where objects have missing values. A single decision rule involves only a part of attributes. So, the algorithm can seek for such selectors on attributes that "avoid" missing values. We plan to use another techniques that deal with missing values. These migth be, for example, surrogate selectors by similarity to surrogate splits in CART [5] and the approach taken in C4.5 [21].

Table 4. Classification results in percents [%], part 1; C indicates *Correctly classified examples*, TP +1 *True Positive* in class +1, P +1 *Precision* in class +1, TP -1 *True Positive* in class -1, P -1 *Precision* in class -1.

Classifier	hepatitis					labor				
	C	TP +1	P +1	TP -1	P -1	C	TP +1	P +1	TP -1	P -1
15 rules, 1-a	81.74	15.94	98.87	77.40	81.90	38.13	90.00	10.08	35.11	64.91
15 rules, 1-b	81.51	16.68	98.39	74.30	81.96	37.72	90.00	9.45	34.95	63.35
15 rules, 2-a	83.01	29.07	97.04	72.29	84.02	84.21	63.50	95.41	88.19	82.93
15 rules, 2-b	82.82	25.63	97.70	75.74	83.47	74.56	31.83	97.66	89.05	72.71
15 rules, 3-a	82.92	27.10	97.46	73.94	83.71	79.82	66.50	87.04	73.92	82.94
15 rules, 3-b	82.86	25.85	97.71	74.73	83.51	84.39	64.00	95.41	88.44	83.21
15 rules, 4-a	81.81	19.09	98.14	73.52	82.34	81.17	58.83	93.25	82.57	80.77
15 rules, 4-b	81.57	17.72	98.20	72.16	82.10	82.75	58.83	95.68	88.07	81.19
30 rules, 1-a	82.95	27.20	97.46	74.39	83.73	38.13	90.00	10.08	35.11	64.91
30 rules, 1-b	82.99	31.05	96.49	70.99	84.33	38.25	90.00	10.26	35.16	65.36
30 rules, 2-a	83.72	40.02	95.08	68.24	85.92	88.66	76.33	95.32	89.96	88.23
30 rules, 2-b	82.60	34.28	95.16	65.12	84.79	79.59	47.67	96.85	89.60	77.54
30 rules, 3-a	84.37	46.36	94.25	67.99	87.12	82.81	77.83	85.51	74.71	87.75
30 rules, 3-b	82.65	38.13	94.22	63.36	85.43	87.95	74.17	95.41	89.90	87.31
30 rules, 4-a	82.06	29.49	95.72	64.49	83.94	84.39	69.17	92.62	83.59	84.83
30 rules, 4-b	82.13	29.81	95.72	64.84	84.00	84.15	66.33	93.79	85.56	83.84
60 rules, 1-a	83.81	33.98	96.76	74.59	84.93	38.13	90.00	10.08	35.11	64.91
60 rules, 1-b	83.25	33.66	96.13	69.57	84.80	38.13	90.00	10.08	35.11	64.91
60 rules, 2-a	83.03	42.93	93.47	63.18	86.30	90.82	83.50	94.78	89.75	91.45
60 rules, 2-b	83.01	37.62	94.81	65.77	85.40	83.28	60.00	95.86	88.94	81.67
60 rules, 3-a	84.52	55.22	92.15	64.78	88.78	86.14	80.33	89.29	80.58	89.40
60 rules, 3-b	83.23	41.89	93.98	64.46	86.15	90.58	81.67	95.41	90.59	90.62
60 rules, 4-a	81.81	33.34	94.41	60.93	84.50	86.37	74.50	92.80	84.97	87.11
60 rules, 4-b	81.59	31.68	94.57	60.46	84.19	86.78	73.83	93.79	86.58	86.99

Acknowledgements. The authors wish to acknowledge financial support from the Ministry of Education and Science (grant no. 3T11F 02127).

References

1. Michalski, R.S.: A Theory and Methodology of Inductive Learning. In Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.): Machine Learning: An Artificial Intelligence Approach. Palo Alto, Tioga Publishing, (1983) 83–129
2. Błaszczyński, J., Dembczyński, K., Kotłowski, W., Słowiński, R., Szeląg, M.: Ensemble of Decision Rules. (submitted) (2005)
3. Boros, E., Hammer, P. L., Ibaraki, T., Kogan, A., Mayoraz, E., Muchnik, I.: An Implementation of Logical Analysis of Data. IEEE Trans. on Knowledge and Data Engineering **12** (2000) 292–306
4. Breiman, L.: Bagging Predictors. Machine Learning **24** 2 (1996) 123–140
5. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.: Classification and Regression Trees. Wadsworth, (1984)

6. Clark, P., Nibbet, T.: The CN2 induction algorithm. Machine Learning **3** (1989) 261–283
7. Cohen, W., W., Singer, Y.: A simple, fast, and effective rule learner. Proc. of 16th National Conference on Artificial Intelligence (1999) 335–342
8. Düntsch, I., Gediga, G., Orłowska, E.: Relational attribute systems. International Journal of Human-Computer Studies, **55** (2001) 293–309
9. Friedman, J. H., Hastie, T. and Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), August (1998)
10. Friedman, J. H., Hastie, T. and Tibshirani, R.: Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer (2003)
11. Friedman, J. H.: Recent advances in predictive (machine) learning. Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), November (2003)
12. Friedman, J. H., Popescu, B. E.: Gradient directed regularization. Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), February (2004)
13. Friedman, J. H., Popescu, B. E.: Predictive Learning via Rule Ensembles. Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), February (2005)
14. Greco S., Matarazzo, B. and Słowiński, R.: Dealing with missing data in rough set analysis of multi-attribute and multi-criteria decision problems. [In]: Zanakis, S. H., Doukidis, G., Zopounidis C. (eds.), Decision Making: Recent Developments and Worldwide Applications. Kluwer Academic Publishers, Dordrecht (2000) 295–316
15. Grzymala-Busse, J., W.: LERS — A system for learning from examples based on rough sets. In Słowiński, R. (ed.): Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory. Kluwer Academic Publishers (1992) 3–18
16. Grzymala-Busse, J. W., Hu, M.: A Comaprison of Several Approaches in Missing Attribute Values in Data Mining, LNAI **2005** (2000) 378–385
17. Grzymala-Busse, J. W.: Incomplete Data and Generalization of Indiscernibility Relation, Definability, and approximation, LNAI **3614** (2005) 244–253
18. Kryszkiewicz, M.,: Rough Set approach to incomplete information systems. Information Sciences **112** (1998) 39–49
19. Newman, D., J., Hettich, S., Blake, C., L., Merz, C.,J. (UCI) Repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (last access: 01.05.2006), Dept. of Information and Computer Sciences, University of California, Irvine (1998)
20. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht, (1991)
21. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
22. Schapire, R. E., Freund, Y., Bartlett, P, Lee, W. E.: Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics **26** 5 (1998) 1651–1686
23. Skowron, A.: Extracting laws from decision tables - a rough set approach. Computational Intelligence **11** 371–388
24. Stefanowki, J.: On rough set based approach to induction of decision rules. In Skowron,A. and Polkowski, L. (eds): Rough Set in Knowledge Discovering, Physica Verlag, Heidelberg (1998) 500–529
25. Witten, I., H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd Edition. Morgan Kaufmann, San Francisco (2005)