RULEM : a novel heuristic rule learning approach for ordinal classification with monotonicity constraints

RULEM: A Novel Heuristic Rule Learning Approach for Ordinal Classification with Monotonicity Constraints

Wouter Verbeke[a], David Martens[b], Bart Baesens[c,d]

[a]*Faculty of Economics and Social Sciences and Solvay Business School, Vrije Universiteit Brussel, Belgium*
[b]*Faculty of Applied Economics, University of Antwerp, Belgium*
[c]*Faculty of Economics and Business, Katholieke Universiteit Leuven, Belgium*
[d]*Southampton Business School, University of Southampton, United Kingdom*

Abstract

In many real world applications classification models are required to be in line with domain knowledge and to respect monotone relations between predictor variables and the target class, in order to be acceptable for implementation. This paper presents a novel heuristic approach, called RULEM, to induce monotone ordinal rule based classification models. The proposed approach can be applied in combination with any rule- or tree-based classification technique, since monotonicity is guaranteed in a postprocessing step. RULEM checks whether a rule set or decision tree violates the imposed monotonicity constraints and existing violations are resolved by inducing a set of additional rules which enforce monotone classification. The approach is able to handle non-monotonic noise, and can be applied to both partially and totally monotone problems with an ordinal target variable. Two novel justifiability measures are introduced which are based on RULEM and allow to calculate the extent to which a classification model is in line with domain knowledge expressed in the form of monotonicity constraints. An extensive benchmarking experiment and subsequent statistical analysis of the results on 14 public data sets indicates that RULEM preserves the predictive power of a rule induction technique while guaranteeing monotone classification. On the other hand, the post-processed rule sets are found to be significantly larger which is due to the induction of additional rules. E.g., when combined with Ripper a median performance difference was observed in terms of PCC equal to zero and an average difference equal to -0.66%, with on average 5 rules added to the rule sets.The average and minimum justifiability of the original rule sets equal respectively 92.66% and 34.44% in terms of the RULEMF justifiability index, and 91.28% and 40.1% in terms of RULEMS, indicating the effective need for monotonizing the rule sets.

Key words: Ordinal Classification, Rule learning, Monotonicity Constraints, Justifiability, Heuristic, Postprocessing

1. Introduction

Classification algorithms are a family of data mining techniques that are used to predict group or class membership of data instances [26, 46]. In ordinal classification, the values of the target class possess a *natural* ordering, e.g., from small to large or from good to bad [52]. A typical example of an ordinal classification problem is the estimation of bond ratings based on financial information [55]. Ratings represent the risk involved in a financial product, for instance a corporate or government bond. Financially healthy organizations are rated higher by rating agencies than organizations in financial distress. The target class, i.e., the rating, hence incorporates an ordering from good to bad. Other examples are the classification of customers in segments according to future spending, or any other classification problem with a continuous target variable that can be segmented into categories, such as for instance age, value, etc. Also binary classification problems can be of ordinal nature, such as good versus bad loan applications in credit scoring [56] and false versus non-false in customer churn prediction [60].

Many powerful classification algorithms have been developed that are able to classify instances with high precision. The workings of most of these classification algorithms are based on modeling repeated patterns or correlations which are present in the data. However, it may occur that patterns or relations which are evident for a human domain expert are not appropriately modeled by a data mining algorithm because the pattern does not appear sufficiently frequent in the data set or non-monotonic noise in the data set perturbs the pattern [38, 41, 50]. Hence, the intervention and interpretation of the induced model by a domain expert remains crucial in many applications. A data mining approach that takes into account the knowledge representing the experience of domain experts is therefore much preferred and of great focus in current data mining research [7, 8, 18, 40]. Domain knowledge in a classification setting is typically expressed as a relation between an attribute and the target class. For instance, the rating of a bond is expected to be positively related to the solvency of a company.

In the literature, monotonicity is the most frequently encountered domain constraint to be incorporated within a classification model. A positive (negative) monotonicity constraint demands that an increase in a certain input cannot lead to a decrease (increase) in the output. Monotonicity constraints exist in almost any domain. It should be stressed that in practice the justifiability and comprehensibility of a model will almost always be more important to the users of the model than the predictive power [2, 40].

For example, let's consider the credit scoring case, where classification models are used to predict the creditworthiness of new loan applications. Suppose a rule-based technique provides a single rule that states *if (profitability* $>$ 10%*) then creditworthiness = bad*. Although this rule might quite accurately describe the available observations, the domain knowledge of the credit analyst will likely state the inverse: *low* profitability should lead to bad creditworthiness. In that case, the analyst will prefer a model that is in line with domain knowledge instead of the, possibly limited amount of, available data. A second example in the field of health analytics concerns predictive classification models that are used to predict the probability of breast cancer recurrence based on the characteristics of a patient. Domain knowledge states that an increase in tumor size leads to higher probability of recurrence. Therefore, a predictive model that classifies two patients with a small and a large tumor, assuming all other characteristics to be exactly the same, as respectively having a high and a low probability of recurrence, violates the expected monotone relation between probability of recurrence and tumor size.

Beyond this intuition, numerous case studies reveal that users are reluctant or even refuse to use a model that is contradicting domain knowledge, e.g. in domains such as credit scoring [55], customer churn prediction [59], medical diagnosis [27], audit mining (predicting the going concern opinion as issued by the auditor) [35], fraud detection [3], and software effort prediction [15].

The main contributions of this paper are as follows. A novel heuristics approach is presented to induce monotone ordinal classification models, and more specifically monotone rule or tree based classifiers, which incorporate and respect monotonicity constraints. The proposed approach is called RULEM, which stands for RUle LEarning with Monotonicity constraints. RULEM enforces monotonicity during a postprocessing step, i.e., after the classifier is induced. Therefore RULEM can be applied in combination with any rule or decision tree induction technique, which is a major advantage. The RULEM algorithm exists of two modules. The first module checks whether an existing rule set or decision tree violates the imposed monotonicity constraints. If so, additional rules are induced by the second module to resolve the violations and to guarantee monotone classification.

The advantages of the novel technique are manifold. Since monotonicity is enforced during a postprocessing step, there is no interference with the inner workings of the classification techniques that RULEM is combined with. The *optimal* model induced by a classification technique is adjusted by the postprocessing module to the smallest possible extent, in order to preserve the predictive power. Furthermore, the imposed constraints can be set freely by a domain expert. The constrained variables can be selected individually and both positive and negative constraints can be imposed directly without need to preprocess the data. Moreover, the number of class labels is unrestricted, and the data set does not have to be monotone and may contain non-monotonic noise, i.e., non-monotone data pairs. The resulting classification model remains comprehensible since a minimal number of additional rules is induced, and most importantly, guarantees monotone relations between the constrained attributes and the class variable. Finally, based on the RULEM technique two novel justifiability measures are formulated which both provide an intuitive and sensible indication of the extent to which a rule or tree based classifier is in line with domain knowledge.

2. Monotone ordinal classification

The first part of this section discusses prior work, after which a general framework to the problem of ordinal monotone classification is provided [14, 33]. The third part finally introduces rule based classifiers.

2.1. Prior work

There is an extensive body of literature on monotone classification, which can be roughly categorized depending on the phase during the model induction process when the algorithm imposes monotonicity constraints or restrictions: before, during, or after inducing the actual model.

A first set of approaches presented in the literature handles monotonicity by *monotonizing* the data set that is used to build the classification model [13, 19, 34, 49]. Such approaches remove or adjust non-monotonic noise in the data set, as will be further explained in section 2.2, with the aim to direct the model induction technique towards a classification model that respects the monotone relations present in the data [42]. However, approaches solely comprising a data pre-processing step do not guarantee the final classifier to be monotone, as will be further discussed below.

A second category of approaches comprises algorithms that directly build a monotone classification model by taking into account the monotonicity constraints during the actual induction phase [1, 4, 5, 9, 10, 12, 16, 18, 25, 29, 30, 31, 33, 36, 45, 51]. Different types of classifiers have been adjusted in order to allow imposing constraints on the nature of the relations between the predictor and target variables as incorporated in the resulting classification model. The most common types of classification techniques that have been adjusted to incorporate monotonicity are decision trees [4, 9, 30, 45], neural networks [12, 14, 31, 51], Bayesian networks [1], linear programming [10], and rule-based classifiers [5, 16, 25, 29, 36]. More recently, Support Vector Machine based approaches and ensembles have been adapted for this purpose [24, 32, 47]. Remark that these approaches apply to a single type of classification technique, whereas the first set of approaches monotonizing the data apply to every possible type, but do not guarantee the monotonicity constraints to be respected.

A final class of approaches consists of algorithms to post-processes classification models in order to resolve possible conflicts with regards to monotonicity. Post-processing approaches have been developed that apply to decision trees [20, 21, 44, 54], linear regression models [55], as well as a dominance-based-rough set approach to monotone ordinal classification [28] that solves a K class problem as a set of $K-1$ binary sub-problems.

The presented approach in this paper to *monotonize* ordinal rule sets is partly similar to the methods developed for decision trees by [54] and [44], but allows a more fine-grained relabeling of parts of the attribute space, i.e., of parts of rules. This is required since changing the class label assigned by one or more rules in a rule set may resolve violations, but likely at the cost of losing significant classification power. Moreover, leaf nodes of a decision trees apply to mutually exclusive regions in the attribute space, whereas rules are typically not mutually exclusive and therefore ordered. This complicates the problem of building a monotone ordinal rule set, as will be further elaborated in the following sections.

Next to these three different types of approaches, four important challenges can be identified in the literature with regards to the monotone classification problem, i.e.: 1) improving versus guaranteeing monotonicity; 2) the ability to handle both partial and total monotonicity; 3) the applicability to a binary versus an ordinal classification problem; 4) the applicability to a single classifier (type) versus multiple classifier (types).

Inherently, the first class of approaches as listed above do not guarantee monotonicity, since no constraints are imposed during model induction and no final check is performed on the resulting classifier whether monotonicity is respected. Moreover, not all of the approaches of the second and third type, which handle monotonicity respectively during and after model induction, guarantee monotone classification, and sometimes may only result in *more* monotone models [4, 21]. In case of the AntMiner+ technique [36], the user has the choice of imposing soft or hard monotonicity constraints. Soft constraints do not guarantee the final model to be monotone, but hint or direct the learning algorithm towards a monotone classifier, to a degree depending on the user's preferences. A fully monotone model can be enforced by imposing hard constraints.

Some techniques are able to handle partially monotone problems [1, 14, 31, 36, 45, 55] without having to remove non-monotone variables or without having to enforce a monotone relationship for non-monotonic variables, i.e., without having to treat the problem as a totally monotone problem while it is not.

Including monotonicity for an ordinal, as opposed to a binary target class variable, is limited to a relatively small number and specific types of approaches [5, 10, 12, 14, 16, 23, 24, 28, 31, 33, 45, 51, 55].

Finally, only the techniques that handle monotonicity in a pre-processing step by removing all non-monotone pairs, are generally applicable to any type of classifier, but as mentioned above do not guarantee monotonicity of the resulting classifier.

The RULEM technique as presented in the following section has been developed with the aim to address these different issues and to offer as much flexibility as possible, as well as to comply with a number of additional requirements with regards to justifiability measurement.

2.2. Problem description

Let $\mathcal{X} = \prod_{i=1}^{k} \mathcal{X}_i$ be an input space represented by k attributes, features, or variables. A particular observation or instance $\mathbf{x} \in \mathcal{X}$ is defined by the vector $\mathbf{x} = (x_1, x_2, \ldots, x_k)$, where $x_i \in \mathcal{X}_i$ and $i = 1$ to k. Furthermore, a totally ordered set of labels $\mathcal{L} = \{\ell_l\}$ is defined, with $l = 1$ to h and $\ell_l < \ell_{l+1}$. A function f is defined which maps to each attribute vector $\mathbf{x}$ a label $\ell \in \mathcal{L}$, i.e., $f : \mathcal{X} \to \mathcal{L}$. In classification problems, the objective is to find an approximation $\hat{f}$ of f as close as possible according to a certain distance measure, based on the information that is contained in a data set $D = (\mathbf{x}^a, \ell^a)$ with $a = 1$ to o and o the number of observations. In the literature, a range of classification techniques have been proposed to induce classification models $\hat{f}$. For an overview, one may refer to, e.g., [26, 53].

