Rank Discrimination Measures for Enforcing Monotonicity in Decision Tree Induction

Christophe Marsala[a], Davide Petturiti[b,c,*]

[a] *LIP6, Université Pierre et Marie Curie, Paris 6, 4 place Jussieu, 75005 Paris, France*
[b] *Dip. Matematica e Informatica, Università degli Studi di Perugia, via Vanvitelli 1, 06123 Perugia, Italy*
[c] *Dip. S.B.A.I., "La Sapienza" Università di Roma, via Scarpa 16, 06123 Roma, Italy*

Abstract

Monotone classification is a relatively recent topic in machine learning in which the classification function to learn is asked to guarantee a sort of monotonicity of the class with respect to attribute values. Nevertheless, real datasets are quite far from being monotone and this can sharply limit the performance of purely monotone classifiers while standard classifiers are simply insensitive to monotonicity. Here we focus on rank discrimination measures to be used in decision tree induction, i.e., functions able to measure the discrimination power of an attribute with respect to the class taking into account the monotonicity of the class with respect to the attribute. Three new measures are studied in detail and a hierarchical construction model is derived allowing the formal definition of a general rank discrimination measure. Our measures have been compared with other well-known proposals, quantifying both the accuracy and the monotonicity of the resulting binary decision tree classifiers.

Keywords: rank discrimination measure, monotone classification, monotone binary decision tree classifier

1. Introduction

Economy, social sciences and medicine are natural fields for decision tasks involving a set of objects $\Omega = \{\omega_1, \ldots, \omega_n\}$ described by attributes a_j's ranging in a totally ordered set X_j, and labelled with a class coming also from a totally ordered set C. Indeed, the introduction of order structures increases the expressive power of the decision model allowing the representation of semantic concepts such as preference, priority, importance and so on. Prototypical examples are monotone price models [35], customer satisfaction analysis [16] and medical diagnosis [24, 36].

*Corresponding author.
Email addresses: christophe.marsala@lip6.fr (Christophe Marsala), davide.petturiti@dmi.unipg.it (Davide Petturiti)

Evidently, these problems are quite similar to a standard classification task [6, 31] in the sense that a classification function must be learned. Nevertheless, in this context an additional monotonicity requirement is imposed on the classifier, expressing the intuitive idea that *objects with better attribute values should not be labelled with worse decision class*.

This problem has been studied in different communities such as machine learning, statistics and multi-criteria decision aid and thus there exist several names for it, like monotone or ordinal classification, ranking or sorting, and multi-criteria classification. In this paper we will adopt the name of monotone classification.

More formally, denoting with X the description space generated by the X_j's, the *monotone classification problem* (see, e.g., [30]) consists in finding a monotone extension $\lambda' : X \to C$, of a monotone consistent labelling function $\lambda : E \to C$ defined on a set of *examples* $E \subseteq X$. Anyway, real data are generally neither monotone consistent nor consistent, i.e., λ could be non-monotone on E or worse λ could be just a relation on $E \times C$.

Hence, to face this problem two options are available:

1. Require monotone consistency and possibly modify the dataset;
2. Do not require any assumption on the dataset and work with all the available information.

In the first option, techniques for dataset monotonization (see, e.g., [12, 21, 30]) must be adopted such as delete all non-monotone examples or perform class relabelling. The result of this approach is generally a classifier which although monotone could be even very far from the information content of the initial dataset. On the other hand, the second option requires to extract monotone structures from data (if present) and this implies a deeper investigation on the given dataset. Evidently, this second approach does not produce a monotone classifier in general.

In our opinion, the second possibility is more preferable since it does not require to remove data or add artificial information, hence in this paper no particular hypothesis is required on the considered datasets. At the same time, we think that a monotone classifier should exploit the presence of monotonicity in the data in order to increase his prediction power and achieve in this way a better understanding of data structure.

Despite its highly applicative nature, monotone classification has gathered only few attention in the past years, being almost obscured by the ordinary non-monotone classification. In the literature, some monotone classifiers have been proposed such as the Ordinal Learning Model (OLM) [2, 3] and the Ordinal Stochastic Dominance Learner (OSDL) [8, 9, 22] but in [4] it is shown the existing monotone classifiers do not possess any statistical advantage w.r.t. non-monotone classifiers in terms of classification accuracy. In the same paper, the authors conjecture the tested monotone classifiers are deeply influenced by non-monotone noise present in real data.

In this paper we consider decision tree classifiers which are known to be particularly understandable by field experts. Classically, a decision tree is built

with a Top Down Induction of Decision Tree (TDIDT) algorithm [6, 31], that is an algorithm building the tree from the root to the leaves by repeated partitioning of the dataset. The main distinguishing feature of existing algorithms is the *discrimination measure* [24, 25, 26] used to select at each step the attribute for splitting. Well-known choices are the conditional Shannon entropy [1, 31], the Gini discrepancy index [6] and the Yuan and Shaw measure of ambiguity (originally proposed to build fuzzy decision trees) [39].

More precisely, our aim is to inductively build a decision tree exploiting somehow the eventual monotonicity present in the dataset, however, since no assumption of monotonicity is made on the data, we need to relax requirements. Indeed, the global monotonicity constraint acts on the final classifier λ' and so it requires an a priori knowledge of the entire tree. Hence, global monotonicity is difficult to enforce in an inductive construction procedure since at each step only one attribute can be taken into account. This is why we adopt a greedy approach: at each step of the building process we choose the attribute a_j "enforcing the most" the *local monotonicity constraint*, i.e., for all $\omega_i, \omega_h \in \Omega$,

$$a_j(\omega_i) \leq a_j(\omega_h) \;\Rightarrow\; \lambda(\omega_i) \leq \lambda(\omega_h),$$

where $a_j(\omega_i)$ and $\lambda(\omega_i)$ are the values of the attribute a_j and the labelling function λ on the object ω_i. As a consequence, we cannot expect a globally monotone classifier at the end of the procedure.

In order to achieve this greedy construction of monotone decision trees we need discrimination measures sensitive to monotonicity, indeed, common discrimination measures do not possess this property [3]. With this aim, we search for functions able to quantify the monotonicity of λ w.r.t. a_j and being robust to non-monotone noise.

In [18] the authors propose a rank generalization of Shannon mutual information, namely *rank mutual information*, which is a combination of Shannon entropy with *dominance rough set* relation [14, 15], based on the object-wise writing of Shannon entropy. In the same paper they underline that this measure is both sensitive to monotonicity and robust to noisy data. In [17] this measure is used to build binary tree classifiers guaranteed to possess a weak form of monotonicity in the case the starting dataset is monotone consistent. They call this TDIDT algorithm REMT and show it behaves well compared to both monotone and non-monotone classifiers. Moreover, in [19] they use the rank mutual information for feature selection.

Here, we apply the same rank generalization procedure given in [18] to other two deeply studied discrimination measures such as Gini measure and Yuan and Shaw measure, moreover, we directly introduce a third measure not having a non-monotone counterpart. Since all the introduced measures share a common functional structure, a *hierarchical construction model* for rank discrimination measures (see [27]) in the spirit of [24, 25, 26] has been proposed and a formal definition of *rank discrimination measure* has been given. As a side effect, the definition of a hierarchical construction model is also important since it provides a base for creating new measures.

In order to show the effectiveness of the proposed measures we wrote a binary tree classifier parametrized by a discrimination measure H used for splitting. This TDIDT algorithm, called RDMT(H), is essentially based on REMT [17] and is written in Java using the WEKA package. RDMT(H) does not produce globally monotone classifiers in general, nevertheless, it guarantees a weak form of monotonicity, namely *rule monotonicity*, in case the dataset is monotone consistent and the measure H is a rank discrimination measure.

Since the main goal of present paper is the study of rank discrimination measures and the resulting splitting criteria, RDMT(H) has been tested on artificial and real datasets varying H among the given rank discrimination measures and other proposals present in the literature (in detail, the conditional Shannon entropy [31], the conditional Gini index [6], the total ambiguity score used in MID algorithm [2] and the conditional Gini_{L_1} impurity used in ICT algorithm [34]).

Our study focused on the impact of the sole discrimination measure on the final classifier in terms of classification accuracy and monotonicity, testing all the measures under the same experimental conditions.

We firstly investigated the effect of non-monotone noise on the considered performance indices for the different measures, generating artificial datasets with an increasing degree of non-monotone noise [28]. In these tests no significant difference on classification accuracy has been detected between the tested measures, while an evident better response of our measures in terms of monotonicity has been noticed, especially for increasing degree of non-monotone noise.

We then compared the measures on 15 real datasets and the obtained results have been validated through the Friedman test and the post-hoc Nemenyi test [11]. Our analysis on real datasets highlighted no statistically significant difference among the tested measures for what concerns classification accuracy. On the converse, from the same tests we detected significantly better results of our measures in terms of monotonicity.

In all the performed tests, our measures showed a tendency toward trees with a higher number of leaves in which the number of pairwise non-monotone label comparisons is minimized. In other terms, our measures give generally rise to a finer partition of the description space in which the non-monotone labellings are minimized.

This fact suggests a possible advantage in using our measures before applying a post-processing of the tree [34] for enforcing global monotonicity, indeed, in this case the possible exogenous information introduced by the post-processing is surely less than starting from a less monotone tree built using different measures.

The paper is organized as follows. In Section 2 some preliminary notions are given together with the rank version of conditional Shannon entropy proposed in [18]. In Section 3 three new candidate rank discrimination measures are introduced while in Section 4 a study of their rank discrimination capabilities is carried on. In Section 5 the hierarchical construction model for rank discrimination measures is introduced and the general definition of rank discrimination measure is given. In Section 6 computational aspects of splitting criteria based on rank discrimination measures are studied. In Section 7 the RDMT(H) classifier is presented. In Section 8 the indices for measuring non-monotonicity in

datasets and trees we use are reproduced in our notation, while in Section 9 we present other discrimination measures used in monotone decision tree induction. In Section 10 the experimental analysis comparing our measures with other proposals present in the literature is shown. Finally, in Section 11 we list conclusions and possible lines of future development.

2. Preliminaries

We consider a set $\Omega = \{\omega_1, \ldots, \omega_n\}$ of *objects* or *alternatives* described by a family $\mathcal{A} = \{a_1, \ldots, a_m\}$ of *attributes* with finite totally ordered range (also called *true criteria* in [8, 9]), i.e., for each $j = 1, \ldots, m$, a_j is a function on Ω ranging in $X_j = \{x_{j_1}, \ldots, x_{j_{t_j}}\}$ with $t_j > 1$ and $(X_j, \leq)$ totally ordered. We assume a *labelling function* $\lambda : \Omega \to C$ is given, where $C = \{c_1, \ldots, c_k\}$ is a set of *classes* with $k > 1$ and $(C, \leq)$ also totally ordered.

We stress that, for $i = 1, \ldots, n$, each object ω_i can be mapped to a corresponding $(m+1)$-tuple $(a_1(\omega_i), \ldots, a_m(\omega_i), \lambda(\omega_i))$, obtaining a *dataset of examples*, moreover the product space $X = X_1 \times \cdots \times X_m$ (also referred to a *description space*) forms a *lattice* $(X, \leq)$ where for each $x, y \in X$,

$$x \leq y \;\Leftrightarrow\; x_j \leq y_j, \text{ for } j = 1, \ldots, m. \tag{1}$$

Remark 2.1. *In order to avoid cumbersome notation we will use the same symbol $\leq$ for all the orderings: the context will clarify which relation we refer to.*

We say that the dataset of examples is *consistent* if and only if for each $\omega_i, \omega_h \in \Omega$ it holds

$$(a_1(\omega_i), \ldots, a_m(\omega_i)) = (a_1(\omega_h), \ldots, a_m(\omega_h)) \;\Rightarrow\; \lambda(\omega_i) = \lambda(\omega_h), \tag{2}$$

moreover, it is said to be *monotone consistent* if and only if for each $\omega_i, \omega_h \in \Omega$ it holds

$$(a_1(\omega_i), \ldots, a_m(\omega_i)) \leq (a_1(\omega_h), \ldots, a_m(\omega_h)) \;\Rightarrow\; \lambda(\omega_i) \leq \lambda(\omega_h). \tag{3}$$

Remark 2.2. *The monotone consistency assumption on the dataset is much stronger than the consistency assumption since the former implies the latter.*

Notice that in the rest of the paper no particular hypothesis is required on the given dataset, that is we will assume the dataset is neither monotone consistent nor consistent.

Recall that each attribute $a_j \in \mathcal{A}$ as well as the labelling function λ determines a partition of Ω, whose elements are denoted, respectively, as

$$\{a_j = x_{j_s}\} = \{\omega_h \in \Omega \,:\, a_j(\omega_h) = x_{j_s}\},\ s = 1, \ldots, t_j, \tag{4}$$

$$\{\lambda = c_q\} = \{\omega_h \in \Omega \,:\, \lambda(\omega_h) = c_q\},\ q = 1, \ldots, k, \tag{5}$$

moreover, the same partitions can be object-wise written, denoting for each $\omega_i \in \Omega$

$$[\omega_i]_{a_j} = \{\omega_h \in \Omega \,:\, a_j(\omega_i) = a_j(\omega_h)\}, \quad (6)$$
$$[\omega_i]_{\lambda} = \{\omega_h \in \Omega \,:\, \lambda(\omega_i) = \lambda(\omega_h)\}, \quad (7)$$

where for each $\omega_h \in [\omega_i]_{a_j}$ we have $[\omega_h]_{a_j} = [\omega_i]_{a_j}$ and, analogously, for each $\omega_h \in [\omega_i]_{\lambda}$ we have $[\omega_h]_{\lambda} = [\omega_i]_{\lambda}$.