The main assumption in this study is that f exhibits monotonicity properties with respect to the input variables, and therefore $\hat{f}$ should obey these properties as well in a strict fashion. Two types of problems and models can be distinguished, based on the set of input variables that are monotonically related to the target variable. Total positive monotonicity of $\hat{f}$ on $\mathbf{x}$ is defined on all independent variables by:

$$\mathbf{x}^1 \geq \mathbf{x}^2 \quad \Rightarrow \quad \hat{f}(\mathbf{x}^1) \geq \hat{f}(\mathbf{x}^2), \tag{1}$$

Partial positive monotonicity of the classifier $\hat{f}$ is defined by:

$$\mathbf{x}^1_{nm} = \mathbf{x}^2_{nm} \text{ and } \mathbf{x}^1_m \geq \mathbf{x}^2_m \quad \Rightarrow \quad \hat{f}(\mathbf{x}^1) \geq \hat{f}(\mathbf{x}^2), \tag{2}$$

		Rule set $\mathcal{R}$						
Rule ID	**Order**	**Rule antecedents**					**Rule consequent**	
r_1	1	*if*	$profits \geq 1$	$\wedge$	$solvency \geq 3$		*then*	$rating = A$
r_2	2	*if*	$profits < 2$	$\wedge$	$solvency \geq 1$	$\wedge$ $solvency < 2$	*then*	$rating = B$
r_3	3	*else*					*then*	$rating = C$

Table 1: A simple example rule set, representing the classification of a company into three possible rating classes A, B, and C based on the values of two attributes, i.e., *profits* and *solvency*.

with the subscripts nm and m referring to respectively the independent variables $\mathcal{X}$ that are non-monotonically and monotonically related to the dependent variable, and which together constitute the instance vector $\mathbf{x} = (\mathbf{x}_{nm}, \mathbf{x}_m)$. Therefore, $\mathcal{X}_{nm}$ denotes the set of *non-monotone variables*, and the complementary set $\mathcal{X}_m$ refers to the *monotone variables*.

In case of positive monotonicity constraints, a pair of instances $(\mathbf{x^1}, \mathbf{x^2})$ of the data set D is called *comparable* if $\mathbf{x}^1 \geq \mathbf{x}^2$. Furthermore, this pair is also a *monotone pair* if the relationship $\hat{f}(\mathbf{x}^1) \geq \hat{f}(\mathbf{x}^2)$ holds. Note that although the relation f might be totally monotone, not all the pairs in the data set are necessarily monotone, since non-monotonic noise can exist in the data set D. Based on the concepts of comparable and monotone pairs, a test to measure the degree of monotonicity (DgrMon) in a data set D has been introduced earlier [14] which is defined as follows:

$$\text{DgrMon}(D) = \frac{\#\text{Monotone pairs}(D)}{\#\text{Comparable pairs}(D)} \tag{3}$$

The degree of monotonicity varies by definition between zero and one. A value close to one indicates that the label has an increasing monotone relationship with the independent variables. A value close to zero on the other hand indicates that the response variable is decreasing with the increase in the independent variables. A value close to 0.5 indicates either that there are no monotone relationships between the class variable and the attributes, for instance when the labels are randomly distributed, or either that there are an even amount of negative and positive monotone relations.

2.3. Rule-based classifiers

Rule-based classifiers are a type of classification model consisting of a number of *if-then* rules. Table 1 provides a very simple example of a rule set, which will be used throughout the paper for illustrative purposes. The rule set assigns a rating to a company (in an entirely fictional manner) based on its *profits* and *solvency*, which is defined as the degree to which the current assets of a company exceed the current liabilities. Remark that rating A represents the highest credit quality and rating C the lowest credit quality. The rules of this classification model are represented in a disjunctive normal form, $\mathcal{R} = (r_1 \vee r_2 \vee \ldots \vee r_n)$, with $\mathcal{R}$ the *rule set* containing q *classification rules* or *disjuncts* r_e with $e = 1$ to q. A classification rule can be expressed as

$$r_e : p_e\,(\mathbf{x}) \rightarrow \ell_e, \tag{4}$$

with

$$p_e\,(\mathbf{x}) = \bigwedge_{i=1}^{k} c_{i,e} \;\text{ and }\; c_{i,e} = (x_i \; op \; v_{i,e}), \tag{5}$$

where $(x_i, v_{i,e})$ is an attribute-value pair and op is a logical operator chosen from the set $\{=, \neq, <, >, \leq, \geq\}$. The *if*-part of a rule, $p_e\,(\mathbf{x})$, is called the *precondition* or the *rule antecedent*, and contains a conjunction of *attribute tests* or *conjuncts* $c_{i,e}$. The *then*-part of each rule is called the *rule consequent*, and assigns a class label $\ell \in \mathcal{L}$ to the instances that match the precondition.

An instance $\mathbf{x}$ can match multiple rules if the rule set is *ordered*. However, a class label will be assigned by the highest ranked rule that covers an instance, i.e., the rule with the highest priority. Mutually exclusive rule sets on the other hand consist of non-overlapping rules, and an instance will trigger at most one precondition. The conversion of a decision tree into rules yields a rule set that is mutually exclusive. The RULEM approach presented in this paper is able to handle both ordered and mutually exclusive rule sets.

Rules can be represented as hypercubes in a k dimensional space, with k the number of attributes in the data set. Each test in a conjunct of a rule defines the bounds of the hypercube in a particular dimension. For instance, the simple rule set of Table 1 is represented by squares in the two-dimensional attribute space as depicted in Figure 1. Rules may not explicitly set bounds for each dimension. For instance, the first rule in the example rule set does not set an upper bound on the value of the *solvency* attribute. Therefore, in Figure 1 the corresponding hypercube stretches towards the maximum possible value of the *solvency* attribute, which is set to four in this example. The representation of rules as hypercubes will be used to explain the workings of the RULEM technique, and allows to visualize how violations of monotonicity by a rule set can be detected and resolved.

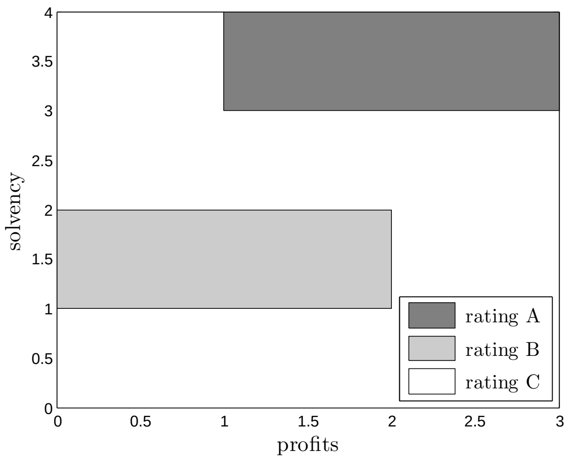

Figure 1: Graphical representation in the two dimensional attribute space of the rule set in Table 1.

3. Detecting violations of monotonicity constraints

3.1. Rule-based classifiers and violations of monotone classification

A totally or partially monotone rule-based classification model is defined as a rule set that complies with respectively Equations 1 or 2 over the entire input space $\mathcal{X} = \prod_{i=1}^{k} \mathcal{X}_i$. This means that Equations 1 or 2 apply to each possible pair of comparable attribute vectors $(\mathbf{x}^1, \mathbf{x}^2) \in \mathcal{X}$, with $\mathbf{x}^1 \geq \mathbf{x}^2$ in case of total monotonicity or $\mathbf{x}^1_{nm} = \mathbf{x}^2_{nm}$ and $\mathbf{x}^1_m \geq \mathbf{x}^2_m$ in case of partial monotonicity. To our knowledge, the only approach that is described in the literature to check whether a rule set is monotone over the entire attribute space $\mathcal{X}$ was introduced in [40]. However, this approach requires the conversion of a rule set in a decision table (see, e.g., [58]) to calculate justifiability, which prohibits automation and incorporation within a general algorithm to induce monotone rule sets.

An alternative approach exists in applying Equations 1 and 2 of total and partial monotonicity in a straightforward manner, and to exhaustively check the classification of all comparable attribute vector pairs in the attribute space. Remark that this is not equivalent to checking all the comparable attribute vector pairs *in a data set*, which is done for instance to calculate the degree of monotonicity as defined by Equation 3. In most settings the attribute space $\mathcal{X}$ is very large and checking all comparable attribute vectors infeasible. Therefore, a novel approach is introduced in this paper which starts from the rules to check whether a rule set complies with the imposed monotonicity constraints.

The first step of this approach consists in partitioning the k-dimensional attribute space $\mathcal{X}$ into a grid $\mathcal{G}$, which consists of elementary cells $\mathbf{g}$ with *homogeneous labeling*, meaning that all attribute vectors $\mathbf{x} \in \mathbf{g}$ yield the same class label. Cells are bounded by the values of the attribute tests in the preconditions of the rules, and by the minimum and maximum attribute values, i.e., the bounds of the attribute space $\mathcal{X}$. All the attribute vectors in an elementary cell trigger the same rule in the rule set and are assigned the same label, since by definition no rule in the rule set makes a further differentiation between the attribute values within an elementary cell.

The grid $\mathcal{G}$ of elementary cells is formally defined as follows:

$$\mathcal{G} = \prod_{i=1}^{k} \mathcal{G}_i, \tag{6}$$

with

$$\mathcal{G}_i = \bigcup_{j=1}^{n_i} g_{i,j} = \bigcup_{j=1}^{n_i} [v^s_{i,j}, v^s_{i,j+1}), \tag{7}$$

and

$$v^s_{i,1} = min(\mathcal{X}_i), \tag{8}$$

$$v^s_{i,n_i+1} = max(\mathcal{X}_i) + \varepsilon. \tag{9}$$

The elementary value ε is added to the maximum attribute value in Equation 9 to account for the open upper bound of the elementary intervals in Equation 7. Each dimension $\mathcal{X}_i$ of the attribute space $\mathcal{X}$ is partitioned into elementary intervals $g_{i,j}$, with $i = 1$ to k and $j = 1$ to n_i. The bounds of the elementary intervals are defined by the ascending set s of attribute values $v^s_{i,j}$, which consists of (1) the minimum and maximum values $v^s_{i,1}$ and v^s_{i,n_i+1} of each attribute, defined in Equations 8 and 9 respectively as the lower and upper bound of each dimension; (2) the unique attribute values $v_{i,e}$ in the preconditions p_e of the rules r_e in the rule set $\mathcal{R}$, with $v_{i,e} \neq v^s_{i,1}$ and $v_{i,e} \neq v^s_{i,n_i+1}$. Each cell $\mathbf{g} \in \mathcal{G}$

consists of a unique combination of elementary intervals, i.e., $\mathbf{g} = (g_1, g_2, \dots, g_k)$, with $g_i \in \{g_{i,1}, g_{i,2}, \dots g_{i,n_i}\}$, and n_i the number of intervals constituting attribute dimension $\mathcal{X}_i$. The number of elementary cells in the grid $\mathcal{G}$ equals $\prod_{i=1}^{k}(n_i)$.

RULEM adopts a convention which restricts the set of logical operators in a rule set to $\{=, \neq, <, \geq\}$. The $\leq$ and $>$ operator are converted respectively into the $<$ and $\geq$ operator by replacing attribute tests $x_i \leq v_{i,e}$ by $x_i < v_{i,e} + \varepsilon$, and attribute tests $x_i > v_{i,e}$ by $x_i \geq v_{i,e} + \varepsilon$, with ε a value smaller than the minimum difference between any two values of an attribute in the data set. This conversion increases the number of elementary intervals n_i of attribute dimension $\mathcal{X}_i$, and consequently the total number of cells in the grid, but allows to partition a rule set in a grid $\mathcal{G}$ with elementary cells that are strictly complementary, i.e., $g_{i,j} \cap g_{i',j'} = \varnothing$ with $i \neq i'$ or $j \neq j'$. Excluding the operators $\leq$ and $>$ impedes cells to overlap in the boundary values. The resulting intervals $g_{i,j}$ are all left-closed and right-open, as indicated in Equation 7.

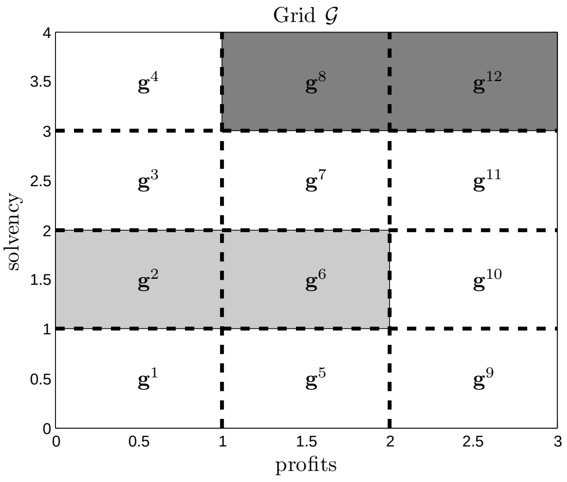

Figure 2: The elementary grid of the example rule set.

Figure 2 shows the partitioning into a grid of the simple example rule set of Table 1, and labels each cell in the grid. As can be seen, each cell in the grid is homogeneously labeled.

3.2. *Detecting violations of monotonicity constraints*

In a second step of the RULEM approach to detect violations of the monotonicity constraints, the *Conflict* or *C-score* is calculated for each cell in the grid. The C-score of a cell indicates the extent to which the label of a cell violates the imposed constraints. It is important to acknowledge that the label of a cell on itself does not cause any violation, but that in combination with the labels of other cells a violation may exist. The C-score of a cell in the grid is calculated as the number of other cells it is conflicting with, with respect to the monotonicity constraints. The sum of all C-scores of the cells yields the total C-score of the rule set; if the total C-score is equal to zero, the rule set respects the imposed monotonicity constraints. If the total C-score is different from zero, the rule set violates the imposed monotonicity constraints. A formal algorithm describing the calculation of the C-score of each cell in the grid $\mathcal{G}$ by the RULEM algorithm is provided by Algorithm 1. The algorithm first constructs the grid $\mathcal{G}$ and subsequently checks for each cell (first for loop at line 3 of the algorithm) whether it is in violation with any of the constraints that are imposed (second for loop at line 5), by checking whether conflicts exist by comparing to other cells in the concerning dimension (third for loop at line 6). The actual comparison with other cells that is made depends on the type of constraint, either positive or negative (if-else clause at lines 7-15-23), as well as whether the controlled cell has an attribute value in the controlled dimension smaller than or larger than the cell it is compared to (inner if-else clause at lines 8-11-14 and 16-19-22).

To illustrate the concept of the C-score let us return to the simple rule set of Table 1 and the corresponding representations of Figures 1 and 2. Assume that a positive constraint is imposed on the *solvency* attribute. According to this constraint, cell $\mathbf{g}^1$ is not in conflict with cells $\mathbf{g}^2$, $\mathbf{g}^3$, or $\mathbf{g}^4$ since for an increasing value of the *solvency* attribute, neither of these cells assigns a label or class that is smaller, with C the lowest and A the highest rating class, than the label assigned by cell $\mathbf{g}^1$. Therefore cell $\mathbf{g}^1$ is assigned a C-score equal to 0. Remark that conflicts are commutative, meaning that if cell $\mathbf{g}^1$ is (not) in conflict with cell $\mathbf{g}^2$, then cell $\mathbf{g}^2$ is (not) in conflict with cell $\mathbf{g}^1$. Therefore in the example, since cell $\mathbf{g}^1$ is not conflicting with cell $\mathbf{g}^2$, cell $\mathbf{g}^2$ is not conflicting with cell $\mathbf{g}^1$. Cell $\mathbf{g}^2$ is however in conflict with cells $\mathbf{g}^3$ and $\mathbf{g}^4$. The labels assigned by cells $\mathbf{g}^3$ and $\mathbf{g}^4$ are smaller than the label assigned by cell $\mathbf{g}^2$, although instances that are situated in cells $\mathbf{g}^3$ and $\mathbf{g}^4$ may have an identical value for *profits* and a larger value for

Algorithm 1 RULEM pseudo-code to calculate the C-score of a rule set $\mathcal{R}$

```
1:  construct grid $\mathcal{G} = \prod_{i=1}^{k} \mathcal{G}_i$, with $i$ the attribute dimension
2:  $\mathcal{G}_i = \bigcup_{j=1}^{n_i} g_{i,j}$, with $g_{i,j}$ the elementary intervals
3:  **for all** cells $\mathbf{g}^u \in \mathcal{G}$, with $u = 1$ to $n$, $n = \prod_{i=1}^{k} n_i$, and $\mathbf{g}^u = (g_1^u, g_2^u, \ldots, g_k^u)$ **do**
4:      set C-score($\mathbf{g}^u$) = 0
5:      **for all** constrained dimensions $\mathcal{X}_m \in \mathcal{X}$ **do**
6:          **for all** cells $\mathbf{g}^{u,j} \in \mathcal{G}$, with $j = 1$ to $n_m$, $g_{i \neq m}^{u,j} = g_i^u$, and $g_m^{u,j} = g_{m,j}$ **do**
7:              **if** positive constraint **then**
8:                  **if** $g_m^{u,j} < g_m^u$ AND $\ell(\mathbf{g}^{u,j}) > \ell(\mathbf{g}^u)$ **then**
9:                      violation of constraint
10:                     increase C-score($\mathbf{g}^u$)
11:                 **else if** $g_m^{u,j} > g_m^u$ AND $\ell(\mathbf{g}^{u,j}) < \ell(\mathbf{g}^u)$ **then**
12:                     violation of constraint
13:                     increase C-score($\mathbf{g}^u$)
14:                 **end if**
15:             **else if** negative constraint **then**
16:                 **if** $g_m^{u,j} < g_m^u$ AND $\ell(\mathbf{g}^{u,j}) < \ell(\mathbf{g}^u)$ **then**
17:                     violation of constraint
18:                     increase C-score($\mathbf{g}^u$)
19:                 **else if** $g_m^{u,j} > g_m^u$ AND $\ell(\mathbf{g}^{u,j}) > \ell(\mathbf{g}^u)$ **then**
20:                     violation of constraint
21:                     increase C-score($\mathbf{g}^u$)
22:                 **end if**
23:             **end if**
24:         **end for**
25:     **end for**
26: **end for**
27: C-score($\mathcal{R}$) = $\sum_{u=1}^{n}$ C-score($\mathbf{g}^u$)
```

solvency. As such, the positive constraint on the solvency attribute is violated. Hence, cell $\mathbf{g}^2$ is assigned a C-score equal to 2.

When imposing a second, positive constraint on the *profits* attribute, the C-score of the cell $\mathbf{g}^1$ remains zero, since no conflict exists with cells $\mathbf{g}^5$ and $\mathbf{g}^9$. The C-score of cell $\mathbf{g}^2$ on the other hand increases to three, since another conflict arises with cell $\mathbf{g}^{10}$. For each cell in the grid the C-score can be calculated in a similar way, and the resulting values are presented in Figure 3. Remark that the C-score is calculated by assessing violations of a single constraint, so not of combinations of constraints. Therefore in the example in Figure 1 cell $\mathbf{g}^{11}$ has a C-value equal to zero, so is indicated not to violate any constraint, although the values of both *profits* and *solvency* are larger in cell $\mathbf{g}^{11}$ than in cell $\mathbf{g}^2$ and the assigned class label is smaller. The presented RULEM approach may be extended to handle such multivariate constraints, and to calculate C-scores that take into account violations of such multivariate constraints. However, the current version of RULEM does not implement this yet.

4. Resolving violations of monotonicity constraints

The previous section explained the workings of the RULEM algorithm to detect violations of monotonicity constraints by a rule set. In a second step, the RULEM algorithm resolves these violations by adding complementary rules to the rule set, as will be explained in Section 4.1. Section 4.2 will describe how the RULEM algorithm induces these complementary rules, in order to guarantee monotone classification with a minimum impact on the predictive power and the size of the rule set.

4.1. Adding rules to resolve monotonicity violations

Violations of monotonicity constraints by a rule set can be resolved by adding complementary rules. For instance, the example rule set of Figure 1 can be *monotonized* in order to respect the monotonicity constraints on the *profits* and *solvency* attribute by adding rules that assign rating A to cells $\mathbf{g}^3$, $\mathbf{g}^4$, $\mathbf{g}^7$, and $\mathbf{g}^{11}$, and rating B to cell $\mathbf{g}^{10}$, as shown in Figure 4(a). By adding these rules, the C-scores of all cells in the grid become zero, which indicates that the resulting rule set respects the positive monotonicity constraints on both attributes. Since cells g^3, g^7, and g^{11} are identically labeled, neighboring, and constituting a rectangular volume in the attribute space (i.e., a hypercube), the

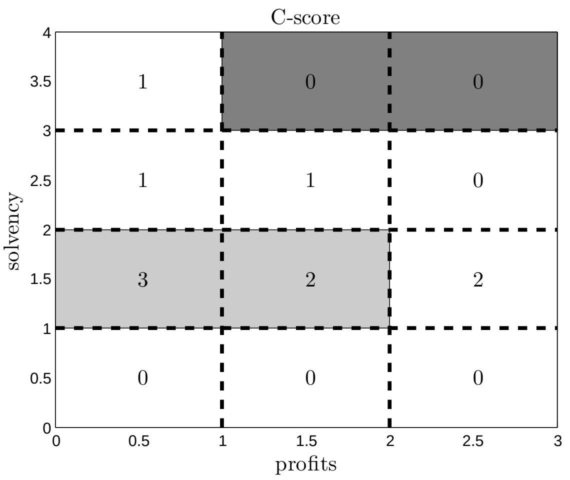

Figure 3: The C-scores of the elementary cells of the example rule set.

				Rule set $\mathcal{R}$						
Rule ID	**Order**			**Rule antecedents**				**Rule consequent**		
$r_{add,1}$	1	*if*	*profits* < 1	∧	*solvency* ≥ 3			*then*	*rating* = *A*
$r_{add,2}$	2	*if*			*solvency* ≥ 2	∧	*solvency* < 3	*then*	*rating* = *A*
$r_{add,3}$	3	*if*	*profits* ≥ 2	∧	*solvency* ≥ 1	∧	*solvency* < 2	*then*	*rating* = *B*
r_1	4	*if*	*profits* ≥ 1	∧	*solvency* ≥ 3			*then*	*rating* = *A*
r_2	5	*if*	*profits* < 2	∧	*solvency* ≥ 1	∧	*solvency* < 2	*then*	*rating* = *B*
r_3	5	*else*						*then*	*rating* = *C*

Table 2: The rule set resulting from adding complementary rules to the rule set of Table 1 to resolve the violations of the positive monotonicity constraint imposed on the attributes *profits* and *solvency*, according to the solution shown in Figure 4(a).

related additional rules to relabel these cells can be merged into a single rule, and in total only three additional rules are required to monotonize the rule set, as provided by Table 2[1]. The additional complementary rules have priority over the original rules in the rule set and are therefore assigned the highest orders in the rule set, since the additional rules have to overrule the existing labels. Consequently, the order of the original rules is decreased. The ordering among the additional rules does not matter, since by definition they cover non-intersecting subspaces of the attribute space.

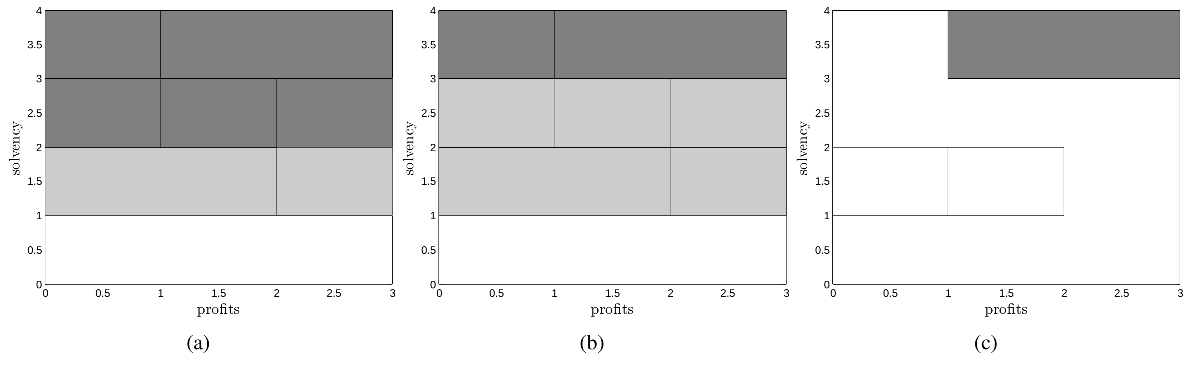

Figure 4: Adding complementary rules to resolve violations of monotonicity.