In [18] the following object-wise writing of conditional Shannon entropy is shown (reported here in our notation).

Proposition 2.1. *Put* $p_s = \frac{|\{a_j = x_{j_s}\}|}{|\Omega|}$ *and* $p_{q,s} = \frac{|\{\lambda = c_q\} \cap \{a_j = x_{j_s}\}|}{|\Omega|}$*:*

$$\begin{aligned} H_S(\lambda|a_j) &= \sum_{s=1}^{t_j} p_s \left(-\sum_{q=1}^{k} \left(\frac{p_{q,s}}{p_s} \right) \log_2 \left(\frac{p_{q,s}}{p_s} \right) \right) \\ &= \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(-\log_2 \left(\frac{|[\omega_i]_{\lambda} \cap [\omega_i]_{a_j}|}{|[\omega_i]_{a_j}|} \right) \right). \end{aligned}$$

In the same paper the authors underline the incapability of conditional Shannon entropy to detect monotonicity of λ w.r.t. a_j. To overcome this obstacle, they go back to the dominance rough set approach (see [14, 15]) introducing the notion of *dominant set* generated, respectively, by a_j and λ. For each $\omega_i \in \Omega$, they define

$$[\omega_i]^{\leq}_{a_j} = \{\omega_h \in \Omega \,:\, a_j(\omega_i) \leq a_j(\omega_h)\}, \quad (8)$$
$$[\omega_i]^{\leq}_{\lambda} = \{\omega_h \in \Omega \,:\, \lambda(\omega_i) \leq \lambda(\omega_h)\}. \quad (9)$$

Then they propose a rank version of conditional Shannon entropy, obtained simply substituting in the object-wise writing the equivalence classes $[\omega_i]_{\lambda} \cap [\omega_i]_{a_j}$ and $[\omega_i]_{a_j}$ with the corresponding dominant sets, deriving the following definition (we keep conditional notation just for uniformity).

Definition 2.1 (Rank Shannon discrimination measure)**.**

$$H^*_S(\lambda|a_j) = \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(-\log_2 \left(\frac{|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|} \right) \right).$$

In Definition 2.1, the ratio $\frac{|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}$ is a measure of satisfaction of the *local monotonicity constraint* for a fixed $\omega_i \in \Omega$, quantifying the validity of

$$a_j(\omega_i) \leq a_j(\omega_h) \;\Rightarrow\; \lambda(\omega_i) \leq \lambda(\omega_h),$$

for all $\omega_h \in \Omega$.

Proposition 2.2. *For a fixed $\omega_i \in \Omega$,*

$$\frac{|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|} = 1 \text{ if and only if } a_j(\omega_i) \leq a_j(\omega_h) \Rightarrow \lambda(\omega_i) \leq \lambda(\omega_h),$$

for all $\omega_h \in \Omega$.

Proof. It immediately follows since $\frac{|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|} = 1$ if and only if $\{\omega_h \in \Omega : \lambda(\omega_i) \leq \lambda(\omega_h) \wedge a_j(\omega_i) \leq a_j(\omega_h)\} = \{\omega_h \in \Omega : a_j(\omega_i) \leq a_j(\omega_h)\}$ and this is true if and only if the local monotonicity constraint is satisfied for ω_i. □

3. New rank discrimination measures

In this section we extend the approach proposed by Hu et al in [18] to other well-known discrimination measures such as Gini measure and Yuan and Shaw measure and we investigate if the obtained functions are proper rank discrimination measures.

We start with Gini measure whose object-wise writing is given in the following proposition.

Proposition 3.1. *Put $p_s = \frac{|\{a_j = x_{j_s}\}|}{|\Omega|}$ and $p_{q,s} = \frac{|\{\lambda = c_q\} \cap \{a_j = x_{j_s}\}|}{|\Omega|}$:*

$$\begin{aligned} H_G(\lambda|a_j) &= \sum_{s=1}^{t_j} p_s \left(1 - \sum_{q=1}^{k} \left(\frac{p_{q,s}}{p_s}\right)^2\right) \\ &= \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(1 - \frac{|[\omega_i]_{\lambda} \cap [\omega_i]_{a_j}|}{|[\omega_i]_{a_j}|}\right). \end{aligned}$$

Proof. Consider fixed q and s. Since $\{a_j = x_{j_s}\} = [\omega_i]_{a_j}$ for each $\omega_i \in \{a_j = x_{j_s}\}$ and $\{\lambda = c_q\} \cap \{a_j = x_{j_s}\} = [\omega_i]_{\lambda} \cap [\omega_i]_{a_j}$ for each $\omega_i \in \{\lambda = c_q\} \cap \{a_j = x_{j_s}\}$, it follows

$$\frac{|\{\lambda = c_q\} \cap \{a_j = x_{j_s}\}|^2}{|\{a_j = x_{j_s}\}|} = \sum_{\omega_i \in \{\lambda = c_q\} \cap \{a_j = x_{j_s}\}} \frac{|[\omega_i]_{\lambda} \cap [\omega_i]_{a_j}|}{|[\omega_i]_{a_j}|}.$$

This implies

$$\begin{aligned} H_G(\lambda|a_j) &= \sum_{s=1}^{t_j} \frac{|\{a_j = x_{j_s}\}|}{|\Omega|} \left(1 - \sum_{q=1}^{k} \left(\frac{|\{\lambda = c_q\} \cap \{a_j = x_{j_s}\}|}{|\{a_j = x_{j_s}\}|}\right)^2\right) \\ &= 1 - \frac{1}{|\Omega|} \sum_{s=1}^{t_j} \sum_{q=1}^{k} \frac{|\{\lambda = c_q\} \cap \{a_j = x_{j_s}\}|^2}{|\{a_j = x_{j_s}\}|} \\ &= 1 - \frac{1}{|\Omega|} \sum_{i=1}^{|\Omega|} \frac{|[\omega_i]_{\lambda} \cap [\omega_i]_{a_j}|}{|[\omega_i]_{a_j}|} \\ &= \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(1 - \frac{|[\omega_i]_{\lambda} \cap [\omega_i]_{a_j}|}{|[\omega_i]_{a_j}|}\right). \end{aligned}$$

□

In analogy with Definition 2.1 the rank version of Gini measure is obtained just replacing the equivalence classes $[\omega_i]_\lambda \cap [\omega_i]_{a_j}$ and $[\omega_i]_{a_j}$ in the object-wise writing with the corresponding dominant sets.

Definition 3.1 (Rank Gini discrimination measure)**.**

$$H^*_G(\lambda|a_j) = \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(1 - \frac{|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}\right).$$

Notice that the rank generalization of Gini measure given in Definition 3.1 differs from the one proposed in [38].

The same procedure is now applied to the Yuan and Shaw measure of ambiguity [39]. In order to achieve the object-wise writing, we have to notice that in the standard definition of this measure a total order on the cardinalities

$$|\{\lambda = c_q\} \cap \{a_j = x_{j_s}\}|, \quad q = 1, \ldots, k,$$

is assumed for each fixed $s = 1, \ldots, t_j$, therefore, we need to "transport" this ordinal structure to objects in Ω by defining the following binary relation.

Definition 3.2. *For each* $\omega_i, \omega_h \in \Omega$

$$\omega_i \precsim_{\lambda,a_j} \omega_h \quad \textit{iff} \quad \begin{array}{l} [\omega_i]_\lambda \cap [\omega_i]_{a_j} = [\omega_h]_\lambda \cap [\omega_h]_{a_j} \\ \qquad\qquad \textit{or} \\ \left\{ \begin{array}{l} a_j(\omega_i) = a_j(\omega_h) \\ [\omega_i]_\lambda \cap [\omega_i]_{a_j} \neq [\omega_h]_\lambda \cap [\omega_h]_{a_j} \\ |[\omega_i]_\lambda \cap [\omega_i]_{a_j}| \leq |[\omega_h]_\lambda \cap [\omega_h]_{a_j}| \end{array} \right. \end{array}.$$

It is easily proven that $\precsim_{\lambda,a_j}$ is a partial preorder on Ω, moreover the symmetric part $\sim_{\lambda,a_j}$ is an equivalence relation on Ω while the asymmetric part $\prec_{\lambda,a_j}$ is a partial strict order on $\Omega_{/\sim_{\lambda,a_j}}$. In particular, to each decreasing $\succ_{\lambda,a_j}$-chain in $\Omega_{/\sim_{\lambda,a_j}}$ we can associate an increasing index starting from 1 and so for each ω_i we can define

$$\rho(\omega_i) = \text{"index of the } \sim_{\lambda,a_j}\text{-equivalence class containing } \omega_i\text{"}. \tag{10}$$

Intuitively, the aim of function ρ is to express at the object level the (not necessarily unique) permutation σ of index q corresponding to the non-increasing ordering of cardinalities $|\{\lambda = c_{\sigma(q)}\} \cap \{a_j = x_{j_s}\}|$ for each fixed $s = 1, \ldots, t_j$. Indeed, each such permutation σ uniquely determines a corresponding ρ and vice versa. This implies the function ρ is generally not unique, but it is trivial to verify that all the possible choices of ρ give rise to the same final result as it holds for the choice of the index permutation in the standard definition of Yuan and Shaw measure.

In the following, to simplify notation we assume $\frac{1}{0} = \infty$ and we define $\text{fin}(x) = \left\{ \begin{array}{ll} x & \text{if } x < \infty \\ 1 & \text{otherwise} \end{array} \right.$.

Proposition 3.2. *Put* $p_s = \frac{|\{a_j = x_{j_s}\}|}{|\Omega|}$ *and* $p_{q,s} = \frac{|\{\lambda = c_q\} \cap \{a_j = x_{j_s}\}|}{|\Omega|}$ *with* $p_{\sigma(1),s} \geq \ldots \geq p_{\sigma(k),s}$ *for* $s = 1, \ldots, t_j$*:*

$$\begin{aligned} H_Y(\lambda|a_j) &= \sum_{s=1}^{t_j} p_s \left(\sum_{q=1}^{k} \frac{p_{\sigma(q),s}}{p_{\sigma(1),s}} \log_2 \left(\text{fin} \left(\frac{q}{q-1} \right) \right) \right) \\ &= \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left[\left(\frac{\max\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]_\lambda \cap [\omega_h]_{a_j}|}{|[\omega_i]_{a_j}|} \right)^{-1} \log_2 \left(\text{fin} \left(\frac{\rho(\omega_i)}{\rho(\omega_i) - 1} \right) \right) \right]. \end{aligned}$$

Proof. Let ρ be the index function corresponding to the permutation σ. Consider fixed q and s, then for each $\omega_i \in \{\lambda = c_{\sigma(q)}\} \cap \{a_j = x_{j_s}\}$ it holds

$$|\{\lambda = c_{\sigma(1)}\} \cap \{a_j = x_{j_s}\}| = \max_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]_\lambda \cap [\omega_h]_{a_j}|,$$

$$\log_2 \left(\text{fin} \left(\frac{q}{q-1} \right) \right) = \log_2 \left(\text{fin} \left(\frac{\rho(\omega_i)}{\rho(\omega_i) - 1} \right) \right).$$

Moreover, since $\{a_j = x_{j_s}\} = [\omega_i]_{a_j}$ for each $\omega_i \in \{a_j = x_{j_s}\}$ it follows

$$|\{a_j = x_{j_s}\}| \cdot |\{\lambda = c_{\sigma(q)}\} \cap \{a_j = x_{j_s}\}| = \sum_{\omega_i \in \{\lambda = c_{\sigma(q)}\} \cap \{a_j = x_{j_s}\}} |[\omega_i]_{a_j}|.$$

This implies

$$\begin{aligned} H_Y(\lambda|a_j) &= \sum_{s=1}^{t_j} \frac{|\{a_j = x_{j_s}\}|}{|\Omega|} \left(\sum_{q=1}^{k} \frac{|\{\lambda = c_{\sigma(q)}\} \cap \{a_j = x_{j_s}\}|}{|\{\lambda = c_{\sigma(1)}\} \cap \{a_j = x_{j_s}\}|} \log_2 \left(\text{fin} \left(\frac{q}{q-1} \right) \right) \right) \\ &= \frac{1}{|\Omega|} \sum_{s=1}^{t_j} \sum_{q=1}^{k} |\{a_j = x_{j_s}\}| \frac{|\{\lambda = c_{\sigma(q)}\} \cap \{a_j = x_{j_s}\}|}{|\{\lambda = c_{\sigma(1)}\} \cap \{a_j = x_{j_s}\}|} \log_2 \left(\text{fin} \left(\frac{q}{q-1} \right) \right) \\ &= \frac{1}{|\Omega|} \sum_{i=1}^{|\Omega|} \frac{|[\omega_i]_{a_j}|}{\max\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]_\lambda \cap [\omega_h]_{a_j}|} \log_2 \left(\text{fin} \left(\frac{\rho(\omega_i)}{\rho(\omega_i) - 1} \right) \right) \\ &= \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left[\left(\frac{\max\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]_\lambda \cap [\omega_h]_{a_j}|}{|[\omega_i]_{a_j}|} \right)^{-1} \log_2 \left(\text{fin} \left(\frac{\rho(\omega_i)}{\rho(\omega_i) - 1} \right) \right) \right]. \end{aligned}$$

□

To introduce the rank version of Yuan and Shaw measure we need a definition of relation $\precsim_{\lambda,a_j}$ taking into account dominant sets.

Definition 3.3. *For each* $\omega_i, \omega_h \in \Omega$

$$\omega_i \precsim'_{\lambda,a_j} \omega_h \quad \textit{iff} \qquad [\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j} = [\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}$$
$$\textit{or}$$
$$\begin{cases} a_j(\omega_i) = a_j(\omega_h) \\ [\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j} \neq [\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j} \\ |[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}| \leq |[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}| \end{cases}.$$

Even in this case it is easy to show that relation $\precsim'_{\lambda,a_j}$ is a partial preorder on Ω and all the properties mentioned before still hold for $\sim'_{\lambda,a_j}$ and $\prec'_{\lambda,a_j}$. Hence, for each ω_i we can define

$$\rho'(\omega_i) = \text{“index of the } \sim'_{\lambda,a_j}\text{-equivalence class containing } \omega_i\text{”}, \qquad (11)$$

and the rank version of Yuan and Shaw measure is obtained as follows.

Definition 3.4 (Rank Yuan and Shaw discrimination measure)**.**

$$H^*_Y(\lambda|a_j) = \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left[\left(\frac{\max\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|} \right)^{-1} \log_2 \left(\text{fin} \left(\frac{\rho'(\omega_i)}{\rho'(\omega_i) - 1} \right) \right) \right].$$

Observing Definition 3.4 one can see that the rank version of Yuan and Shaw measure considers for each object ω_i only the set $[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}$ with the maximum cardinality having $\omega_h \in [\omega_i]_{a_j}$. This optimistic approach may produce a sort of blindness w.r.t. monotonicity since the ratio $\frac{\max\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}$ could be 1 even if $\frac{|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}$ is less than 1. Furthermore, the measure takes into account an ordering on the cardinalities of dominant sets (expressed by ρ') which may conflict with the order on the values determining the dominant sets themselves, which is the one we wish to preserve. Previous discussion suggests the rank generalization of Yuan and Shaw measure may not behave as a proper rank discrimination measure.

Therefore, in the next definition we directly introduce a third measure which is inspired to the functional structure of Definition 3.4 but has a cautious nature and so we call it *pessimistic*.

Definition 3.5 (Pessimistic rank discrimination measure)**.**

$$H^*_P(\lambda|a_j) = \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left[- \left(\frac{\min\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|} \right)^{-1} \log_2 \left(\frac{\min\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|} \right) \right].$$

We stress that the ratio $\frac{\min\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}$ can be equal to 1 only in the case $\frac{|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}$ is 1 but it could be less than 1 even in the case the last equality holds.

Example 3.1 shows the computation of H^*_S, H^*_G, H^*_Y and H^*_P.

Example 3.1. *Consider the set of objects $\Omega = \{\omega_1, \ldots, \omega_5\}$ together with the attribute a_1 ranging in $\{0, 1, 2\}$ and the labelling function λ ranging in $\{0, 1\}$ whose definition is reported in Table 1.*

	a_1	λ
ω_1	0	0
ω_2	1	1
ω_3	2	0
ω_4	1	1
ω_5	1	0

Table 1: Definition of a_1 and λ

The dominant sets generated by a_1 and λ are: $[\omega_1]^{\leq}_{a_1} = [\omega_1]^{\leq}_{\lambda} = [\omega_3]^{\leq}_{\lambda} = [\omega_5]^{\leq}_{\lambda} = \Omega$, $[\omega_2]^{\leq}_{a_1} = [\omega_4]^{\leq}_{a_1} = [\omega_5]^{\leq}_{a_1} = \{\omega_2, \omega_3, \omega_4, \omega_5\}$, $[\omega_3]^{\leq}_{a_1} = \{\omega_3\}$ and $[\omega_2]^{\leq}_{\lambda} = [\omega_4]^{\leq}_{\lambda} = \{\omega_2, \omega_4\}$. The corresponding intersection then are: $[\omega_1]^{\leq}_{\lambda} \cap [\omega_1]^{\leq}_{a_1} = \Omega$, $[\omega_2]^{\leq}_{\lambda} \cap [\omega_2]^{\leq}_{a_1} = [\omega_4]^{\leq}_{\lambda} \cap [\omega_4]^{\leq}_{a_1} = \{\omega_2, \omega_4\}$, $[\omega_3]^{\leq}_{\lambda} \cap [\omega_3]^{\leq}_{a_1} = \{\omega_3\}$ and $[\omega_5]^{\leq}_{\lambda} \cap [\omega_5]^{\leq}_{a_1} = \{\omega_2, \omega_3, \omega_4, \omega_5\}$.

Now it is easily verified that relation $\precsim'_{\lambda,a_1}$ has the following descending $\succ'_{\lambda,a_1}$-chains: $\omega_5 \succ'_{\lambda,a_1} \omega_2 \sim'_{\lambda,a_1} \omega_4$, while ω_1 and ω_3 form each one a one-element chain. This implies $\rho'(\omega_1) = \rho'(\omega_3) = \rho'(\omega_5) = 1$ and $\rho'(\omega_2) = \rho'(\omega_4) = 2$. A straightforward computation leads to $H^*_S(\lambda|a_1) = H^*_Y(\lambda|a_1) = \frac{2}{5}$, $H^*_G(\lambda|a_1) = \frac{1}{5}$ and $H^*_P(\lambda|a_1) = \frac{6}{5}$.

4. Rank discrimination capabilities

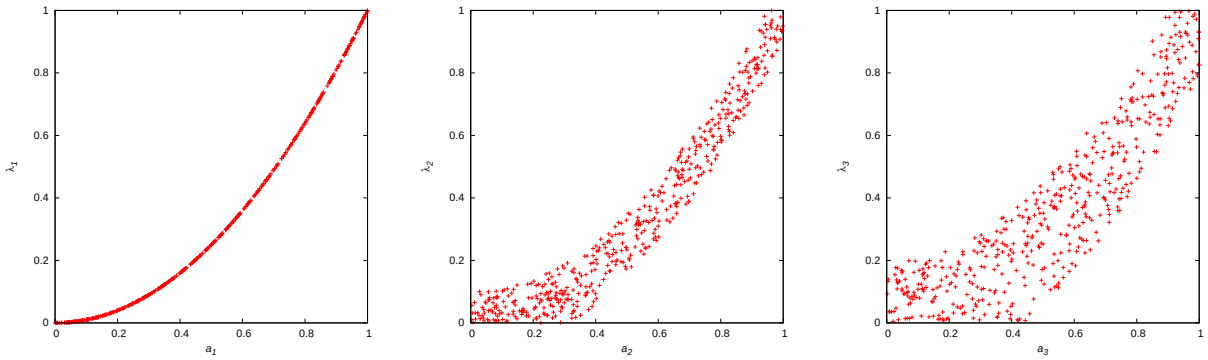
In previous section three new functions have been presented as candidate rank discrimination measures. Evidently, the next step is to verify if they possess good properties, this in turn will allow a deeper study of only those functions behaving properly.

We recall that the purpose of a rank discrimination measure is to be used at each step of a greedy tree induction algorithm to select the attribute "enforcing the most" the local monotonicity constraint: each time the attribute minimizing the rank discrimination measure is chosen for splitting. More precisely, each measure induces a total preoder on the set of attributes $\mathcal{A}$ that we will denote, respectively, as $\leq_{H^*_S}$, $\leq_{H^*_G}$, $\leq_{H^*_Y}$ and $\leq_{H^*_P}$, of which only the (not necessarily unique) minimal element is considered.

From previous discussion it follows that our goal is to find functions able to distinguish the presence of monotonicity of λ w.r.t. an attribute a_j and presenting robustness to possible non-monotone noise in the data.

For what concerns the monotonicity discrimination power, we generated three datasets of 500 examples where, for $j = 1, 2, 3$, a_j is a uniform random variable on $[0, 1]$ and $\lambda_j = a_j^2 + \varepsilon_j$, with ε_j uniform random variable ranging, respectively, on $\{0\}$, $[-0.1, 0.1]$ and $[-0.2, 0.2]$, assuring $\lambda_j \in [0, 1]$. Table 2 shows the three datasets and the corresponding values of the proposed measures.

$$H_S(\lambda_j|a_j) = H_G(\lambda_j|a_j) = H_Y(\lambda_j|a_j) = 0, \text{ for } j = 1, 2, 3$$



$H_S^*(\lambda_1\|a_1) = 0$	$H_S^*(\lambda_2\|a_2) = 0.1919$	$H_S^*(\lambda_3\|a_3) = 0.3334$
$H_G^*(\lambda_1\|a_1) = 0$	$H_G^*(\lambda_2\|a_2) = 0.1037$	$H_G^*(\lambda_3\|a_3) = 0.1729$
$H_Y^*(\lambda_1\|a_1) = 0$	$H_Y^*(\lambda_2\|a_2) = 0$	$H_Y^*(\lambda_3\|a_3) = 0$
$H_P^*(\lambda_1\|a_1) = 0$	$H_P^*(\lambda_2\|a_2) = 0.5852$	$H_P^*(\lambda_3\|a_3) = 1.0811$

Table 2: Monotonicity discrimination power

Observing Table 2 it is immediate to see that standard discrimination measures H_S, H_G and H_Y are constantly equal to 0, showing their insensitivity to non-monotonicity. On the contrary, measures H_S^*, H_G^* and H_P^* are equal to 0 in case of perfect monotonicity while their value increases as the non-monotone noise increases. As expected, the rank generalization of Yuan and Shaw measure does not behave in a meaningful way since it is not able to distinguish the increment of non-monotone noise, being equal to 0 in the three datasets.

We investigate now the robustness with respect to non-monotone noise, considering the dataset represented in Table 3.

In Table 3 one can see that measures H_S, H_G, H_Y and H_Y^* consider equivalent, respectively, situations (a) and (b) and situations (c) and (d). More precisely, it holds $a_1 =_{H_Y^*} a_2 <_{H_Y^*} a_3 =_{H_Y^*} a_4$ (all the other measures induce the same total preorder on $\mathcal{A}$). In other terms, this means that situations of non-monotonicity such as (b) and (d) are judged equivalent, respectively, to situations of perfect (a) or almost perfect (c) monotonicity. But even worse, situation (b) is judged strictly better than situation (c).

Fr.	λ		
a_1	c_1	c_2	c_3
x_{1_1}	10	0	0
x_{1_2}	0	10	0
x_{1_3}	0	10	0
x_{1_4}	0	0	10
x_{1_5}	0	0	10

$H_S(\lambda|a_1) = 0$
$H_G(\lambda|a_1) = 0$
$H_Y(\lambda|a_1) = 0$

$H^*_S(\lambda|a_1) = 0$
$H^*_G(\lambda|a_1) = 0$
$H^*_Y(\lambda|a_1) = 0$
$H^*_P(\lambda|a_1) = 0$

(a)

Fr.	λ		
a_2	c_1	c_2	c_3
x_{2_1}	10	0	0
x_{2_2}	0	10	0
x_{2_3}	0	0	10
x_{2_4}	0	10	0
x_{2_5}	10	0	0

$H_S(\lambda|a_2) = 0$
$H_G(\lambda|a_2) = 0$
$H_Y(\lambda|a_2) = 0$

$H^*_S(\lambda|a_2) = 0.6000$
$H^*_G(\lambda|a_2) = 0.2833$
$H^*_Y(\lambda|a_2) = 0$
$H^*_P(\lambda|a_2) = 1.4616$

(b)

Fr.	λ		
a_3	c_1	c_2	c_3
x_{3_1}	9	0	1
x_{3_2}	0	10	0
x_{3_3}	0	10	0
x_{3_4}	0	0	10
x_{3_5}	1	0	9

$H_S(\lambda|a_3) = 0.1875$
$H_G(\lambda|a_3) = 0.0720$
$H_Y(\lambda|a_3) = 0.0444$

$H^*_S(\lambda|a_3) = 0.0856$
$H^*_G(\lambda|a_3) = 0.0516$
$H^*_Y(\lambda|a_3) = 0.2000$
$H^*_P(\lambda|a_3) = 0.7279$

(c)

Fr.	λ		
a_4	c_1	c_2	c_3
x_{4_1}	0	0	10
x_{4_2}	1	0	9
x_{4_3}	9	0	1
x_{4_4}	0	10	0
x_{4_5}	0	10	0

$H_S(\lambda|a_4) = 0.1875$
$H_G(\lambda|a_4) = 0.0720$
$H_Y(\lambda|a_4) = 0.0444$

$H^*_S(\lambda|a_4) = 0.7225$
$H^*_G(\lambda|a_4) = 0.2743$
$H^*_Y(\lambda|a_4) = 0.2000$
$H^*_P(\lambda|a_4) = 31.7023$

(d)

Table 3: Robustness with respect to non-monotone noise

On the other hand, measures H^*_S, H^*_G and H^*_P behave in a coherent way since the situation (c) of almost perfect monotonicity is ranked after situation (a) of perfect monotonicity but before situations (b) and (d).