In many cases multiple solutions exist to resolve violations of monotonicity by adding complementary rules. For instance, an alternative solution to monotonize the rule set of Table 1 would be to assign class label A to cell $\mathbf{g}^4$, and class label B to cells $\mathbf{g}^3$, $\mathbf{g}^7$, $\mathbf{g}^{10}$, and $\mathbf{g}^{11}$, as shown in Figure 4(b). Figure 4(c) proposes a third possible solution to resolve the violations, i.e., by *overruling* the entire second rule of the rule set and changing its class label to C. Many more solutions could be thought of, which illustrates the need for a formal strategy to induce complementary rules. The next section will elaborate such a solution strategy, with the aim to induce the *optimal* set of additional rules.

[1]In fact, RULEM will even further merge the additional rules with the original rules, yielding a rule set of in total only two rules.

Optimal will be defined in terms of both the predictive power, which is preferably high, and the number of rules, which is preferably small in order to maintain comprehensibility.

The RULEM approach to resolve violations of monotonicity constraints by adding complementary rules can also be applied to ensure monotone classification by decision trees. A decision tree can easily be converted into a rule set, which can then be checked for violations of monotonicity by applying the RULEM algorithm. Violations can be resolved by adding rules, and the resulting rule set can be converted back into a decision tree or table [57, 58]. A decision tree induction technique is included in the experimental part of the study to illustrate this approach.

An alternative to the proposed approach of adding rules in a post-processing step would be to alter the inner workings of existing rule induction techniques, or to develop a novel rule-based classification technique that allows to incorporate monotonicity constraints during the induction of a rule set and directly yield a fully monotone rule set.

In case of a binary target variable such approaches exist and involve a rather simple adjustment to sequential covering algorithms such as AntMiner+ [36] or Ripper, i.e. a limitation of the use of the logical operator in the attribute tests for a constrained variable to either *larger than* or *smaller than*, depending on the nature of the imposed constraint (cfr. Section 6). However, in case of ordinal target variable a more profound adaptation would be required in order for a sequential covering algorithm to generate a monotone rule set. Two alternative meta-approaches could be conceived.

A first approach would be to alter the rule induction mechanism to only generate rules that respect the imposed monotonicity constraints in combination with the already induced rules, by reducing the search space of candidate rules to be added to the rule set. A second approach would be to discard rules that are not compliant with the imposed monotonicity constraints immediately after their induction and in combination with the already induced rules, before inducing a next rule. In fact, both approaches may generate identical rule sets but will significantly differ implementation-wise.

Compared to the post-processing method presented in this paper however, these approaches will likely produce suboptimal rule sets, since both approaches are unable to take into account the impact of rules that will be generated in following iterations with respect to possible violations of monotonicity constraints by a rule that is (to be) induced. In other words, an approach that incorporates constraints during rule set induction, will drastically and unnecessarily reduce the solution space of candidate rule-sets, since *good* rules in terms of classification accuracy would be omitted from the set of candidate rules to add to the rule set in each iteration.

This stems from the fact that rules may not be violating constraints in combination with rules that will be induced in subsequent iterations, and consequently before the entire rule set is induced not every rule will already be respecting the imposed constraints although they might eventually. Additionally, since the attribute space is initially unlabeled we cannot check rules (unless some approach to deal with unlabeled parts of the attribute space would be conceived). For instance, when the second rule in the example rule set of Table 1 is induced, the third rule, i.e., the default rule, has not been defined yet, and no label is assigned to cells $\mathbf{g}^1$, $\mathbf{g}^3$, and $\mathbf{g}^4$. Hence, at the moment rule r_2 is induced it essentially cannot be judged whether it is violating monotonicity constraints or not.

Therefore approaches imposing constraints before the entire rule set is induced, i.e. approaches that incorporate monotonicity constraints during the induction of a rule set and directly yield a fully monotone rule set, most likely will generate suboptimal rule sets. Hence, a sensible approach is to control and enforce monotonicity during a post-processing phase by assessing the induced rule-set as a whole, in order to induce a monotone classifier with good predictive power.

4.2. Solution strategy

The basic idea behind the solution strategy to resolve violations is that the impact of the additional rules on the workings of the original rule set should be as small as possible, in order to restrict possible negative effects on the predictive performance, and to limit the number of additional rules. As illustrated by Figure 4, multiple solutions may exist to resolve violations of monotonicity by adding rules. RULEM aims to induce the *optimal* solution; the set of additional rules that is induced to resolve the violations and to guarantee monotonicity should preserve or even improve predictive power and consist of a small number of rules. Algorithm 2 provides the pseudo-code of the solution strategy, and concerns a greedy minimization procedure similar to an approach presented by [13]. Alternative relabeling approaches as presented by [49] or [19] and originally developed to relabel data sets could be adopted, however may not scale to large data sets. Since the grid that is to be relabeled by RULEM may contain a relatively large amount of cells, the presented greedy approach is justified although further research may identify a more advanced (i.e. resulting in more optimal labels), yet scalable, relabeling procedure.. For instance, in the experiments described in Section 6 the largest grid that is to be relabeled contains more than $500,000$ cells, whereas the approaches presented by [49] and [19] have been designed to relabel data sets containing up to respectively 1000 and 100 data points. Moreover, one may remark that optimal relabeling does not guarantee an *optimal* (in terms of predictive accuracy as well as number) set of additional rules to be induced, since in a following step the relabeled cells are merged which has an impact on the final number of additional rules that is generated.

Algorithm 2 Pseudo-code of the RULEM algorithm to resolve violations of monotonicity constraints

```
1:  while C-score of rule set $\mathcal{R} > 0$ do
2:      sort cells $\mathbf{g} \in \mathcal{G}$ in descending order of C-score, yielding the set $\mathbf{g}^{s,j}$, with $j = 1$ to $n$
3:      set $j = 0$
4:      while no decrease in C-score of rule set $\mathcal{R}$ do
5:          $j = j + 1$
6:          select cell $\mathbf{g}^{s,j}$
7:          for all labels $\ell_l \in \mathcal{L}$, with $l = 1$ to $h$ do
8:              add rule $r_{\text{add}} : \mathbf{g^{s,j}} \rightarrow \ell_l$ to rule set $\mathcal{R}$
9:              calculate C-score$(\mathcal{R})^l$
10:             remove rule $r_{\text{add}}$ from $\mathcal{R}$
11:         end for
12:         select $l$, with $\arg\min_l(\text{C-score}(\mathcal{R})^l)$
13:         if C-score$(\mathcal{R})^l <$ C-score$(\mathcal{R})$ then
14:             add rule $r_{\text{add}} : \mathbf{g^{s,j}} \rightarrow \ell_l$ to rule set $\mathcal{R}$
15:         end if
16:     end while
17: end while
18: merge additional rules
19: remove redundant rules
```

The RULEM algorithm resolves violations and guarantees monotonicity by adjusting class labels of cells in the grid in a stepwise procedure until the C-score of the rule set equals zero. In each iteration of the outer while loop starting on line 1 in Algorithm 2, RULEM first calculates the C-score of each cell in the elementary grid resulting from the current total rule set, which consists of the original rules and the complementary rules that have been generated in previous iterations of this while loop, and subsequently ranks the cells in order of decreasing C-score (line 2). The underlying reasoning is to first adjust cells which appear to conflict with many other cells, since adjusting these cells will likely have the largest impact in terms of reducing the C-score. Remark that as such a greedy procedure is adopted for minimizing the C-score. The inner while loop starting on line 4 iteratively selects the cell with the highest C-score that has not been selected before during this iteration of the outer-while loop, until *adjusting* the label of a selected cell yields a reduced total C-score. The optimal relabeling of the selected cell is found by exhaustively calculating the impact of assigning all alternative class labels on the C-score of the rule set (for loop running from line 7 to 11). The optimal relabeling that results in the smallest total C-score is selected (line 12) and effectively implemented by adding a rule to the rule set if reducing the C-score (if clause on lines 13 to 15). As long as the total C-score of the rule set is larger than zero, the procedure as such continues to adjust labels of cells by adding rules. Once the total C-score of the complemented rule set equals zero, the added rules are merged to reduce the number of complementary rules (line 18). Finally, redundant rules, such as original rules that are fully overruled by additional rules, are removed from the rule set (line 19).

When this procedure is applied to the example rule set in Figure 3, in a first step the label of cell $\mathbf{g}^2$, having the highest C-score in the grid equal to three, would be changed into rating C. This will result in the C-scores of cells $\mathbf{g}^3$ and $\mathbf{g}^4$ to become equal to zero, and the C-score of cell $\mathbf{g}^{10}$ equal to one. In a second iteration the label of cell $\mathbf{g}^6$, having the highest remaining C-score equal to two, would be changed into rating C as well. This will resolve all remaining conflicts, and the C-score of the rule set becomes zero. So, solution c in Figure 4 would be generated by algorithm 2.

4.3. Refining the solution strategy

The solution strategy of the RULEM algorithm, as presented in the previous section, can be further refined in order to yield improved classification performance of the final total rule set, consisting of the original rules induced by any rule or decision tree induction technique and the additional complementary rules induced by RULEM. The definition of the C-score of a cell, which in fact controls the induction of additional rules, can be extended to take into account the predictive power of the cells it is conflicting with, leading to the definition of the I-score. The I-score is calculated similar to the C-score, but instead of counting the number of cells a cell is conflicting with, it counts the number of correctly classified instances within the cells it is conflicting with. Hence, the I-score is calculated following Algorithm 1, increasing the score of a cell by adding the number of correctly classified instances of a conflicting cell. By implementing the I-score, the RULEM algorithm will avoid to induce rules that overrule existing rules with high accuracy, thus preserving the predictive accuracy. The C-score on the other hand guides the algorithm towards a small number of additional rules, by selecting cells that conflict with many other cells.

As discussed above, besides the justifiability, both the predictive power and the comprehensibility of a rule set are two important requirements for a successful implementation of a classification model. By implementing the CI-score, which is defined as a weighted sum of the normalized C-score and the normalized I-score with a weight that can be set by the user, the RULEM algorithm aims to induce a small set of additional rules that preserves the predictive power of a rule set. The C-scores and I-scores of the cells are normalized to lie within a range between zero and one by dividing each score by the largest prevalent score. Adjusting the weights of the C- and the I-score in the formula of the CI-score allows to indicate a preference towards a smaller rule set by increasing the weight of the C score, possibly at the cost of losing predictive accuracy, or towards a rule set with good predictive accuracy by increasing the weight of the I-score, possibly at the cost of generating more additional rules. By default, the weights of both scores are set equal to 0.5.

The total CI-score of a rule set provides to some extent an indication of the justifiability of a rule set and the number of additional rules that will be required to resolve the violations. A high CI-score indicates that the rule set is strongly violating the imposed constraints, and that many additional rules are likely needed to resolve the violations. This in turn might result in poor predictive power. Therefore, in the experiments in Section 6, a maximum value for the CI-score is implemented. If a rule set yields a total CI-score larger than the maximum value, no additional rules will be generated and the rule set is deemed to be unsuitable to generate an acceptable final classification model.

When a rule set yields a high CI-score, the imposed constraints appear to be contradicted by the empirical evidence in the data and may need to be checked by the user. Hence, by calculating the CI-score, the RULEM algorithm may provide the user an indication of the reliability of the imposed constraints. However, a fully monotone data set may yield a non-monotone classification model with a high CI-score, and therefore a formal assessment of the support or evidence for a constraint in the data should make use of the degree of monotonicity as defined by Equation 3 or a similar measure. Moreover, there is no explicit or direct relation between the CI-score of a rule set on the one hand and its justifiability and the required number of additional rules on the other hand. Although unlikely, a very high CI-score may be resolved by adding a single rule to the rule set for instance. Therefore, the CI-score provides merely an indication of the justifiability of a rule set. In the next section, two formal measures based on the RULEM algorithm to calculate the justifiability of a rule set will be introduced.