The given examples show H^*_Y is not a good rank discrimination measure, hence, in what follows we will focus on H^*_S, H^*_G and H^*_P.

As already pointed out, it is extremely important to investigate the order structure the measures H^*_S, H^*_G and H^*_P induce on the set of attributes $\mathcal{A}$. In particular, the three measures possess an individual meaning only in the case they are not a monotone transformation of each other. Example 4.1 shows it is not the case, in fact the three measures generally induce different total preorders and so they will produce trees with different shape in general.

Example 4.1. *Consider the set of objects $\Omega = \{\omega_1, \ldots, \omega_5\}$ together with the attributes a_1 ranging in $\{0, 1, 2, 3\}$ and a_2 ranging in $\{0, 1\}$, and the labelling function λ ranging in $\{0, 1, 2, 3\}$.*

If a_1,a_2 and λ are defined as in Table 4 (a) then we have $a_1 <_{H^*_S} a_2$ and $a_1 <_{H^*_P} a_2$ while $a_1 >_{H^*_G} a_2$. Furthermore, if a_1,a_2 and λ are defined as in Table 4 (b) then we have $a_1 <_{H^*_S} a_2$ and $a_1 <_{H^*_G} a_2$ while $a_1 >_{H^*_P} a_2$.

5. Hierarchical construction model for rank discrimination measures

In the spirit of [24, 25, 26] we aim to develop a *hierarchical construction model* for rank discrimination measures (see [27]), with the goal of isolating which properties a function must satisfy to be a measure of this type. As a side effect, the definition of a hierarchical construction model is also important since it provides a base for creating new measures.

	a_1	a_2	λ
ω_1	3	0	0
ω_2	2	1	1
ω_3	1	1	2
ω_4	0	0	1
ω_5	3	0	3

$H^*_S(\lambda|a_1) = 0.5813$
$H^*_S(\lambda|a_2) = 0.7287$
$H^*_G(\lambda|a_1) = 0.3066$
$H^*_G(\lambda|a_2) = 0.3000$
$H^*_P(\lambda|a_1) = 1.4559$
$H^*_P(\lambda|a_2) = 7.7657$

(a)

	a_1	a_2	λ
ω_1	0	1	3
ω_2	2	0	2
ω_3	1	1	1
ω_4	3	0	1
ω_5	0	1	0

$H^*_S(\lambda|a_1) = 0.6643$
$H^*_S(\lambda|a_2) = 0.7627$
$H^*_G(\lambda|a_1) = 0.2600$
$H^*_G(\lambda|a_2) = 0.3600$
$H^*_P(\lambda|a_1) = 5.0438$
$H^*_P(\lambda|a_2) = 4.1748$

(b)

Table 4: Definitions of a_1, a_2 and λ

In order to simplify notation, for a fixed $a_j \in \mathcal{A}$ and λ denote:

$$\mathrm{dsr}(\omega_i) = \frac{|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}, \tag{12}$$

$$\mathrm{mindsr}(\omega_i) = \frac{\min\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}, \tag{13}$$

$$\mathrm{maxdsr}(\omega_i) = \frac{\max\limits_{\omega_h \in [\omega_i]_{a_j}} |[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}|}{|[\omega_i]^{\leq}_{a_j}|}, \tag{14}$$

$$\mathrm{avgdsr}(\omega_i) = \frac{\sum\limits_{\omega_h \in [\omega_i]_{a_j}} \frac{|[\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}|}{|[\omega_i]_{a_j}|}}{|[\omega_i]^{\leq}_{a_j}|}. \tag{15}$$

Notice that the function dsr considers only the object ω_i, while the functions mindsr, maxdsr and avgdsr consider all the objects "in the same conditions" for what concerns the attribute a_j, that is those belonging to the equivalence class $[\omega_i]_{a_j}$. In particular, it holds for every $\omega_h \in [\omega_i]_{a_j}$, $\mathrm{mindsr}(\omega_h) = \mathrm{mindsr}(\omega_i)$, $\mathrm{maxdsr}(\omega_h) = \mathrm{maxdsr}(\omega_i)$ and $\mathrm{avgdsr}(\omega_h) = \mathrm{avgdsr}(\omega_i)$.

The functions H^*_S, H^*_G and H^*_P can be rewritten as

$$H^*_S(\lambda|a_j) = \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(-\log_2(\mathrm{dsr}(\omega_i))\right), \tag{16}$$

$$H^*_G(\lambda|a_j) = \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(1-\mathrm{dsr}(\omega_i)\right), \tag{17}$$

$$H^*_P(\lambda|a_j) = \sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(-\frac{\log_2\left(\mathrm{mindsr}(\omega_i)\right)}{\mathrm{mindsr}(\omega_i)}\right). \tag{18}$$

After a careful look all the measures presented so far share a common functional structure, in which we can distinguish three functions f^*, g^* and h^*, composed hierarchically. In particular, for fixed λ and a_j, the h^*-layer considers all the objects in Ω, while both the g^*-layer and the f^*-layer take into account a single object ω_i. Table 5 lists the different layers for the measures introduced so far. We will use subscripts S, G and P to refer to layers f^*, g^* and h^* of each measure.

Layer	H^*_S	H^*_G	H^*_P
f^*	$\mathrm{dsr}(\omega_i)$		$\mathrm{mindsr}(\omega_i)$
g^*	$-\log_2 f^*(\omega_i)$	$1-f^*(\omega_i)$	$-\frac{\log_2 f^*(\omega_i)}{f^*(\omega_i)}$
h^*	$\sum_{i=1}^{\lvert\Omega\rvert} \frac{1}{\lvert\Omega\rvert} g^*(f^*(\omega_i))$		

Table 5: Hierarchical construction model for measures H^*_S, H^*_G and H^*_P

The f^*-layer is a function quantifying the validity of the local monotonicity constraint of λ with respect to a_j for a fixed $\omega_i \in \Omega$, i.e., it measures the satisfaction of $a_j(\omega_i) \leq a_j(\omega_h) \Rightarrow \lambda(\omega_i) \leq \lambda(\omega_h)$, for every $\omega_h \in \Omega$. For this, f^* can be referred to as *object-wise local monotonicity measure* and it is asked to satisfy the following conditions for every $\omega_i \in \Omega$:

(F1) $\mathrm{mindsr}(\omega_i) \leq f^*(\omega_i) \leq \mathrm{maxdsr}(\omega_i)$;

(F2) if $f^*(\omega_i) = 1$, then $a_j(\omega_i) \leq a_j(\omega_h) \Rightarrow \lambda(\omega_i) \leq \lambda(\omega_h)$, for every $\omega_h \in \Omega$;

(F3) if $[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j} \subseteq [\omega_h]^{\leq}_{\lambda} \cap [\omega_h]^{\leq}_{a_j}$ and $[\omega_i]_{a_j} = [\omega_h]_{a_j}$, then $f^*(\omega_i) \leq f^*(\omega_h)$.

Notice that condition **(F1)** implies $f^*(\omega_i) \in (0,1]$. From a semantic point of view, condition **(F1)** imposes two natural boundaries to $f^*(\omega_i)$ which are determined by objects belonging to $[\omega_i]_{a_j}$. Condition **(F2)** requires that $f^*(\omega_i)$ is equal to 1 only in the case of complete satisfaction of the local monotonicity constraint for ω_i. Finally, condition **(F3)** is a monotonicity requirement related to other objects in $[\omega_i]_{a_j}$. It is immediate to verify that dsr, mindsr and avgdsr

satisfy conditions **(F1)**–**(F3)**, and so f_S^*, f_G^* and f_P^*. On the contrary, maxdsr can fail to satisfy **(F2)**. From previous discussion, we have that for every $\omega_i \in \Omega$

$$\frac{1}{|\Omega|} \leq f_P^*(\omega_i) \leq f_G^*(\omega_i) = f_S^*(\omega_i) \leq 1. \tag{19}$$

Going on, the g^*-layer is a strictly decreasing transformation of the f^*-layer, and it is a real function defined on $(0, 1]$, i.e., it is an *object-wise local non-monotonicity measure*. Putting $f_i = f^*(\omega_i)$, g^* must satisfy the following conditions:

(G1) $g^*(f_i) \in [0, +\infty)$;

(G2) g^* is a strictly decreasing function of f_i;

(G3) $g^*(1) = 0$.

Notice that g_G^*, g_S^* and g_P^* satisfy conditions **(G1)**–**(G3)**, moreover on the interval $(0, 1]$, g_P^* dominates g_S^* which, in turn, dominates g_G^*. Considering (19), for every $\omega_i \in \Omega$ we also have

$$g_G^*(f_G^*(\omega_i)) \leq g_S^*(f_S^*(\omega_i)) \leq g_P^*(f_P^*(\omega_i)). \tag{20}$$

Finally, the h^*-layer is an aggregation operator of the g^*-layers corresponding to objects in Ω, and thus it is a real function defined on $[0, \infty)^n$, which can be referred to as *aggregated local non-monotonicity measure*. Putting $g_i = g^*(f^*(\omega_i))$ for $i = 1, \ldots, n$, h^* must satisfy the following conditions:

(H1) $h^*(g_1, \ldots, g_n) \in [0, +\infty)$;

(H2) $h^*(g_1, \ldots, g_n) = h^*(g_{\sigma(1)}, \ldots, g_{\sigma(n)})$ for every permutation σ;

(H3) if $g_i \leq g_i'$, then $h^*(g_1, \ldots, g_i, \ldots, g_n) \leq h^*(g_1, \ldots, g_i', \ldots, g_n)$;

(H4) $h^*(g_1, \ldots, g_n) = 0$ if and only if $g_i = 0$ for $i = 1, \ldots, n$.

Again, it is easily seen that the arithmetic mean satisfies conditions **(H1)**–**(H4)**, nevertheless, it is not the only possible choice, indeed, also the maximum operator and the quadratic mean satisfy such conditions.

Next proposition summarizes some properties of H_G^*, H_S^* and H_P^*.

Proposition 5.1. *The following statements hold:*

(i) $H_G^*(\lambda|a_j) \leq H_S^*(\lambda|a_j) \leq H_P^*(\lambda|a_j)$;

(ii) $0 \leq H_G^*(\lambda|a_j) < \frac{|\Omega|-1}{|\Omega|}$;

(iii) $0 \leq H_S^*(\lambda|a_j) < \log_2(|\Omega|)$;

(iv) $0 \leq H_P^*(\lambda|a_j) < |\Omega| \log_2(|\Omega|)$.

Proof. All the properties follow by inequalities (19) and (20). In particular, for properties *(ii)–(iv)* the upper bound cannot be reached since the f^*-layers of objects in Ω cannot be simultaneously all equal to $\frac{1}{|\Omega|}$. □

The layered decomposition we just presented suggests the following definition of a general rank discrimination measure.

Definition 5.1. *Let f^*, g^* and h^* be functions satisfying conditions* **(F1)**–**(F3)**, **(G1)**–**(G3)** *and* **(H1)**–**(H4)**, *respectively, then we call* **rank discrimination measure**

$$H^*(\lambda|a_j) = h^*(g^*(f^*(\omega_1)), \ldots, g^*(f^*(\omega_n))).$$

In Definition 5.1 we kept conditional notation for uniformity.

Remark 5.1. *In the following we will use the symbol H^* to denote a generic rank discrimination measure (i.e., satisfying Definition 5.1) while the symbol H will stand for a generic discrimination measure used for splitting in a TDIDT algorithm: in other terms the functions H^*'s are just a subset of the possible functions H's that can be used for splitting.*

In next theorem we prove that a rank discrimination measure H^* defined as in Definition 5.1 reaches its minimum value 0 if and only if λ is monotone w.r.t. a_j.

Theorem 5.1. *Let f^*, g^* and h^* be functions satisfying conditions* **(F1)**–**(F3)**, **(G1)**–**(G3)** *and* **(H1)**–**(H4)**, *respectively, then $H^*(\lambda|a_j) = 0$ if and only if λ is monotone with respect to a_j, that is for every $\omega_i, \omega_h \in \Omega$,*

$$a_j(\omega_i) \leq a_j(\omega_h) \;\Rightarrow\; \lambda(\omega_i) \leq \lambda(\omega_h).$$

Proof. Condition **(H4)** implies that the h^*-layer is 0 if and only if the g^*-layer related to each $\omega_i \in \Omega$ is equal 0 and by virtue of conditions **(G2)** and **(G3)** this can happen if and only if the corresponding f^*-layer is equal to 1. Finally, by conditions **(F1)** and **(F2)** the f^*-layer is equal to 1 for every ω_i if and only if the local monotonicity constraint of λ w.r.t. a_j is satisfied for every ω_i. □

Definition 5.1 enables us to introduce new rank discrimination measures, as the two proposed in next example.

Example 5.1. *Consider the functions*

$$H^*_M(\lambda|a_j) \;=\; \max_{i=1,\ldots,|\Omega|} \left\{1 - \mathrm{dsr}(\omega_i)^2\right\};$$

$$H^*_Q(\lambda|a_j) \;=\; \sqrt{\sum_{i=1}^{|\Omega|} \frac{1}{|\Omega|} \left(1 - \mathrm{avgdsr}(\omega_i)\right)^2}.$$

It is easily verified that both functions H^*_M and H^*_Q respect all the conditions in Definition 5.1 and Table 6 lists the hierarchical decomposition of the two new measures.

Layer	H^*_M	H^*_Q
f^*	$\mathrm{dsr}(\omega_i)$	$\mathrm{avgdsr}(\omega_i)$
g^*	$1 - f^*(\omega_i)^2$	$1 - f^*(\omega_i)$
h^*	$\max_{i=1,\ldots,\lvert\Omega\rvert} \{g^*(f^*(\omega_i))\}$	$\sqrt{\sum_{i=1}^{\lvert\Omega\rvert} \frac{1}{\lvert\Omega\rvert} g^*(f^*(\omega_i))^2}$

Table 6: Hierarchical construction model for measures H^*_M and H^*_Q

6. Computational aspects

Since a rank discrimination measure is supposed to be used for splitting at each step of an inductive algorithm, it is crucial to evaluate the complexity of the resulting splitting criterion in terms of time and space. The proposed measures are essentially based on dominant sets, thus the most complex part seems to be the construction and storage of such structures.

In the following we propose an algorithm to compute the splitting using one between H^*_S, H^*_G or H^*_P. Suppose $\Omega = \{\omega_1, \ldots, \omega_n\}$, $\mathcal{A} = \{a_1, \ldots, a_m\}$ and λ are given. For a fixed $j \in \{1, \ldots, m\}$, the collections of dominant sets related to the attribute a_j and the labelling function λ, respectively, can be stored into two two-dimensional arrays A and D both of size $n \times 4$. We assume to identify the values in the range of a_j and λ with their ordinal index, starting from 0, moreover, arrays are indexed starting from 0 in Java-like fashion.

The structure of both A and D is as follows: the first column is used to contain the row position of each object, the second column contains the index of each object in Ω minus 1 (used as identifier), the third column contains the corresponding value of a_j (or λ), and the fourth column contains an integer value that will simplify the computation of cardinalities.

We describe the initialization of A:

(a) The second and the third columns are filled, setting for $h = 0, \ldots, n-1$, $A[h][1] := h$ and $A[h][2] = a_j(\omega_{h+1})$, while the first and fourth columns are left empty.

(b) The rows are ordered increasingly with respect to the third column.

(c) The fourth column is filled searching for a change in the value reported in the third column. Set $A[0][3] := 0$ and for $h = 1, \ldots, n-1$, if $A[h][2] > A[h-1][2]$ then set $A[h][3] := h$, otherwise set $A[h][3] := A[h-1][3]$.

(d) The first column is filled with the row index of each object, i.e., for $h = 0, \ldots, n-1$, set $A[A[h][1]][0] := h$.

The points *(a)*, *(c)* and *(d)* can be solved in $O(n)$ steps, while point *(b)* requires $O(n \log n)$ steps if executed with a comparison-based sorting algorithm such as MergeSort (which will consider the third column, having index 2).

Hence, the construction of A requires $O(n \log n)$ steps, while the space complexity is $O(n)$. Algorithm 1 shows the initialization of A. The same procedure and considerations can be applied to D.

Algorithm 1 Initialization of the two-dimensional array A

```
function InitDomSet(A, a_j)
    for h = 0 to n − 1 do
        A[h][1] := h
        A[h][2] := a_j(ω_{h+1})
    end for
    MergeSort(A, 2)
    A[0][3] := 0
    for h = 1 to n − 1 do
        if A[h][2] > A[h − 1][2] then A[h][3] := h
        else A[h][3] := A[h − 1][3]
    end for
    for h = 0 to n − 1 do A[A[h][1]][0] := h
end function
```

Once A and D have been initialized we have $|[\omega_i]^{\preceq}_{a_j}| = n - A[A[i-1][0]][3]$ and $|[\omega_i]^{\preceq}_{a_j}| = n - D[D[i-1][0]][3]$, hence the computation of $|[\omega_i]^{\preceq}_{a_j}|$ or $|[\omega_i]^{\preceq}_{\lambda}|$ has complexity $O(1)$.

To compute $|[\omega_i]^{\preceq}_{\lambda} \cap [\omega_i]^{\preceq}_{a_j}|$, we assume to use an auxiliary array V of dimension n: notice that the space complexity is still $O(n)$. If $|[\omega_i]^{\preceq}_{a_j}| = n$, then no computation on V is needed since $|[\omega_i]^{\preceq}_{\lambda} \cap [\omega_i]^{\preceq}_{a_j}| = |[\omega_i]^{\preceq}_{\lambda}|$. Analogously, if $|[\omega_i]^{\preceq}_{\lambda}| = n$, it holds $|[\omega_i]^{\preceq}_{\lambda} \cap [\omega_i]^{\preceq}_{a_j}| = |[\omega_i]^{\preceq}_{a_i}|$. Furthermore, if $|[\omega_i]^{\preceq}_{a_j}| = 1$ or $|[\omega_i]^{\preceq}_{\lambda}| = 1$, then $|[\omega_i]^{\preceq}_{\lambda} \cap [\omega_i]^{\preceq}_{a_j}| = 1$.

When previous conditions are not met, the auxiliary array V is initialized to 0 as well as a temporary counter *count*. Then for $h = D[D[i-1][0]][3], \ldots, n-1$, we set $V[D[h][1]] := V[D[h][1]] + 1$ and after, for $h = A[A[i-1][0]][3], \ldots, n-1$, if $V[A[h][1]] > 0$ we set $count := count + 1$. At the end of those operations we have $|[\omega_i]^{\preceq}_{\lambda} \cap [\omega_i]^{\preceq}_{a_j}| = count$, thus to determine $|[\omega_i]^{\preceq}_{\lambda} \cap [\omega_i]^{\preceq}_{a_j}|$ we need at most $O(n)$ steps. Algorithm 2 shows the computation of $|[\omega_i]^{\preceq}_{\lambda} \cap [\omega_i]^{\preceq}_{a_j}|$.

The temporal complexity for calculating $\mathrm{dsr}(\omega_i)$ is determined by the computation of $|[\omega_i]^{\preceq}_{\lambda} \cap [\omega_i]^{\preceq}_{a_j}|$, and so requires $O(n)$ steps. Since $\mathrm{dsr}(\omega_i)$ must be computed for $i = 1, \ldots, n$, the computation of $H^*_S(\lambda|a_j)$ or $H^*_G(\lambda|a_j)$ requires $O(n^2)$ steps.

The computation of $\mathrm{mindsr}(\omega_i)$ for all $i = 1, \ldots, n$, requires an auxiliary array I of size n: also in this case the space complexity remains $O(n)$. For a fixed ω_i, the equivalence class $[\omega_i]_{a_j}$ is located in rows from $A[A[i-1][0]][3]$ to u, where u is the last row index for which $A[A[i-1][0]][2] = A[u][2]$, thus $\mathrm{mindsr}(\omega_i)$ can be computed scanning these rows and once computed, it can be stored in I for all the objects belonging to $[\omega_i]_{a_j}$, as $\mathrm{mindsr}(\omega_i) = \mathrm{mindsr}(\omega_h)$ for all $\omega_h \in [\omega_i]_{a_j}$. This shows that the construction of array I requires at most

Algorithm 2 Computation of $|[\omega_i]^{\leq}_{\lambda} \cap [\omega_i]^{\leq}_{a_j}|$

```
function CardInt(A, D, i)
    if A[A[i − 1][0]][3] = n then return n − D[D[i − 1][0]][3]
    if D[D[i − 1][0]][3] = n then return n − A[A[i − 1][0]][3]
    if A[A[i − 1][0]][3] = 1 or D[D[i − 1][0]][3] = 1 then return 1
    count := 0
    for h = 0 to n − 1 do V[h] := 0
    for h = D[D[i − 1][0]][3] to n − 1 do V[D[h][1]] := V[D[h][1]] + 1
    for h = A[A[i − 1][0]][3] to n − 1 do
        if V[A[h][1]] > 0 then count := count + 1
    end for
    return count
end function
```

$O(n^2)$ steps. Algorithm 3 copes with the construction of I. Finally, once I has been built, the measure $H^*_P(\lambda|a_j)$ can be computed in $O(n)$ steps, implying that the computation of $H^*_P(\lambda|a_j)$ has global complexity $O(n^2)$.

Algorithm 3 Computation of the array I containing mindsr(ω_i) for $i = 1, \ldots, n$

```
function MinDsr(A, D, I)
    temp := n
    for h = 0 to n − 1 do
        temp := min(temp, CardInt(A,D,A[h][1]))
        if h = n − 1 or (h < n − 1 and A[h + 1][2] > A[h][2]) then
            for u = A[h][3] to h do I[A[h][1]] := temp
            temp := n
        end if
    end for
end function
```

Let us stress that the construction of D can be executed once and this two-dimensional array can be preserved in memory until the end of the splitting phase. Nevertheless, all the previous work related to the construction of A and the computation of the rank discrimination measure must be repeated for $j = 1, \ldots, m$.

In conclusion the global time complexity for splitting with H^*_S, H^*_G or H^*_P is $O(m(n \log n + n^2)) = O(mn^2)$ while the space complexity is $O(n)$.

The following example shows the two-dimensional arrays used to store the collections of dominant sets and their use in the computation of the rank discrimination measure H^*_S, H^*_G and H^*_P.

Example 6.1. *Consider the dataset of Example 3.1. The two-dimensional arrays A and D corresponding to a_1 and λ are reported in Table 7.*

We have that:

$A =$

0	0	0	0
1	1	1	1
4	3	1	1
2	4	1	1
3	2	2	4

$D =$

0	0	0	0
3	2	0	0
1	4	0	0
4	1	1	3
2	3	1	3

Table 7: Two-dimensional arrays A and D representing dominant sets related to a_1 and λ

- $|[\omega_1]|^{\leq}_{a_1} = 5 - A[A[0][0]][3] = 5 - 0 = 5$ *and* $|[\omega_1]|^{\leq}_{\lambda} = 5 - D[D[0][0]][3] = 5 - 0 = 5$, *which implies* $|[\omega_1]^{\leq}_{\lambda} \cap [\omega_1]^{\leq}_{a_1}| = |[\omega_1]^{\leq}_{a_1}| = 5$, *thus it follows* $\mathrm{dsr}(\omega_1) = 1$;
- $|[\omega_2]|^{\leq}_{a_1} = 5 - A[A[1][0]][3] = 5 - 1 = 4$ *and* $|[\omega_2]|^{\leq}_{\lambda} = 5 - D[D[1][0]][3] = 5 - 3 = 2$, *and since* $V = \langle 0, 1, 0, 1, 0\rangle$ *we have that* $|[\omega_2]^{\leq}_{\lambda} \cap [\omega_2]^{\leq}_{a_1}| = 2$, *thus* $\mathrm{dsr}(\omega_2) = \frac{1}{2}$;
- $|[\omega_3]|^{\leq}_{a_1} = 5 - A[A[2][0]][3] = 5 - 4 = 1$ *and* $|[\omega_3]|^{\leq}_{\lambda} = 5 - D[D[2][0]][3] = 5 - 0 = 5$, *which implies* $|[\omega_1]^{\leq}_{\lambda} \cap [\omega_1]^{\leq}_{a_1}| = 1$, *thus it follows* $\mathrm{dsr}(\omega_3) = 1$;
- $|[\omega_4]|^{\leq}_{a_1} = 5 - A[A[3][0]][3] = 5 - 1 = 4$ *and* $|[\omega_4]|^{\leq}_{\lambda} = 5 - D[D[3][0]][3] = 5 - 3 = 2$, *and since* $V = \langle 0, 1, 0, 1, 0\rangle$ *we have that* $|[\omega_1]^{\leq}_{\lambda} \cap [\omega_1]^{\leq}_{a_1}| = 2$, *thus* $\mathrm{dsr}(\omega_1) = \frac{1}{2}$;
- $|[\omega_5]|^{\leq}_{a_1} = 5 - A[A[4][0]][3] = 5 - 1 = 4$ *and* $|[\omega_5]|^{\leq}_{\lambda} = 5 - D[D[4][0]][3] = 5 - 0 = 5$, *which implies* $|[\omega_1]^{\leq}_{\lambda} \cap [\omega_1]^{\leq}_{a_1}| = 4$, *thus* $\mathrm{dsr}(\omega_5) = 1$.

It is easily verified that $I = \langle 5, 2, 1, 2, 2\rangle$, *hence it follows that* $H^*_S(\lambda|a_1) = \frac{2}{5}$, $H^*_G(\lambda|a_1) = \frac{1}{5}$ *and* $H^*_P(\lambda|a_1) = \frac{6}{5}$.

7. RDMT(H) classifier

In order to make a comparison of the introduced measures with other proposal present in the literature we wrote a binary decision tree classifier in Java relying on the WEKA package [37]. Our algorithm is essentially based on REMT classifier [17], differing from it for the use of a user-specified discrimination measure H to minimize for splitting, instead of the rank mutual information. We call it RDMT(H) classifier, where the acronym RDMT stands for *Rank Discrimination Measure Tree* since it is mainly thought to work with H^*_G, H^*_S or H^*_P. RDMT(H) is a simple classifier parametrized by the choice of a discrimination measure H in a (possible enlargeable) set comprising H^*_G, H^*_S and H^*_P, and by other three pre-pruning parameters. No post-pruning is executed on the resulting tree.