5. Measuring justifiability

In this section two novel justifiability measures are defined based on the C-scores of the elementary cells in the grid as calculated by the RULEM algorithm. These measures indicate the extent to which a rule set or decision tree is in concordance with the imposed monotonicity constraints. The second section introduces a feature of the RULEM algorithm which allows to set a minimum justifiability that a rule set or decision tree needs to attain. This parameter allows to restrict the number of additional rules induced by the RULEM algorithm in case a trade-off exists between justifiability and comprehensibility.

5.1. Two novel justifiability measures: the RULEMS and RULEMF measures

The idea of measuring the justifiability of a rule or tree based classification model, i.e., the extent to which a rule set or decision tree is intuitively correct and respects domain knowledge in the form of hard monotonicity constraints, was pioneered by [40]. A metric was introduced, which requires the conversion of rule sets or decision trees into decision tables to find violations of monotonicity. The number of existing violations are then weighted to yield a measure between zero and one hundred percent. Decision tables are a tabular representation used to describe and analyze decision situations [57, 58].

The C-scores of the elementary cells in the grid as calculated by the RULEM algorithm allow to formulate two novel justifiability metrics, the rule set monotonicity Space (RULEMS) and Fraction (RULEMF) measure. These measures can be calculated in an automated manner and have a straightforward intuitive interpretation.

The RULEMS measure is calculated as one minus the fraction of the attribute space that is occupied by the elementary cells in the grid with a C-score that is different from zero, times 100%. The intuition behind this measure is that instances in these cells are not classified in line with the imposed monotonicity constraints. The lower and upper bound of a continuous numeric attribute are defined by Equations 8 and 9, which allows to calculate the fraction of the related dimension that is occupied by a rule. The fraction of a dimension related to a categorical variable that is covered by a rule is defined as the fraction of the categorical values that are covered by the rule. For instance, imagine an attribute space that consists of a single, categorical attribute with ten possible values. A rule that assigns a certain class label to instances with one specific value of this attribute then covers $1/10 = 10\%$ of the attribute space.

Let us return to the simple example rule set of Table 1 represented in Figure 1. The total size of the attribute space equals $4 \times 3 = 12$. The proportion of the attribute space covered by cells with a C-score different from zero, as indicated by Figure 3, equals $(3 \times 1 + 2 \times 1 + 1 \times 1)/12 = 6/12$. The RULEMS justifiability measure therefore equals

$(1 - 6/12) \times 100\% = 50\%$, which means that the rule set of Table 1 is 50% in line with the imposed constraint. The following equation provides a formal definition of the RULEMS measure:

$$\text{RULEMS} = (1 - \frac{\sum_{\mathcal{G}} \mathcal{V}(\mathbf{g}^{C\neq 0})}{\sum_{\mathcal{G}} \mathcal{V}(\mathbf{g})}) \times 100\%, \tag{10}$$

with $\mathbf{g}^{C\neq 0}$ the elementary cells in the grid $\mathcal{G}$ with C-score different from zero, and $\mathcal{V}(\mathbf{g})$ the volume in the attribute space occupied by an elementary cell.

The RULEMS measure has a very intuitive interpretation, i.e., the fraction of the attribute space that is in concordance with domain knowledge expressed in the form of monotonicity constraints. It suffers however from one major drawback, which is the fact that large parts of the attribute space might be empty or sparsely populated with data points or observations, and other parts more densely. Therefore, from an intuitive point of view it might be more interesting to know how many of the observations in the data set are classified monotonically by a rule set. This is the basic idea behind the RULEMF measure, which indicates the fraction of the data instances that are classified in accordance with the imposed monotonicity constraints. The RULEMF measure hence equals oone minus the fraction of instances that are covered by the elementary cells with a C-score larger than zero, times 100%:

$$\text{RULEMF} = (1 - \sum_{\mathcal{G}} Cov(\mathbf{g}^{C\neq 0})) \times 100\%, \tag{11}$$

with $Cov(\mathbf{g}^{C\neq 0})$ the coverage of the elementary cells $\mathbf{g}^{C\neq 0}$ in the grid $\mathcal{G}$ with a C-score different from zero. Both the RULEMS and RULEMF measure are independent of the selected set of additional rules, and only depend on the original rule and data set.

5.2. *Setting a minimum justifiability*

In practical applications a trade-off may exist between justifiability on the one hand side, and comprehensibility and predictive power on the other hand. Monotonizing a rule set or decision tree using RULEM, resulting in a 100% justifiable classifier, might come at the cost of a large number of additional rules and/or a large decrease in predictive power. Typically, the more rules or branches and leaves, the less comprehensible a rule set or decision tree is considered to be [40]. This might be undesirable when the model needs to be comprehensible, for instance to understand why exactly a rating is assigned by a classification model. In case the number of rules should be as small as possible to guarantee comprehensibility, or when the predictive power should be maintained at a certain level, RULEM can be tuned in order to result in a smaller number of additional rules or to yield a required minimum level of predictive power, at the cost of a reduction in justifiability.

The RULEMS and RULEMF measures introduced in the previous section can be used as parameters within RULEM to fine-tune the trade-off between justifiability, and comprehensibility or predictive power. When the user deems the decrease in predictive power or the number of additional rules added by RULEM to a rule set or a decision tree to be too large, a smaller value of the RULEMS or RULEMF justifiability measure that needs to be attained by the resulting monotone classifier can be specified. For instance, instead of demanding that a rule set is entirely monotone and yields a RULEMS or RULEMF measure equal to 100%, the user could set the required RULEMS or RULEMF measure equal to a lower value, e.g. 95%.

The RULEM algorithm will then induce in a first step additional rules following Algorithm 2, including steps 18 and 19 where rules are merged and removed when necessary, to yield a fully monotone classifier. In a second step a number of additional rules will be removed again from the rule set. Therefore, the rules are ranked according to their volume in case a RULEMS value is set, and according to their coverage in case a RULEMF value is set. Subsequently, rules are added to the rule set in order from large to small volume or coverage, until the requested justifiability is attained by the resulting rule set. A dense description of this procedure in pseudo-code is provided as Algorithm 3.

Equivalently, RULEM allows to specify a maximum number of additional rules or a minimum value for the predictive power that needs to be guaranteed by the final rule set. A similar procedure is followed as described by Algorithm 3. First, additional rules are generated to yield a fully monotone classifier. In case a maximum number of additional rules is specified, the induced additional rules are ranked by coverage or volume, depending on the preferred justifiability measure, respectively RULEMF or RULEMS. This will ensure the resulting rule set to yield maximal justifiability for the specified number of additional rules. In the while loop of the algorithm additional rules are added to the resulting rule set until the maximum number of rules is reached. In case a minimum predictive power is required, the additional rules are ranked according to both their classification accuracy and the justifiability of the resulting rule set, in order to attain maximal justifiability for the specified predictive power.

Algorithm 3 Pseudo-code of the RULEM algorithm to induce a rule set with a minimum justifiability parameter

```
1:  original rule set $\mathcal{R}_{or} = \bigvee_{e=1}^{n_{or}} r_{e,or}$
2:  apply RULEM to induce additional rules
    $\mathcal{R}_{add} = \bigvee_{e=1}^{n_{add}} r_{e,add}$, with $r_{e,add} : p_{e,add}(\mathbf{x}) \rightarrow \ell_{e,add}$
3:  **switch:** set minimal justifiability value
4:  **case:** RULEMS = $v_s$
5:      rank additional rules: $\mathcal{R}^r_{add} = \bigvee_{e=1}^{n_{add}} r^r_{e,add}$,
        with $\mathcal{V}(p^r_{e,add}) \geq \mathcal{V}(p^r_{e+1,add})$
6:      set $\mathcal{R}_{tot} = \mathcal{R}_{or}$
7:      set $i = 1$
8:      **while** RULEMS($\mathcal{R}_{tot}$) $< v_s$ **do**
9:          add rule $r^r_{i,add}$ to rule set $\mathcal{R}_{tot}$
10:         RULEMS($\mathcal{R}_{tot}$) = RULEMS($\mathcal{R}_{tot}$) + $\mathcal{V}(p^r_{e,add})$
11:         set $i = i + 1$
12:     **end while**
13: **case:** RULEMF = $v_f$
14:     rank the additional rules $\mathcal{R}^r_{add} = \bigvee_{e=1}^{n_{add}} r^r_{e,add}$,
        with $cov(p^r_{e,add}) \geq cov(p^r_{e+1,add})$
15:     set $\mathcal{R}_{tot} = \mathcal{R}_{or}$
16:     set $i = 1$
17:     **while** RULEMF($\mathcal{R}_{tot}$) $< v_f$ **do**
18:         add rule $r^r_{i,add}$ to rule set $\mathcal{R}_{tot}$
19:         RULEMF($\mathcal{R}_{tot}$) = RULEMF($\mathcal{R}_{tot}$) + $cov(p^r_{e,add})$
20:         set $i = i + 1$
21:     **end while**
22: **end switch**
```

6. Experiments

A benchmarking experiment is set up in order to assess the performance of the RULEM algorithm to induce monotone classifiers, both in terms of classification accuracy, induced number of rules, and average number of terms per rule. Section 6.1 describes the data sets and the imposed constraints, and Section 6.2 the experimental setup. Finally, in Section 6.3, the results of the experiment are discussed.

6.1. Data sets

Fourteen publicly available data sets with both monotone (positive and negative) and non-monotone attributes and an ordinal target variable have been collected. The main characteristics of these data sets are summarized in Table 3.

ID	Name	# Att.	# Obs.	# Class	Source	# Pos. C.	# Neg. C.
1	Auto	7	398	2	UCI	2	1
2	Balance	4	625	3	UCI	2	2
3	Bcl	9	286	2	UCI	3	0
4	Car	6	1728	3	UCI	6	0
5	Churn	19	5000	2	UCI	0	1
6	Contraceptive	8	1473	3	UCI	2	0
7	Era	4	1000	5	MLD	4	0
8	Esl	4	488	5	MLD	4	0
9	German	24	1000	2	UCI	3	1
10	Haberman	3	306	2	UCI	2	1
11	Housing	13	506	5	UCI	0	1
12	Lev	4	1000	5	MLD	4	0
13	Pima	8	768	2	UCI	4	0
14	Swd	10	1000	4	MLD	10	0

Table 3: The characteristics of the 14 ordinal data sets included in the benchmarking study: ID, name, number of attributes, observations, and class labels, and the source of the data sets, which is either the UCI Machine Learning Repository (archive.ics.uci.edu/ml), or the MLD Machine Learning Data Set Repository (www.mldata.org). The last two columns indicate the number of positive and negative constraints that are imposed in the experiments.

The monotonicity constraints that are imposed on the attributes in the *Auto*, *Bcl*, *German*, *Haberman*, and *Pima* data sets have been copied from [36]. The *Auto* data set concerns the prediction of car fuel consumption in gallons per mile (less or more than 28), based on eight car properties. Domain knowledge states that larger weight and displacement will lead to higher fuel consumption, where more recent models are presumed to be more fuel-efficient. In the *Bcl* data set the recurrence of breast cancer needs to be predicted, where it is expected that an increase in tumor size, number of nodes involved, and the degree of malignancy will lead to higher probability of recurrence. The target variable to predict in the *Pima* data set is whether a person shows signs of diabetes. Increasing age, number of pregnancies, body mass index, and pedigree risk would suggest a higher chance of being diabetic. In the *German* data set, the classification problem consists of predicting clients as either good or bad (defaulted). Expert knowledge suggests bad customers will have less amount on checking and savings accounts, and have had more problems with their credit history. The *Haberman* data set concerns the prediction of the survival status of a patient that has undergone breast cancer surgery. Medical knowledge suggests a lower survival rate for patients that are older, have more detected positive axillary nodes, and whose operation was less recent.