Our classifier can deal with both numeric and ordinal attributes while the class is required to be ordinal. In case of numeric attributes the standard numeric order is considered, while for ordinal attributes (which are treated as

nominal attributes by WEKA) the order is determined by the writing order of values in the `@attribute` statement of the `.arff` file (extension of WEKA input files). Missing values are not allowed.

As common practice in tree induction [6], the RDMT(H) algorithm is completely specified once are known the following parts: *splitting rule*, *stopping rule* and *labelling rule*. The algorithm proceeds recursively applying this three rules, working at each step on a local set of objects Ω_α where $\Omega_0 = \Omega$.

For the splitting rule, since we restrict to binary trees, each attribute a_j must be binaryzed as it is done in [31] for numeric attributes. In detail, if $X_j = \{x_{j_1}, \ldots, x_{j_{t_j}}\}$, we denote with $a_j^{x_{j_s}}$ the binary attribute defined as

$$a_j^{x_{j_s}}(\omega_i) = \begin{cases} 0 & a_j(\omega_i) \leq x_{j_s} \\ 1 & \text{otherwise} \end{cases} .$$

Now the splitting rule consists simply in finding the binary attribute $a_*^{x_*}$ minimizing $H(\lambda|a_j^{x_{j_s}})$, where a_* is the attribute for splitting and x_* is the splitting value, in symbol

$$x_* = \arg\min\{H(\lambda|a_j^{x_{j_s}}) \ : \ j = 1, \ldots, m, s = 1, \ldots, t_j - 1\}.$$

Then the local object set Ω_α is partitioned into two subsets, defined as

$$\{\omega_i \in \Omega_\alpha \ : \ a_*(\omega_i) \leq x_*\} \text{ and } \{\omega_i \in \Omega_\alpha \ : \ a_*(\omega_i) > x_*\}, \tag{21}$$

and the procedure is repeated on these two subsets.

We stop growing the tree in the case λ is constant on Ω_α, moreover, to avoid overfitting, three pre-pruning parameters determine further stopping conditions. The parameter $measureThreshold$ sets a lower bound for the discrimination measure H, the parameter $maxDepth$ sets the maximum length of a path from the root to a leaf node and the parameter $percMinSize$ sets the minimum size of the current object set Ω_α, which is computed as $percMinSize \cdot |\Omega|$. Notice that, since the discrimination measures H's that can be used in RDMT(H) have different range, the parameter $measureThreshold$ is deeply tied to the chosen measure and so it must be properly tuned.

Once a stopping condition is reached a leaf node is created and is properly labelled [17]. If λ is constant on Ω_α then the constant value is chosen as label, otherwise if λ is not constant, then the median value is taken. In the particular case λ assumes only two values $c_{l_1} < c_{l_2}$, both on the same number of objects of Ω_α, then c_{l_1} is chosen in the case of a left leaf node, while c_{l_2} is chosen in the case of a right leaf node.

Concerning the measures proposed in this paper, we underline that, generally, even if the training dataset is monotone consistent, the greedy tree induction with H_G^*, H_S^* and H_P^* (or, more generally, any rank discrimination measure H^* satisfying Definition 5.1) does not guarantee a globally monotone classifier.

Hence, it is important to investigate if RDMT(H^*) can assure at least a weaker form of monotonicity. Algorithm REMT is shown to guarantee a weak kind of monotonicity that we call *rule monotonicity* [17]. Let $\mathcal{T} = (N, A)$ be

the decision tree generated by the induction procedure, where $N = \{r\} \cup I \cup L$ is the set of nodes (partitioned in the singleton formed by the root r, the set of internal nodes I and the set of leaves L) and A is the set of directed arcs. It is known (see, e.g., [31]) that each path $r \rightsquigarrow l$ with $l \in L$ induces a decision rule R_l, thus we denote with $\mathcal{R}_\mathcal{T}$ the set of decision rules generated by $\mathcal{T}$.

Given $l_1, l_2 \in L$, R_{l_1} and R_{l_2} are *comparable* (see [17]) only in the case they are generated by the same attributes, in this case we say that $R_{l_1} < R_{l_2}$ if and only if attribute values of R_{l_1} are less than R_{l_2}. We simply denote with $\lambda_\mathcal{T}(l)$ the label attached to leaf node l. Then we say that $\mathcal{T}$ is *rule monotone* if and only if for each $R_{l_1}, R_{l_2} \in \mathcal{R}_\mathcal{T}$:

$$R_{l_1} < R_{l_2} \;\Rightarrow\; \lambda_\mathcal{T}(l_1) < \lambda_\mathcal{T}(l_2). \tag{22}$$

In [17] it is proven that in the case the dataset is monotone consistent, then algorithm REMT guarantees rule monotonicity. The following proposition states that the same also holds for RDMT(H^*), where H^* is an arbitrary rank discrimination measure.

Proposition 7.1. *Let $\mathcal{D} = \{(a_1(\omega_i), \ldots, a_m(\omega_i), \lambda(\omega_i)) \;:\; i = 1, \ldots, n\}$ be a dataset of examples and $\mathcal{T}$ a binary decision tree built with RDMT(H^*) on $\mathcal{D}$, where H^* is rank discrimination measure satisfying Definition 5.1. If $\mathcal{D}$ is monotone consistent (i.e., it satisfies (3)) then $\mathcal{T}$ is rule monotone.*

Proof. The proof follows by Theorem 5.1 proceeding on the same line of the proof of Property 1 in [17]. □

The following Example 7.1 shows the construction of a tree with RDMT(H) for different H's, starting from a monotone consistent dataset. A first construction is executed using standard discrimination measures H_G and H_S, and a second one is carried on using rank discrimination measures H^*_G, H^*_S and H^*_P. In the first case a non-globally monotone tree classifier is obtained while in the second case we get a rule monotone tree classifier which is also globally monotone.

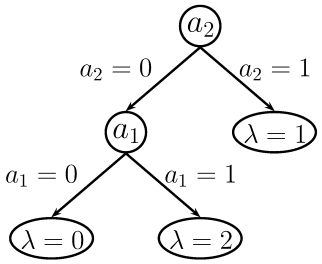
Example 7.1. *Consider the set of objects $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ described by attributes a_1, a_2 and a_3 ranging in $\{0, 1\}$, and the labelling function λ ranging in $\{0, 1, 2\}$, defined as in Table 8.*

	a_1	a_2	a_3	λ
ω_1	0	0	1	0
ω_2	0	1	0	1
ω_3	0	1	1	1
ω_4	1	0	1	2

Table 8: Definition of a_1, a_2, a_3 and λ

Considering measures H_G and H_S we compute $H_G(\lambda|a_1) = 0.3333$, $H_G(\lambda|a_2) = 0.25$, $H_G(\lambda|a_3) = 0.5$, $H_S(\lambda|a_1) = 0.6887$, $H_S(\lambda|a_2) = 0.5$ and $H_S(\lambda|a_3) =$

1.1887. *Then both measures agree to select a_2 for splitting. For the next step, the right branch allows to add a leaf node having label $\lambda = 1$, while for the left branch the only possible split is on a_1 for which $H_G(\lambda|a_1) = H_S(\lambda|a_1) = 0$, so also in this case both measures agree to select a_1 for splitting. At this point a left leaf node with label $\lambda = 0$ and a right leaf node with label $\lambda = 2$ are added. The resulting tree $\mathcal{T}_1$ is shown in Figure 1.*

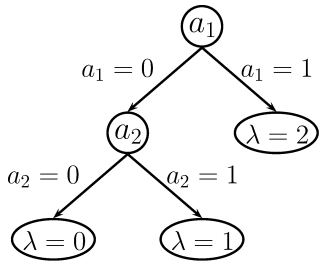


a_1	a_2	a_3	$\lambda_{\mathcal{T}_1}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	2
1	0	1	2
1	1	0	1
1	1	1	1

Figure 1: Non-globally monotone decision tree classifier

It is easily verified that the obtained classifier $\lambda_{\mathcal{T}_1} : X \to C$ is not globally monotone, since for example $(1,0,1) \leq (1,1,1)$ and $2 = \lambda_{\mathcal{T}_1}(1,0,1) > \lambda_{\mathcal{T}_1}(1,1,1) = 1$.

If we take into account measures H^*_G, H^*_S and H^*_P, we have $H^*_G(\lambda|a_1) = 0.125$, $H^*_G(\lambda|a_2) = 0.1875$, $H^*_G(\lambda|a_3) = 0.3125$, $H^*_S(\lambda|a_1) = 0.2075$, $H^*_S(\lambda|a_2) = 0.5$, $H^*_S(\lambda|a_3) = 0.6462$, $H^*_P(\lambda|a_1) = 0.4150$, $H^*_P(\lambda|a_2) = 4$ and $H^*_P(\lambda|a_3) = 3.7045$. Hence, all the measures agree to select a_1 for splitting. At this point, for the right branch it is possible to create a leaf node labelled with $\lambda = 2$, while for the left branch we compute $H^*_G(\lambda|a_2) = 0$, $H^*_G(\lambda|a_3) = 0.2777$, $H^*_S(\lambda|a_2) = 0$, $H^*_S(\lambda|a_3) = 0.5283$, $H^*_P(\lambda|a_2) = 0$ and $H^*_P(\lambda|a_3) = 1.6258$. Again all the measures agree to select a_2 for splitting and the construction is stopped adding a left leaf node labelled with $\lambda = 0$ and a right leaf node labelled with $\lambda = 1$. Figure 2 displays the built tree $\mathcal{T}_2$.



a_1	a_2	a_3	$\lambda_{\mathcal{T}_2}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	2
1	0	1	2
1	1	0	2
1	1	1	2

Figure 2: Globally monotone decision tree classifier

In this second case, the obtained tree is easily seen to be rule monotone,

moreover, the corresponding classifier $\lambda_{\mathcal{T}_2} : X \to C$ is also globally monotone.

Let us stress that Proposition 7.1 provides only a sufficient condition for the rule monotonicity of a binary decision tree $\mathcal{T}$. In fact, one can observe that also the first decision tree built in previous example (by using H_S and H_G) is rule monotone.

8. Quantification of non-monotonicity

We already mentioned that no hypothesis of monotonicity is asked on the datasets we deal with in this paper. At the same time, the rank discrimination measures we proposed do not allow, in general, to guarantee a globally monotone classifier even in presence of a monotone consistent dataset.

In this environment, in order to provide meaningful comparisons, it is of crucial importance to quantify to which degree a dataset is not monotone consistent or a tree is not globally monotone.

The quantification of non-monotonicity in a dataset has been recently investigated in [28]. A dataset $\mathcal{D} = \{(a_1(\omega_i), \ldots, a_m(\omega_i), \lambda(\omega_i)) \ : \ i = 1, \ldots, n\}$ is not monotone consistent if it contains at least a pair $\omega_i, \omega_h \in \Omega$ satisfying one of the following conditions:

(i) $a_1(\omega_i), \ldots, a_m(\omega_i)) \leq (a_1(\omega_h), \ldots, a_m(\omega_h))$ and $\lambda(\omega_i) > \lambda(\omega_h)$;

(ii) $a_1(\omega_i), \ldots, a_m(\omega_i)) \geq (a_1(\omega_h), \ldots, a_m(\omega_h))$ and $\lambda(\omega_i) < \lambda(\omega_h)$.

For each pair $\omega_i, \omega_h \in \Omega$ we denote with $NMP(\omega_i, \omega_h)$ the function which is 1 if *(i)* or *(ii)* are satisfied and 0 otherwise. Notice that $NMP(\omega_i, \omega_i) = 0$ for $i = 1, \ldots, n$.

In [28] the following index of non-monotonicity is proposed (in our notation)

$$NMI1(\mathcal{D}) = \frac{\sum_{i=1}^{n} \sum_{h=1}^{n} NMP(\omega_i, \omega_h)}{n^2 - n}, \tag{23}$$

which is easily seen to be a number in $[0, 1]$, equal to 0 whenever the dataset is monotone consistent. In next sections we remove the reference to the dataset $\mathcal{D}$, if the dataset we are referring to is clear from the context.

Now we cope with measuring non-monotonicity in trees, focusing on binary decision trees with discrete ordinal attributes, which are the target of present work.

Given such a tree $\mathcal{T}$, the corresponding set $L = \{l_1, \ldots, l_q\}$ of leaves induces a partition of the description space X generated by the X_j's. It is known [30] that such a partition can be expressed as a collection of disjoint closed intervals $\{[a_1, b_1], \ldots, [a_q, b_q]\}$, where $[a_u, b_u] = \{x \in X \ : \ a_u \leq x \leq b_u\}$ and $a_u, b_u \in X$, $u = 1, \ldots, q$. With a little abuse of notation we identify the leaf l_u with the corresponding interval $[a_u, b_u]$, moreover, we denote $\min(l_u) = a_u$ and $\max(l_u) = b_u$, while $\lambda_{\mathcal{T}}(l_u)$ stands for the label of leaf l_u. A tree $\mathcal{T}$ is not monotone if it contains at least a pair $l_u, l_v \in L$ satisfying one of the following conditions:

(i') $\min(l_u) < \max(l_v)$ and $\lambda_{\mathcal{T}}(l_u) > \lambda_{\mathcal{T}}(l_v)$;

(ii') $\max(l_u) > \min(l_v)$ and $\lambda_{\mathcal{T}}(l_u) < \lambda_{\mathcal{T}}(l_v)$.