The *ERA*, *ESL*, *LEV*, and *SVD* data sets have been described in [6]. All attributes in these data sets are positively related to the target variables, and total monotonicity of the classification model is demanded. The *Balance*, *Car*, *Churn*, *Contraceptive*, and *Housing* data sets have been gathered from the UCI Machine Learning Repository. The instances in the *Balance* data set are classified as having the balance scale tip to the left, tip to the right, or being balanced. The correct way to find the class is the greater of right-distance times right-weight and left-distance times left-weight. If they are equal, an instance is classified as balanced. Hence, a negative constraint is imposed on the left-distance and the left-weight, and a positive constraint on the right-distance and right-weight, with the ordering of the class variable left, balanced, and right. The target variable to predict in the *Car* evaluation database is the acceptability of a car. A higher buying and maintenance price is expected to have a negative impact on acceptability, while the number of doors and persons the car can carry, the size of the luggage boot, and the estimated safety of the car are expected to have a positive impact. The *Churn* data set concerns the prediction of customers that are about to churn. Typically, customers that have called the helpdesk are expected to have a lower probability to churn. The problem in the *Contraceptive* data set is to predict the current contraceptive method choice (no use, short-term methods, or long-term methods) of a woman based on her demographic and socio-economic characteristics. A positive constraint is imposed on the number of children a women has given birth, and on an attribute that indicates the standard of living (low to high). Finally, the class variable in the *Housing* data set is the median value of owner occupied homes in suburbs of Boston, which is expected to be smaller in suburbs where a large fraction of the population has a lower status. A full description of the data sets can be obtained from the respective source repositories as indicated in Table 3.

6.2. Experimental setup

Three classification techniques will be applied on the selected data sets to illlustrate and assess the impact of the application of RULEM in terms of predictive power and rule set comprehensibility; Ripper [11] and AntMiner+ [37], two state-of-the-art rule induction techniques, and C4.5 [48], a decision tree induction technique. The classifiers induced with Ripper, AntMiner+, and C4.5 will be postprocessed using RULEM to check whether the imposed constraints are respected, and if not, to resolve violations by inducing additional rules. Both Ripper and C4.5 have been implemented in the Weka[2] environment [62]. AntMiner+ on the other hand has been implemented in Matlab and can be downloaded freely from the web[3].

When the target variable is binary, AntMiner+ allows to impose monotonicity constraints directly during the induction of classification rules [36]. This version of AntMiner+, denoted AntMiner+ DK (Domain Knowledge), will be applied in the benchmarking experiments on the data sets with a binary target variable. A similar feature has been developed and implemented for the Ripper algorithm, allowing to impose monotonicity constraints and to induce monotone rule sets for binary classification in a direct manner during the actual data mining process. This extended version of the Ripper algorithm will be denoted Ripper DK.

Each classification technique is applied to five random split ups of the data sets in 2/3 training and 1/3 test data. The discrimination power of the classification models will be reported as the Percentage Correctly Classified instances (PCC), since the number of values of the target attribute varies over the data sets in a range from two to five. The PCC is calculated as the fraction of observations in the test set that are assigned a correct class label by the classification model. When the output of a classifier is a score or probability, the PCC measure depends on a threshold value(s). However, the applied classification techniques result in a model that assigns an explicit class label to each instance, and not a score or probability. Furthermore, PCC can be applied regardless of the number of values of the class attribute. Therefore, in this experimental setting the PCC serves well to compare the discrimination power of the original classification models resulting from the applied techniques and the RULEM postprocessed classifiers. Additionally,

[2]www.cs.waikato.ac.nz/ml/weka
[3]www.antminerplus.com

the Mean Absolute Error (MAE) and the Mean Squared Error (MSE) will be reported for the data sets having a non-binary ordinal target variable. These measures are calculated as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{12}$$

$$\text{MAD} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{13}$$

with n the number of observations in the test set, $\hat{y}_i$ the estimated class label for observations i and y_i the true class label. Remark that for applying these measures the ordinal target variable needs to be transformed into an interval variable, e.g. $A = 1$, $B = 2$, $C = 3$. Application of these measures will provide additional insight with respect to the impact of RULEM on the accuracy in an ordinal classification context, when the total difference between the predicted and the observed labels may be important with regards to the use of the prediction. For instance in a credit scoring context, misclassification of a triple A rated obligor into the double A category has a less severe impact than misclassification into the double B category.

As mentioned in the introduction, the existing literature on predictive modeling or classification mainly focuses on the predictive power of classification models. However, it must be stressed that in a practical setting the comprehensibility and justifiability of classification models are often the determinant factors deciding upon the effective implementation and use of a model. Therefore, the number of rules in rule sets in the experiments will be reported as a measure of the comprehensibility of the induced models, as well as the average number of attribute tests in each rule. The justifiability of the original classification models on the other hand will be reported in terms of both the RULEMS and RULEMF measures, which were introduced and discussed in the previous section.

A procedure described in [17] is followed to statistically test the results of the benchmarking experiments and contrast the accuracy, the number of induced rules, and the average number of terms per rule of the different techniques. To compare two techniques, the Wilcoxon signed-ranks test [61] is applied, which ranks the differences in performances for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences. When comparing multiple techniques, in a first step the non-parametric Friedman test [22] is performed to check whether differences in performance are due to chance. If the null hypothesis that no significant differences exist is rejected by the Friedman test, then the post-hoc Nemenyi test [43] is performed to compare the individual classifiers.

6.3. Results

Table 4 summarizes the average results of the experiments over the five random holdout splits for the original and the postprocessed rule sets with RULEM. The top panel reports the mean accuracy, and the middle and bottom panel provide respectively mean absolute errors and mean squared errors for data sets with a non-binary ordinal target variable. Table 5 reports the average size of the induced rule sets in terms of mean number of rules and mean number of conjuncts per rule in the resulting rule sets, for both the original and the monotonized rule sets. Table 6 provides the mean RULEMS and RULEMF measures of the induced rule sets.

A maximum value of the CI-score equal to 50 was set in the experiments for RULEM to be executed and to induce additional rules to resolve violations of the imposed monotonicity constraints. When the initial CI-score is high, a large number of additional rules will likely be required to resolve the violations, and the resulting rule set becomes unacceptable for implementation. The maximum CI-score has been set to provide an unbiased indication of the performance of the RULEM technique, both in terms of predictive power and required number of additional rules that are induced to make a rule set monotone. It needs to be stressed that setting a minimum CI-score does not imply that RULEM is not able to induce a solution, but the quality of the induced solution is likely to be low. By setting a minimum CI-score we believe to provide a fair indication of the applicability of RULEM. In a real life setting it is the user which has to decide about the acceptability of a classification model, as well as upon the constraints to impose.

The results in Tables 4 and 5 that are marked with an asterisk are the average over less than five holdout splits. In case of RULEM, a dash indicates that for all five holdout splits an initial CI-score was obtained that was larger than 50. In case of Ripper DK and AntMiner+ DK, a dash indicates that the target variable in the data set is not binary, and consequently the classifiers could not be executed. Table 7 reports the number of times RULEM was not executed because the initial total CI-score was larger than 50.

The reason why the average number of induced rules by AntMiner+ with RULEM is smaller than without RULEM for the *Churn* and *Housing* data sets is because the result of AntMiner+ with RULEM is averaged over respectively two and one holdout splits, yielding an average number of rules for the monotonized rule sets that is smaller than the average number of rules over five hold-out splits for the original rule set.

As can be seen from Table 4, in general, the performance of the RULEM postprocessed rule sets is either slightly worse or slightly better than the performance of the original rule set in terms of PCC. RULEM yields the best classifier

	Ripper			**AntMiner+**			**C4.5**	
	OR	RULEM	DK	OR	RULEM	DK	OR	RULEM
				PCC (%)				
Auto	**89.33**	81.04	86.81	**89.47**	83.46	84.66	**90.07**	82.47*
Balance	81.32	81.32	–	77.22	**77.51**	–	**80.57**	81.23
Breast Cancer Ljubljana	71.13	71.13	71.13	71.25	**73.26***	72.50	**71.13**	70.93
Car	94.80	**94.86***	–	92.92	–	–	96.67	**97.04**
Churn	**93.88**	93.68*	91.44	88.85	88.79*	**89.24**	92.76	–
Contraceptive	51.14	**51.98**	–	42.16	**42.36***	–	52.61	–
Era	**45.06**	44.53	–	**39.34**	38.08	–	**45.18**	44.71
Esl	70.84	70.84	–	48.83	–	–	70.00	70.00
German Credit Scoring	**73.65**	71.65	72.18	70.06	67.96*	**70.24**	70.94	–
Haberman	75.00	75.00	73.46	71.76	72.75	72.75	72.69	72.69
Lev	59.53	**60.41**	–	**47.66**	45.63	–	60.76	**60.88**
Pima	74.94	74.94	74.41	71.17	70.70*	**71.33**	73.95	**74.71***
Swd	**56.24**	56.18*	–	**46.71**	39.82*	–	**55.47**	55.29*
Housing	71.74	71.74	–	**58.34**	56.80*	–	70.93	–
				MAE				
Balance	0.30	0.30	–	0.37	0.38	–	0.29	0.28
Car	0.06	0.08*	–	0.09	–	–	0.04	0.03
Contraceptive	0.76	0.76	–	0.91	0.91*	–	0.69	–
Era	0.77	0.78	–	0.84	0.88	–	0.77	0.78
Esl	0.30	0.32	–	0.75	–	–	0.30	0.30
Lev	0.45	0.45	–	0.60	0.61	–	0.44	0.44
Pima	0.25	0.25	–	0.29	0.30*	–	0.26	0.25*
Housing	0.33	0.33	–	0.57	0.60*	–	0.31	–
				MSE				
Balance	0.53	0.53	–	0.66	0.67	–	0.48	0.46
Car	0.08	0.11*	–	0.12	–	–	0.05	0.04
Contraceptive	1.30	1.30	–	1.59	1.57*	–	1.12	–
Era	1.24	1.26	–	1.32	1.42	–	1.29	1.32
Esl	0.34	0.35	–	1.36	–	–	0.31	0.31
Lev	0.54	0.55	–	0.75	0.75	–	0.57	0.55
Pima	0.25	0.25	–	0.29	0.30*	–	0.26	0.25*
Housing	0.45	0.45	–	0.95	1.06*	–	0.35	–

Table 4: Results of the experiments in terms of percentage correctly classified instances, mean absolute error and mean squared error. A result marked with an asterisk is averaged over less than five hold out splits, as indicated in Table 7

	Ripper			**AntMiner+**			**C4.5**	
	OR	RULEM	DK	OR	RULEM	DK	OR	RULEM
# Rules								
Auto	4.4	13.6	3.2	4.0	15.0	4.4	10.4	30.3*
Balance	9.0	9.0	–	9.4	17.6	–	33.0	34.8
Breast Cancer Ljubljana	2.0	2.0	2.0	4.2	5.0*	4.2	8.8	14.0
Car	17.4	39.3*	–	20.4	–	–	44.8	61.2
Churn	8.0	20.8*	5.6	5.8	4.5*	6.0	30.6	–
Contraceptive	4.0	8.4	–	6.4	27.3*	–	94.8	–
Era	4.4	6.0	–	6.6	25.6	–	10.6	18.4
Esl	10.4	16.8	–	15.6	–	–	22.0	33.6
German Credit Scoring	4.4	16.6	2.8	3.2	11.3*	4.2	37.2	–
Haberman	2.0	2.0	2.0	4.0	10.4	4.4	3.8	4.0
Lev	9.6	26.6	–	8.0	22.2	–	24.4	46.2
Pima	3.4	3.4	–	4.2	9.3*	–	16.2	36.3*
Swd	8.4	17.7*	3.2	11.6	33.0*	7.6	17.2	60.5*
Housing	7.8	7.8	–	7.8	6.0*	–	29.8	–
# Tests/Rule								
Auto	2.1	3.3	1.5	2.4	4.3	2.0	2.9	3.3*
Balance	2.8	2.8	–	3.4	3.6	–	3.6	3.3
Breast Cancer Ljubljana	1.8	1.8	1.8	3.5	3.9*	3.8	3.1	3.4
Car	4.7	5.2*	–	5.1	–	–	4.1	3.9
Churn	2.6	5.7*	2.6	3.2	3.2*	3.2	5.0	–
Contraceptive	3.1	3.6	–	4.8	6.4*	–	4.6	–
Era	2.9	3.1	–	3.4	3.8	–	2.7	2.8
Esl	1.7	1.9	–	3.4	–	–	3.1	3.2
German Credit Scoring	2.9	4.0	2.9	3.3	3.8*	5.0	5.1	–
Haberman	1.8	1.8	1.6	2.7	2.9	2.8	1.9	1.3
Lev	2.9	3.4	–	3.6	3.8	–	3.2	3.5
Pima	2.0	2.0	–	3.8	5.0*	–	3.7	4.1*
Swd	2.9	4.7*	2.0	5.6	7.8*	3.7	3.3	5.6*
Housing	1.9	1.9	–	5.7	6.9*	–	4.3	–

Table 5: Results of the experiments in terms of average number of rules and conjuncts per rule over the hold-out splits.