This allows to build a symmetric $q \times q$ *non-monotonicity matrix* $M = [m_{u,v}]$ where $m_{u,v}$ is 1 if leaves l_u, l_v satisfy *(i')* or *(ii')* and 0 otherwise.

In [2] the following *non-monotonicity index* $I(\mathcal{T})$ has been introduced in order to quantify the non-monotonicity of $\mathcal{T}$

$$I(\mathcal{T}) = \frac{\sum_{u=1}^{q} \sum_{v=1}^{q} m_{u,v}}{q^2 - q}. \tag{24}$$

In next sections we remove the reference to the tree $\mathcal{T}$, if the tree we are referring to is clear from the context.

It is easily verified that $I(\mathcal{T})$ ranges in $[0, 1]$, and is 0 whenever the tree gives rise to a globally monotone classifier. In the same paper, the following transformation of $I(\mathcal{T})$ is proposed with the name *order-ambiguity score*

$$A(\mathcal{T}) = \begin{cases} 0 & \text{if } I(\mathcal{T}) = 0, \\ +\infty & \text{if } I(\mathcal{T}) = 1, \\ -\frac{1}{\log_2(I(\mathcal{T}))} & \text{otherwise.} \end{cases} \tag{25}$$

The function $A(\mathcal{T})$ will play an important role in next section.

Example 8.1. *Consider the tree $\mathcal{T}_1$ displayed in Figure 1. To simplify notation we denote a vector in the description space X by juxtaposing its elements. The tree $\mathcal{T}_1$ induces the partition $\{[000, 001], [100, 101], [010, 111]\}$ of X, with labels $\lambda_{\mathcal{T}_1}([000, 001]) = 0$, $\lambda_{\mathcal{T}_1}([100, 101]) = 2$ and $\lambda_{\mathcal{T}_1}([010, 111]) = 1$. In order to compute $I(\mathcal{T}_1)$ and $A(\mathcal{T}_1)$ we introduce the non-monotonicity matrix M*

		$\lambda = 0$ $[000, 001]$	$\lambda = 2$ $[100, 101]$	$\lambda = 1$ $[010, 111]$
$\lambda = 0$	$[000, 001]$	0	0	0
$\lambda = 2$	$[100, 101]$	0	0	1
$\lambda = 1$	$[010, 111]$	0	1	0

for which $I(\mathcal{T}_1) = \frac{2}{3^2-3} = \frac{1}{3}$ and $A(\mathcal{T}_1) = -\frac{1}{\log_2(\frac{1}{3})}$.

We stress that, since both indices $NMI1$ and I range in the unit interval, they can be conveniently expressed as percentages.

9. Other splitting rules for monotone decision trees

In the last decades, there have been some proposals concerning monotone classification with decision trees. The most known methods are the Monotone Induction of Decision trees (MID) algorithm [2], the Isotonic Classification Tree (ICT) algorithm [34], the Positive Decision Tree (P-DT) algorithm [23] and

the Monotone Decision Tree (MDT) algorithm [29, 30] (see [7] for a careful discussion of quoted algorithms).

Each of the above algorithms relies on some assumptions on the input dataset which essentially influence the result of the classifier in terms of monotonicity. Moreover, all of them have to face the intrinsic problem of partial observability during tree induction, either during splitting or by adding exogenous information via updating rules.

P-DT and MDT require a monotone consistent dataset and guarantee a globally monotone classifier in output: P-DT is limited to two-class problems while MDT has not this limitation. P-DT uses a modified version of binary Shannon entropy (due to the hypothesis of two-class problems), while MDT relies on standard discrimination measures. Both algorithms are essentially based on an updating rule which alters the input dataset by adding artificial data with the only purpose of enforcing monotonicity.

MID and ICT, instead, are able to work with non-monotone datasets but MID does not guarantee any form of monotonicity on the final classifier, while ICT always results in a globally monotone classifier. They differ in the discrimination measure used for growing trees, which in both cases tries to enforce "somehow" monotonicity. Furthermore, ICT post-processes the resulting tree coping with relabelling of non-monotone leaves.

Since we do not limit to two-class problems, discrimination measures used in MID and ICT represent possible alternatives to rank discrimination measures presented in this paper, besides classical discrimination measures used in MDT.

The MID algorithm is based on the following discrimination measure called *total ambiguity score* (in our notation)

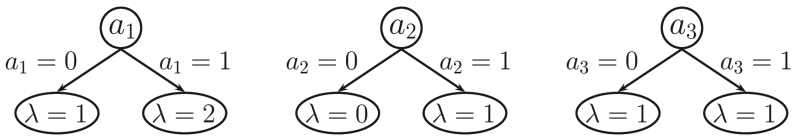
$$H^{\gamma}_{MID}(\lambda|a_j) = H_S(\lambda|a_j) + \gamma \cdot A(\mathcal{T}^{a_j}_{\alpha}) \tag{26}$$

where $\gamma \in [0, +\infty)$ and $A(\mathcal{T}^{a_j}_{\alpha})$ is the order-ambiguity score (25) evaluated on the tree $\mathcal{T}^{a_j}_{\alpha}$, which is obtained by adding to the current tree $\mathcal{T}_{\alpha}$ the test node a_j and labelling the terminal nodes (if not already labelled).

The H^{γ}_{MID} measure is essentially tied to the choice of the parameter γ and the labelling rule adopted in the formation of $\mathcal{T}^{a_j}_{\alpha}$. We stress that, even if H^{γ}_{MID} takes into account all the "history" that led to the tree $\mathcal{T}^{a_j}_{\alpha}$, this measure is deeply influenced by the order of expansion of the tree nodes which is, evidently, a cause of partial observability. Previous claim implies that, also H^{γ}_{MID} is not able to enforce global monotonicity and, moreover, also the enforcing of local monotonicity is not guaranteed. The next example shows that, starting from a monotone consistent dataset, H^{γ}_{MID} produces a non-globally monotone classifier while each of our rank discrimination measures produces a globally monotone classifier.

Example 9.1. *Consider the dataset of Example 7.1 and denote with $\mathcal{T}_0$ the empty tree. In order to compute $H^{\gamma}_{MID}(\lambda|a_j)$, for $j = 1, 2, 3$, we need to consider the possible trees $\mathcal{T}^{a_j}_0$ obtainable by adding a test node involving a_j, for $j = 1, 2, 3$. By considering the RDMT's labelling rule we get the trees reported in*

Figures 3a–3c. Let us stress that the obtained labelling is consistent also with the majority class criterion.



(a) Tree $\mathcal{T}_0^{a_1}$ (b) Tree $\mathcal{T}_0^{a_2}$ (c) Tree $\mathcal{T}_0^{a_3}$

Figure 3: Trees $\mathcal{T}_0^{a_j}$ for computing $H^{\gamma}_{MID}(\lambda|a_j)$, for $j = 1, 2, 3$

It is easily verified that $A(\mathcal{T}_0^{a_j}) = I(\mathcal{T}_0^{a_j}) = 0$, *for* $j = 1, 2, 3$, *thus for every* $\gamma \in [0, +\infty)$ *it holds* $H^{\gamma}_{MID}(\lambda|a_1) = H_S(\lambda|a_1) = 0.6887$, $H^{\gamma}_{MID}(\lambda|a_2) = H_S(\lambda|a_2) = 0.5$ *and* $H^{\gamma}_{MID}(\lambda|a_3) = H_S(\lambda|a_3) = 1.1887$. *This implies that the attribute* a_2 *is selected for splitting. Trivial computations show that the final tree built using* H^{γ}_{MID} *coincides with the non-globally monotone tree built using either* H_S *or* H_G, *which is reported in Figure 1.*

Another ensuing problem with the H^{γ}_{MID} measure is the estimation of parameter γ. In a naive analysis one could argue that the higher the value of γ, the more monotone is the final tree. Nevertheless, the following example shows that a higher value of γ can lead to worse results in terms of monotonicity.

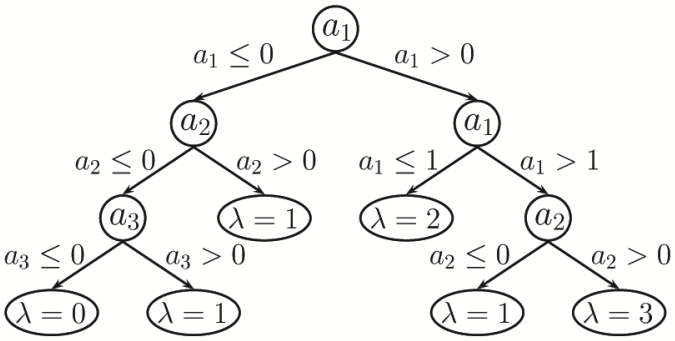
Example 9.2. *Consider the object set* $\Omega = \{\omega_1, \ldots, \omega_9\}$ *described by the attributes* a_1, a_2, a_3 *ranging in* $\{0, 1, 2\}$ *and the labelling function* λ *ranging in* $\{0, 1, 2, 3\}$, *defined as in Table 9. Applying RDMT(*H^{γ}_{MID}*) with* $\gamma = 1$, *the tree*

	a_1	a_2	a_3	λ
ω_1	0	0	0	0
ω_2	0	0	1	1
ω_3	1	1	1	2
ω_4	2	0	2	1
ω_5	2	1	1	3
ω_6	0	1	0	1
ω_7	0	2	0	1
ω_8	1	1	0	2
ω_9	2	1	0	3

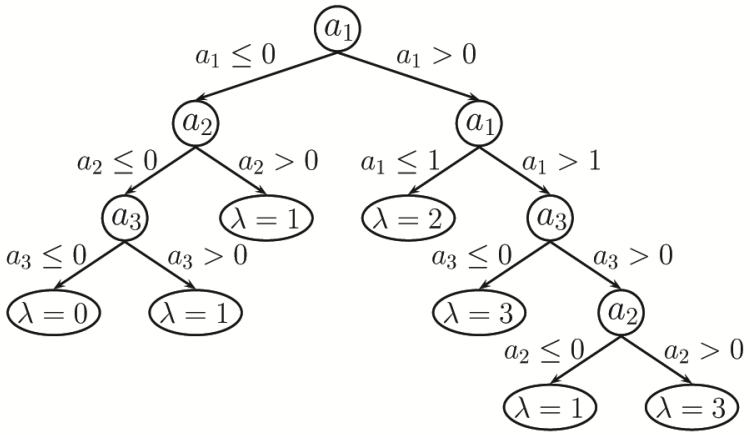
Table 9: Definition of a_1, a_2, a_3 and λ

$\mathcal{T}_1$ *shown in Figure 4a is obtained, while for* $\gamma \in \{10, 100, 100\}$ *we get the tree* $\mathcal{T}_2$ *shown in Figure 4b.*

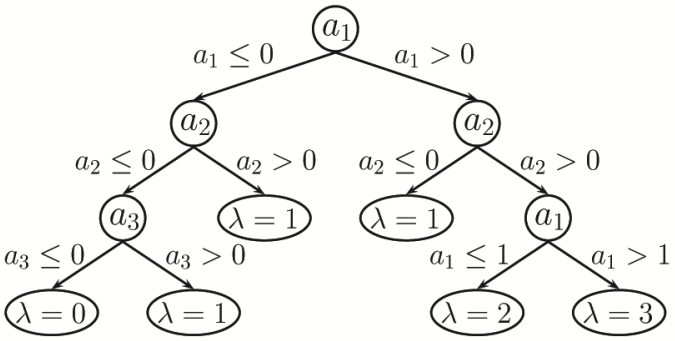
Even if the dataset is monotone consistent, both $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *are non-globally monotone, nevertheless, it holds* $I(\mathcal{T}_1) = 0.0666 < 0.0952 = I(\mathcal{T}_2)$, *i.e.,* $\mathcal{T}_1$ *can be considered better than* $\mathcal{T}_2$ *in terms of monotonicity. On the other hand, applying RDMT(*H^*_G*) we get the globally monotone tree* $\mathcal{T}_3$ *shown in Figure 4c, for which we have* $I(\mathcal{T}_3) = 0$.



(a) Tree $\mathcal{T}_1$ with $I(\mathcal{T}_1) = 0.0666$, built using H^{γ}_{MID} for $\gamma = 1$



(b) Tree $\mathcal{T}_2$ with $I(\mathcal{T}_2) = 0.0952$, built using H^{γ}_{MID} for $\gamma \in \{10, 100, 1000\}$



(c) Tree $\mathcal{T}_3$ with $I(\mathcal{T}_3) = 0$, built using H^{*}_{G}

Figure 4: Trees $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$

Assuming that a_j is a binary attribute ranging in $\{x_{j_1}, x_{j_2}\}$ and that the set of classes $C = \{c_1, \ldots, c_k\}$ is indexed by $\{1, \ldots, k\}$, the Gini discrimination measure can be expressed as

$$H_G(\lambda|a_j) = \sum_{s=1}^{2} p_s \left(1 - \sum_{q=1}^{k} \left(\frac{p_{q,s}}{p_s}\right)^2\right) = \sum_{s=1}^{2} p_s \left(\sum_{u \neq v} \left(\frac{p_{u,s}}{p_s}\right)\left(\frac{p_{v,s}}{p_s}\right)\right).$$

The ICT algorithm builds binary trees and is based on a modified version of the Gini measure, incorporating the L_1 distance which is referred to as $Gini_{L_1}$ *impurity* [34]. Here we report the conditional version of this measure (in our notation)

$$H_{ICT}(\lambda|a_j) = \sum_{s=1}^{2} p_s \left(\sum_{u \neq v} |u - v| \left(\frac{p_{u,s}}{p_s}\right)\left(\frac{p_{v,s}}{p_s}\right)\right) \tag{27}$$

Remark 9.1. *In [34] the splitting criterion consists in maximizing the gain in $Gini_{L_1}$ impurity. Notice that this is equivalent to minimize H_{ICT}.*

The following example shows that H_{ICT} can produce a globally non-monotone tree classifier even in presence of a monotone consistent dataset, for which also rule monotonicity may fail.