	Ripper		**AntMiner+**		**C4.5**	
	RULEMF	RULEMS	RULEMF	RULEMS	RULEMF	RULEMS
Auto	34.44	40.10	74.43	60.25	25.57	32.66
Balance	100.00	100.00	98.91	97.18	99.53	98.98
Breast Cancer Ljubljana	100.00	100.00	100.00	94.52	81.92	97.69
Car	100.00	97.87	–	–	96.85	97.40
Churn	92.48	98.34	100.00	100.00	–	–
Contraceptive	84.56	90.45	100.00	97.74	–	–
Era	99.69	97.64	98.75	65.44	84.84	84.94
Esl	94.06	96.23	–	–	87.66	94.26
German Credit Scoring	94.33	65.76	72.20	57.43	–	–
Haberman	100.00	100.00	79.04	74.84	95.52	99.80
Lev	97.73	93.78	100.00	73.76	81.17	84.06
Pima	100.00	100.00	100.00	96.27	97.71	94.62
Swd	100.00	97.77	100.00	89.80	81.25	82.10
Housing	100.00	100.00	100.00	99.80	–	–

Table 6: Justifiability measures of the original, without monotoniciy constraints, rule sets induced by Ripper, AntMiner+, and C4.5 classifiers.

Name	Ripper	AntMiner+	C4.5
Auto	0	0	2
Balance	0	0	0
Breast Cancer Ljubljana	0	2	0
Car	1	5	0
Churn	1	1	5
Contraceptive	0	2	5
Era	0	0	0
Esl	0	5	0
German Credit Scoring	0	1	5
Haberman	0	0	0
Lev	0	0	0
Pima	0	2	2
Swd	2	4	3
Housing	0	3	5

Table 7: The number of hold out splits resulting in a CI-score of the rule set or decision tree above the threshold value. These runs have not been included in calculating the values in Table 4.

in 11 cases, and the original classifier in 17 cases. In eight cases the monotonicity is respected by the original rule set for all data points in the test set, resulting in equal performance. The Wilcoxon signed-ranks test to compare the predictive power of RULEM vs. the original rule set was not able to reject the null hypothesis that RULEM does not affect the predictive power of a rule set in a statistically significant manner ($p \cong 0.14$). When comparing the RULEM postprocessed rule sets to the rule sets induced using Ripper DK and AntMiner+ DK on the data sets with a binary target variable, the same conclusion holds; the predictive power in terms of PCC of the RULEM postprocessed rule sets is found not to be significantly different from the predictive power of the rule sets obtained using the DK versions ($p \cong 0.23$). The Mean Absolute and Squared Error performance measures for the original and postprocessed rule sets are only marginally different as can be seen from the lower two panels of Table 4. This confirms the above finding based on the assessment of the PCC measure that the postprocessed rule sets perform similar when compared to to the original rule set.

The size of the induced rule sets, both in terms of number of rules ($p < 0.01$) and average number of attribute tests per rule ($p = 0.075$) appears to be significantly larger when applying the RULEM technique. Since RULEM induces additional rules to resolve monotonicity constraints, this is not surprising. However, the comprehensibility of the RULEM postprocessed rule sets appears to be maintained, given the overall relatively small total number of rules and tests per rule of these rule sets. Converting a C4.5 decision tree into a rule set generally appears to yield a quite large number of rules. Consequently, RULEM also requires a relatively large number of rules to resolve violations and enforce monotonicity.

Table 6 reports the justifiability of the induced rule sets both in terms of the RULEMS and RULEMF measure, as defined in the previous section. For each data set in the experiment either Ripper or AntMiner+, and in most cases even both, results in a rule set that does not respect the imposed monotonicity constraints. C4.5 yields a non-monotone classification model for each data set. This result indicates that in general rule and tree induction techniques do not yield monotone classification models, and that there is a need for techniques which are able to enforce monotone relations. Remark in Table 6 that the RULEMF measure may be equal to one hundred percent while the RULEMS measure is not, for instance on the Car and swd data sets for the combination RULEM and Ripper. When no instances in the data set are situated in the part of the attribute space where classification does not respect the imposed constraints, the RULEMF measure will be equal to one hundred percent, while the RULEMS measure clearly will not.

The results of the experiments indicate that the RULEM technique does not outperform the DK versions of the classifiers, however, the DK versions can only be applied with a binary target variable, and not every rule or tree induction technique has a DK version in place allowing to impose monotonicity constraints in a direct manner. This exactly illustrates the main strength of the RULEM approach, which lies in the fact that RULEM can be combined with any rule or tree induction technique and does not require adjustments to the workings of these techniques. As shown by the experiments, RULEM preserves the predictive power of the original classification techniques. As such, the resulting predictive power mainly depends on the performance of the base rule induction technique RULEM is combined with, opening opportunities to further improve predictive power, for instance by applying active learning strategies [39].

A final remark regarding the impact of RULEM on the predictive performance of a rule set concerns the definition of the imposed constraints. When this impact is large, both in terms of predictive accuracy and number of rules, the constraints that are imposed on the classification model might need to be reconsidered since they appear not to be in line with the data. The domain knowledge or intuitive relations that result in the imposed constraints always has to be

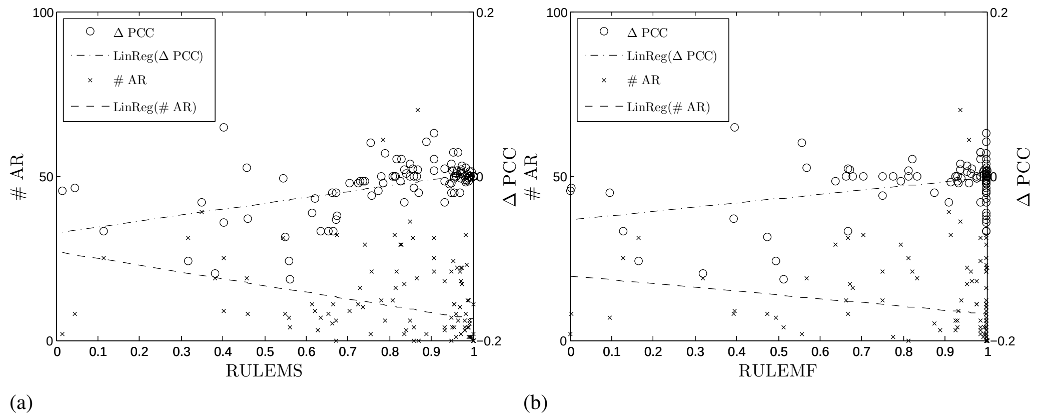

Figure 5: Number of additional rules (# AR) and difference in percentage correctly classified between the RULEM postprocessed and the original rule set (Δ PCC), as a function of the justifiability of a rule set measured using the RULEMS (upper panel) and RULEMF measure (lower panel), and simple linear regression models fitting # AR and Δ PCC as a function of RULEMS and RULEMF.

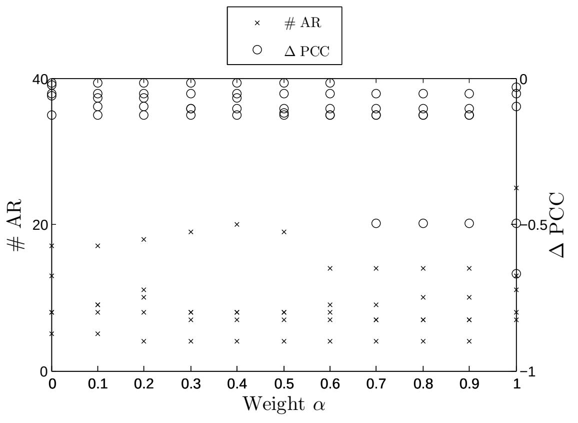

Figure 6: The number of additional rules (# AR) and the difference in accuracy (Δ PCC) as a function of the weight parameter α in the CI-score ($CI = \alpha C + (1 - \alpha)I$), for data set Auto, rule induction technique Ripper, and α ranging between zero and one.

questioned before effectively imposing constraints on the resulting model.

Figure 5 plots the number of additional rules (# AR) and difference in percentage correctly classified between the RULEM postprocessed and the original rule set (Δ PCC), as a function of the justifiability of a rule set measured using the RULEMS and RULEMF measure. The figures also plot simple linear regression models, fitting the number of additional rules and the difference in accuracy as a function of the justifiability. The inclination of these linear regression models indicates that the number of additional rules decreases as a function of the justifiability ($R^2 = 0.19$ and $R^2 = 0.33$ for the linear regression models fitting the number of additional rules as a function of respectively RULEMS and RULEMF), and that the difference in accuracy between the original model and the RULEM postprocessed rule set decreases as well ($R^2 = 0.41$ and $R^2 = 0.54$ for the linear regression models fitting ΔPCC as a function of respectively RULEMS and RULEMF). The plots in Figure 5 indicate that the stronger a rule set violates the imposed monotonicity constraints, the lower the predictive power of the RULEM postprocessed rule set will be, and the more additional rules will be required to resolve the violations.

Figure 6 plots the number of additional rules and the difference in accuracy between the RULEM postprocessed rule set and the original rule set as a function of the weight parameter α in the CI-score, with $CI = \alpha C + (1 - \alpha)I$, for each of the five holdout splits of the *Auto* data set, and with the original rule sets induced by Ripper. The weight α was varied between zero and one. The figure indicates that for a smaller value of the weight RULEM in general results in a more accurate rule set and a larger number of additional rules, while for a larger value of α less additional rules are induced at the cost of a decrease in predictive power compared to the original rule set. As explained in Section 4.2, the I-score was designed to preserve the predictive power of the original rule set, while the C-score aims at inducing a small number of additional rules. The results of Figure 6 confirm that by varying the weight in the CI-score, RULEM effectively allows to a certain extent to express a preference towards a smaller rule set or towards a rule set with better predictive power. In the benchmarking experiment, the weight was set to 0.5, which incorporates a trade-off between both comprehensibility and accuracy.

Finally, Figure 7 plots the number of additional rules induced by RULEM as a function of the initial CI-score

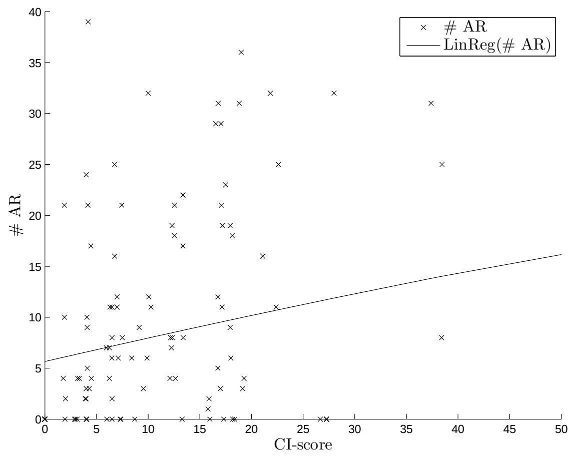

Figure 7: The number of additional rules (# AR) as a function of the CI-score, and a third order polynomial fitting the data points.

of the original rule sets. The figure also plots a third order[4] polynomial fitting the number of additional rules as a function of the initial CI-score, with $R^2 = 0.32$. The inclination of the polynomial clearly indicates that the number of additional rules is a positive function of the initial CI-score, and therefore justifies the maximum threshold value of the CI-score that was set in the experiments.

7. Conclusions and future research

Many real world applications require classification models to be in line with domain knowledge and to satisfy monotone relations between predictor variables and the target class, in order to be acceptable for implementation. Therefore, this paper presents the novel RULEM algorithm to induce monotone ordinal rule based classification models. A main asset of the proposed approach is its complementarity, which allows the RULEM approach to be combined with any rule- or tree-based classification technique, since monotonicity is guaranteed during a post-processing step. The RULEM algorithm checks whether a rule set or decision tree violates the imposed monotonicity constraints, and existing violations are resolved by inducing a set of additional rules which enforce monotone classification. The algorithm is able to handle non-monotonic noise, and can be applied to both partially and totally monotone problems with an ordinal target variable.

Based on the RULEM algorithm, two novel justifiability measures are introduced. The RULEMS and RULEMF measures allow to calculate the extent to which a classification model is in line with domain knowledge expressed in the form of monotonicity constraints. Both measures provide an intuitive indication of the justifiability of a rule set, and can be calculated in a fully automated manner.

An extensive benchmarking experiment has been set up to test the impact of the RULEM approach on the predictive power and the comprehensibility of the resulting rule set. The results of the experiments indicate that the proposed approach preserves the predictive power of the original rule induction techniques while guaranteeing monotone classification, at the cost of a small increase in the size of the rule set. Hence, the RULEM algorithm is shown to yield accurate, comprehensible, and justifiable rule based classification models. The predictive power of the final rule set therefore depends on the selected rule induction technique that RULEM is combined with.

An important topic for future research is the further development and refinement of the RULEM algorithm, and more precisely the improvement of the heuristic approach to merge the induced additional rules. This will allow to further reduce the size of the final rule set. Moreover, further analysis and experiments are needed to examine the exact nature of the relation between the C(I)-score, the justifiability, and the induced number of additional rules. Finally, an interesting and challenging topic for future research will be the development of a justifiability measure for non-rule based classification models.

8. Appendix

Table 8 below lists the mathematical symbols used throughout the paper and their meaning.

[4]A first and second order polynomial yield the same conclusion, but result in a value of $R^2 < 0.20$.

Symbol	Meaning
$\mathcal{X}$	Input space
$\mathcal{X}_i$	Attribute dimension
i	Attribute index
m	Constrained attribute index
k	Number of attributes
$\mathbf{x}$	Attribute vector
$\mathcal{L}$	Ordered set of labels
l	Label index
h	Number of labels
f	True classification function
$\hat{f}$	Approximated classification function
D	Data set
a	Observation index
o	Number of observations
DgrMon	Degree of monotonicity
$\mathcal{R}$	Rule set
e	Rule index
q	Number of rules
r	Classification rule or disjunct
op	Logical operator
v	Value
p	Rule precondition or antecedent
c	Conjunct or attribute test
$\mathcal{G}$	Grid
$\mathbf{g}$	Elementary cell in grid
u	Cell index
n	Number of cells
$g_{i,j}$	Elementary interval
j	Elementary interval index
n_i	Number of intervals constituting attribute dimension $\mathcal{X}_i$
ε	Elementary value
s	Ordered set of attribute values or cells
$\mathcal{V}(\mathbf{g})$	The volume in the attribute space of an elementary cell $\mathbf{g}$
# AR	Number of additional rules generated by RULEM
α	Weight parameter of the CI-score

Table 8: Overview of mathematical notations.

References

[1] E. Altendorf, E. Restificar, and T.G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, Edinburgh, U.K.*, 2005.

[2] Irit Askira-Gelman. Knowledge discovery: Comprehensibility of the results. In *HICSS '98: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 5*, page 247, Washington, DC, USA, 1998. IEEE Computer Society.

[3] Bart Baesens, Veronique Van Vlasselaer, and Wouter Verbeke. *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*. John Wiley & Sons, 2015.

[4] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1):29–43, 1995.

[5] A. Ben-David, L. Sterling, and Y.H. Pao. Learning and classification of monotonic ordinal concepts. *Computational Intelligence*, 5(1):45–49, 1989.

[6] A. Ben-David, L. Sterling, and T. Tran. Adding monotonicity to learning algorithms may impair their accuracy. *Expert Systems with Applications*, 36(3):6627–6634, 2009.

[7] K. Berteloot, W. Verbeke, G. Castermans, T. Van Gestel, D. Martens, and B. Baesens. A novel credit rating migration modeling approach using macroeconomic indicators. *Journal of Forecasting*, 32(7):654–672, 2013.

[8] Cristián Bravo and Richard Weber. Semi-supervised constrained clustering with cluster outlier filtering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 347–354. Springer Berlin Heidelberg, 2011.

[9] K. Cao-Van and B. De Baets. Growing decision trees in an ordinal setting. *International journal of intelligent systems*, 18(7):733–750, 2003.

[10] R. Chandrasekaran, Y. U. Ryu, V.S. Jacob, and S. Hong. Isotonic separation. *INFORMS Journal on Computing*, 17:462–474, 2005.

[11] W.W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning, ICML*, pages 115–123, 1995.

[12] H. Daniels and B. Kamp. Application of MLP networks to bond rating and house pricing. In *Neural Computing and Applications*, volume 8, pages 226–234. August 1999.

[13] H. Daniels and M. Velikova. Derivation of monotone decision models from noisy data. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 36(5):705–710, 2006.

[14] H. Daniels and M. Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

[15] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens. Data mining techniques for software effort estimation: a comparative study. *IEEE Transactions on Software Engineering*, 38(2):375–397, 2012.

[16] Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. Learning rule ensembles for ordinal classification with monotonicity constraints. *Fundamenta Informaticae*, 94(2):163–178, 2009.

[17] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[18] A. Fallah Tehrani, W. Cheng, K. Dembczynski, and E. Hullermeier. Learning monotone nonlinear models using the choquet integral. *Machine Learning*, 89(1-2):183–211, 2012.

[19] A. Feelders. Monotone relabeling in ordinal classification. In *Proceedings of the 10th IEEE International Conference on Data Mining ICDM10*, 2010.

[20] A.J. Feelders. Prior knowledge in economic applications of data mining. In *Proceedings of the 4th European conference on principles and practice of knowledge discovery in data bases*, volume 1910 of *Lecture Notes in Computer Science*, pages 395–400, New York, NY, U.S.A., 2000. Springer.

[21] A.J. Feelders and M. Pardoel. Pruning for monotone classification trees. In *Lecture Notes in Computer Science*, volume 2810, pages 1–12, New York, NY, U.S.A., 2003. Springer.

[22] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940.

[23] Javier García, Adnan M AlBar, Naif R Aljohani, José-Ramón Cano, and Salvador García. Hyperrectangles selection for monotonic classification by using evolutionary algorithms. *International Journal of Computational Intelligence Systems*, 9(1):184–201, 2016.

[24] Sergio González, Francisco Herrera, and Salvador García. Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing*, 33(4):367–388, 2015.

[25] S. Greco, B. Matarazzo, and R. Slowinski. Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research*, 129:1–47, 2001.

[26] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. Springer, New York, NY, U.S.A., 2001.

[27] P. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. Fast and effective retrieval of medical tumor shapes. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):889–904, 1998.

[28] W. Kotlowski, K. Dembczynsky, S. Greco, and R. Slowinski. Stochastic dominance-based rough set model for ordinal classification. *Information Sciences*, 178:4019–4037, 2008.

[29] W. Kotlowski and R. Slowinski. Rule learning with monotonicity constraints. In *Proceedings of the 26th Annual International Conference on Machine Learning ICML'09*, pages 537–544, 2009.

[30] S. Kramer, N. Lavrač, and P. Flach. Relational data mining. chapter Propositionalization approaches to relational data mining, pages 262–286. Kluwer, Berlin, Germany, 2001.

[31] B. Lang. Monotonic multi-layer perceptron networks as universal approximators. *Lecture Notes in Computer Science*, 3697:31–37, 2005.

[32] Sheng-Tun Li and Chih-Chuan Chen. A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge. *Fuzzy Systems, IEEE Transactions on*, 23(5):1713–1727, 2015.

[33] S. Lievens and B. De Baets. Supervised ranking in the weka environment. *Information Sciences*, 180:4763–4771, 2010.

[34] Kazuhisa Makino and Toshihide Ibaraki. Data analysis by positive decision trees. *IEICE Transactions on Information and Systems*, 82(1):76–88, 1999.

[35] D. Martens, L. Bruynseels, B. Baesens, M. Willekens, and J. Vanthienen. Predicting going concern opinion with data mining. *Decision Support Systems*, 45:765–777, 2008.

[36] D. Martens, M. De Backer, R. Haesen, B. Baesens, C. Mues, and J. Vanthienen. Ant-based approach to the knowledge fusion problem. In *Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence*, Lecture Notes in Computer Science, pages 85–96, New York, NY, U.S.A., 2006. Springer.

[37] D. Martens, M. De Backer, R. Haesen, M. Snoeck, J. Vanthienen, and B. Baesens. Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation*, 11(5):651–665, 2007.

[38] D. Martens and F. Provost. Explaining data-driven document classifications. 2013.

[39] D. Martens, T. Van Gestel, and B. Baesens. Decompositional rule extraction from support vector machines by active learning. *IEEE Transactions on Knowledge and Data Engineering*, 21(2):178–191, 2009.

[40] D. Martens, J. Vanthienen, W. Verbeke, and B. Baesens. Performance of classification models from a user perspective. *Decision Support Systems*, 51:782–793, 2011.

[41] I. Milstein, A. Ben David, and R. Potharst. Generating noisy monotone ordinal datasets. *Artificial Intelligence Research*, 3(1):30–37, 2014.

[42] Irena Milstein, Arie Ben David, and Rob Potharst. About the sensitivity of ordinal classifiers to non-monotone noise. *Artificial Intelligence Research*, 4(2):p83, 2015.

[43] P.B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

[44] V.N. Popova. *Knowledge discovery and monotonicity*. PhD thesis, Rotterdam School of Economics, Erasmus University, Rotterdam, The Netherlands, 2004.

[45] R. Potharst and A.J. Feelders. Classification trees for problems with monotonicity constraints. *SIGKDD Explorations*, 4(1):1–10, 2002.

[46] Foster Provost and Tom Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media, Inc., 2013.

[47] Yuhua Qian, Hang Xu, Jiye Liang, Bing Liu, and Jieting Wang. Fusing monotonic decision trees. *Knowledge and Data Engineering, IEEE Transactions on*, 27(10):2717–2728, 2015.

[48] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, U.S.A., 1993.

[49] Michaël Rademaker, Bernard De Baets, and Hans De Meyer. Loss optimal monotone relabeling of noisy multi-criteria data sets. *Information Sciences*, 179(24):4089–4096, 2009.

[50] Michaël Rademaker, Bernard De Baets, and Hans De Meyer. Optimal monotone relabelling of partially non-monotone ordinal data. *Optimization Methods and Software*, 27(1):17–31, 2012.

[51] J. Sill. Monotonic networks. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, Cambridge, MA, U.S.A., 1998. MIT Press.

[52] MM Stenina, MP Kuznetsov, and VV Strijov. Ordinal classification using pareto fronts. *Expert Systems with Applications*, 42(14):5947–5953, 2015.

[53] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, Boston, MA, U.S.A., 2006.

[54] R. Van de Kamp, A. Feelders, and N. Barile. Isotonic classification trees. In *Advances in Intelligent Data Analysis VIII IDA'09*, pages 405–416. Springer, 2009.

[55] T. Van Gestel, D. Martens, B. Baesens, D. Feremans, J. Huysmans, and J. Vanthienen. Forecasting and analyzing insurance companies' ratings. *International Journal of Forecasting*, 23(3):513–529, 2007.

[56] J. Van Gool, W. Verbeke, P. Sercu, and B. Baesens. Credit scoring for microfinance: Is it worth it? *International Journal of Finance & Economics*, 17(2):103–123, 2012.

[57] J. Vanthienen, C. Mues, and A. Aerts. An illustration of verification and validation in the modelling phase of KBS development. *Data and*

Knowledge Engineering, 27(3):337–352, 1998.
[58] J. Vanthienen, C. Mues, G. Wets, and K. Delaere. A tool-supported approach to inter-tabular verification. *Expert Systems with Applications*, 15(3-4):277–285, 1998.
[59] W. Verbeke, K. Dejaeger, D. Martens, J. Hur, and B. Baesens. New insights into churn prediction in the telecommunication sector: a profit driven data mining approach. *European Journal of Operational Research*, 218(1):211–229, 2012.
[60] W. Verbeke, D. Martens, C. Mues, and B. Baesens. Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications*, 38(3):2354–2364, 2011.
[61] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.
[62] I.H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, U.S.A., 2000.