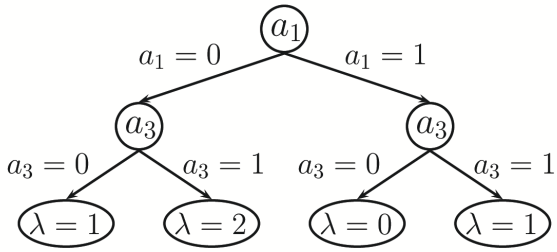
Example 9.3. *Consider the object set $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ described by attributes a_1, a_2 and a_3 ranging in $\{0, 1\}$, and the labelling function λ all ranging in $\{0, 1, 2\}$, defined as in Table 10. The corresponding dataset is monotone consistent.*

	a_1	a_2	a_3	λ
ω_1	0	1	0	1
ω_2	0	1	1	2
ω_3	1	0	0	0
ω_4	1	0	1	1

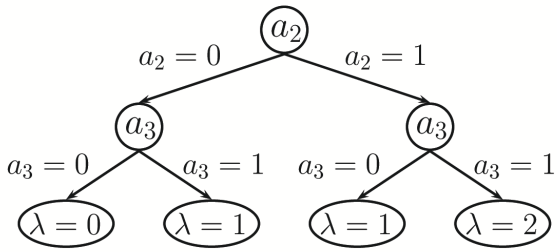
Table 10: Definition of a_1, a_2, a_3, and λ

Using H_{ICT} in RDMT for splitting we have $H_{ICT}(\lambda|a_1) = H_{ICT}(\lambda|a_2) = H_{ICT}(\lambda|a_3) = 0.5$, i.e., all attributes are judged the same thus the first is selected. By trivial computations it follows that the final result is the tree $\mathcal{T}_1$ shown in Figure 5a which is not globally monotone as $I(\mathcal{T}_1) = 0.3333$. It is easily seen that $\mathcal{T}_1$ is neither rule monotone since, denoting with $l_1, \ldots, l_4$ its leaves from left to right, it holds $R_{l_1} < R_{l_3}$ and $\lambda_{\mathcal{T}_1}(l_1) > \lambda_{\mathcal{T}_1}(l_3)$.

On the contrary, using H_G^*, H_S^* or H_P^* the rule monotone tree $\mathcal{T}_2$ in Figure 5b is obtained, which is also globally monotone.



(a) Tree $\mathcal{T}_1$ with $I(\mathcal{T}_1) = 0.3333$, built using H_{ICT}



(b) Tree $\mathcal{T}_2$ with $I(\mathcal{T}_2) = 0$, built using H^*_G, H^*_S, or H^*_P

Figure 5: Trees $\mathcal{T}_1$ and $\mathcal{T}_2$

10. Experimental analysis

The goal of this section is to compare the splitting criteria obtained using H^*_G, H^*_S and H^*_S, with other splitting criteria relying on different discrimination measures.

More in detail, we are interested in evaluating the impact of the sole splitting criteria on the final classifiers both in terms of accuracy and monotonicity. We also want to investigate the response of the different measures to datasets with increasing degree of non-monotonicity.

We used RDMT(H) varying the discrimination measure H among H^*_G, H^*_S, H^*_P, H_G, H_S, H^{10}_{MID} and H_{ICT}, on classification tasks involving artificial and real datasets. Recall that for different discrimination measures H_1 and H_2, RDMT(H_1) and RDMT(H_2) can be formally viewed as different algorithms.

Remark 10.1. *For H^{λ}_{MID} the value* 10 *has been chosen for λ: this value revealed to be a good compromise between accuracy and monotonicity in some preliminary tests we did.*

Each test has been executed performing a stratified 10-folds cross-validation with the same seed (equal to 1) for the pseudo-random number generator: the WEKA environment guarantees all the folds are equal for each tested discrimination measure.

For each test we measured the classification accuracy, the monotonicity and the size of the resulting trees in the 10 folds.

The classification accuracy has been measured through the percentage of Correctly Classified Instances (denoted CCI for short), the Kappa statistic (denoted K for short), and the Mean Absolute Error (denoted MAE for short).

Remark 10.2. *Recall that K ranges in $[-1, 1]$ and is a measure of accuracy corrected for random successes [4]. In detail, a classifier is as much more accurate as K is close to 1.*

For what concerns the measurement of monotonicity of the generated trees, we considered both the non-monotonicity index I of trees, and the $NMI1$ index of the classified instances on each test set: both this two indices have been averaged through the 10 folds and the resulting performance indices have been denoted as $avgI$ and $avgNMI1$.

We also measured the size of trees in terms of number of leaves, averaging through the 10 folds: the resulting performance index has been denoted as $avgLeaves$.

In order to execute a fair comparison between the tested measures we set $maxDepth = 100$, $measureThreshold = 0$ and $percMinSize = 0.01$ for all the tests. Indeed, since measures H^*_G, H^*_S, H^*_P, H_G, H_S, H^{10}_{MID} and H_{ICT} have different ranges, setting $measureThreshold > 0$ could favour some measures and penalize the others.

Remark 10.3. *The decision to set $measureThreshold = 0$ came after a preliminary experimental analysis where we observed that this parameter has a significant impact on the performance of the classifier, depending both on the chosen dataset and the chosen discrimination measure. Moreover, we verified that its tuning is particularly difficult in a comparative study as the one presented in this section. Indeed, for suitable choices of this parameter (possibly different values for the different discrimination measures) it is possible to produce a better behaviour of some measures to the detriment of the others. This unpleasant effect could render the comparison quite arbitrary, so we decided to turn off such parameter. We want to stress that measureThreshold is a pre-pruning parameter of RDMT(H) which acts on the different H's while the parameter γ discussed in previous section is an intrinsic parameter of the measure H^{γ}_{MID}.*

The main difficulty when comparing monotone classifiers is the fact that real datasets are generally not monotone consistent and, furthermore, it is difficult to find a real dataset with a specified value of $NMI1$ index. This is why in [28] an algorithm to generate artificial datasets with a fixed value of $NMI1$ index is introduced. The same authors suggest to use the proposed algorithm as a test bed for comparing different monotone classification algorithms on sensitivity to different degrees of non-monotone noise.

The algorithm in [28] essentially consists in generating n m-tuples of values for attributes $a_1, \ldots, a_m$, where each a_j is assumed to be a uniform random variable on the first v_j non-negative integers, i.e., ranging in $\{0, \ldots, v_j - 1\}$.

The n m-tuples are then ordered according to the value of a monotone function $f(a_1, \ldots, a_j)$. After that, considering the first c non-negative integers as class labels, i.e., the set $\{0, \ldots, c-1\}$, the n m-tuples are divided in n/c groups according to the ordering given by f and an increasing class label is associated to the elements of each group. This assures a balanced distribution of class labels. The obtained dataset is then manipulated until it is monotone consistent, i.e., the corresponding $NMI1$ is equal to 0. This is done with an iterative search of non-monotone labels of pairs of m-tuples which are resolved by a class relabelling. Finally, a further iterative manipulation introduces non-monotone noise through non-monotone class relabelling of pairs of m-tuples, until the desired value of $NMI1$ is reached.

We used the algorithm reported in [28] to generate 11 artificial datasets with increasing $NMI1$ from 0% to 10% with 1% steps. Each dataset consists of 500 examples described by five attributes a_1, a_2, a_3, a_4, a_5 ranging in $\{0, \ldots, 4\}$, and classes also taken from $\{0, \ldots, 4\}$. The underlying monotone function is $f(a_1, a_2, a_3, a_4, a_5) = \sum_{j=1}^{5} a_j$.

Figure 6 displays plots of CCI, K and MAE for each tested discrimination measure. Observing Figure 6 it is possible to see a general decrease in performance of all measures for CCI, K and MAE as $NMI1$ increases. The decrease is quite fast in the first values of $NMI1$ and then it tends to a much slower rate until reaching a sort of stability with some fluctuations. We stress that in plots related to CCI, K and MAE there is no measure whose performance dominates the others. Hence, tests on artificial datasets do not single out any significant difference between the tested measures for what concerns accuracy indices.

Figure 7 displays plots of $avgI$, $avgNMI1$ and $avgLeaves$ for each tested discrimination measure. Also in this case the performances concerning $avgI$ and $avgNMI1$ decrease as $NMI1$ increases (recall that, the higher the value of $avgI$ and $avgNMI1$ the worse are the generated trees in terms of monotonicity). In this case, $avgI$ and $avgNMI1$ tend to increase almost linearly with $NMI1$: this clearly shows the link of this two performance indices with non-monotone noise in the dataset. Nevertheless, H_S^*, H_G^* and H_P^* are always dominated by other measures for $avgI$ and a similar phenomenon realizes also for $avgNMI1$. For $avgNMI1$, in particular, H_S^* and H_G^* are quite far from other measures. This puts in evidence that in the aforementioned tests our measures always produced better trees in terms of monotonicity, especially for increasing $NMI1$.

The plot of $avgLeaves$ shows that our measures always dominate the others in average number of leaves, thus a relation between $avgLeaves$, $avgI$ and $avgNMI1$ comes to the fore. We notice that, also for $avgLeaves$ the value of the index increases rapidly on the first values of $NMI1$ and then has a much slower rate until reaching a sort of stability with some fluctuations.

We conclude this section reporting results of tests executed on real datasets taken from UCI [33] and WEKA [37] repositories. We pre-processed all the datasets, removing all the examples with missing values and discretizing real attributes applying WEKA filter `Discretize`, moreover, integer attributes have

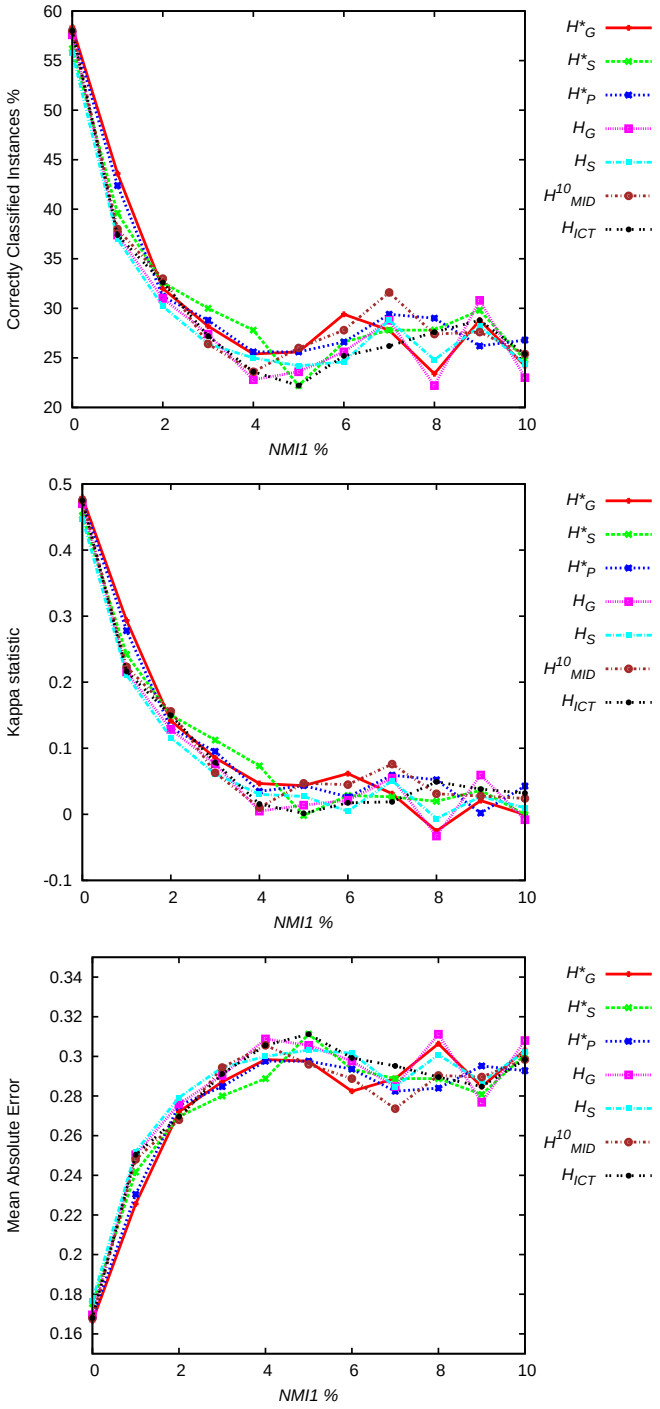


Figure 6: CCI, K, and MAE on artificial data with increasing $NMI1$

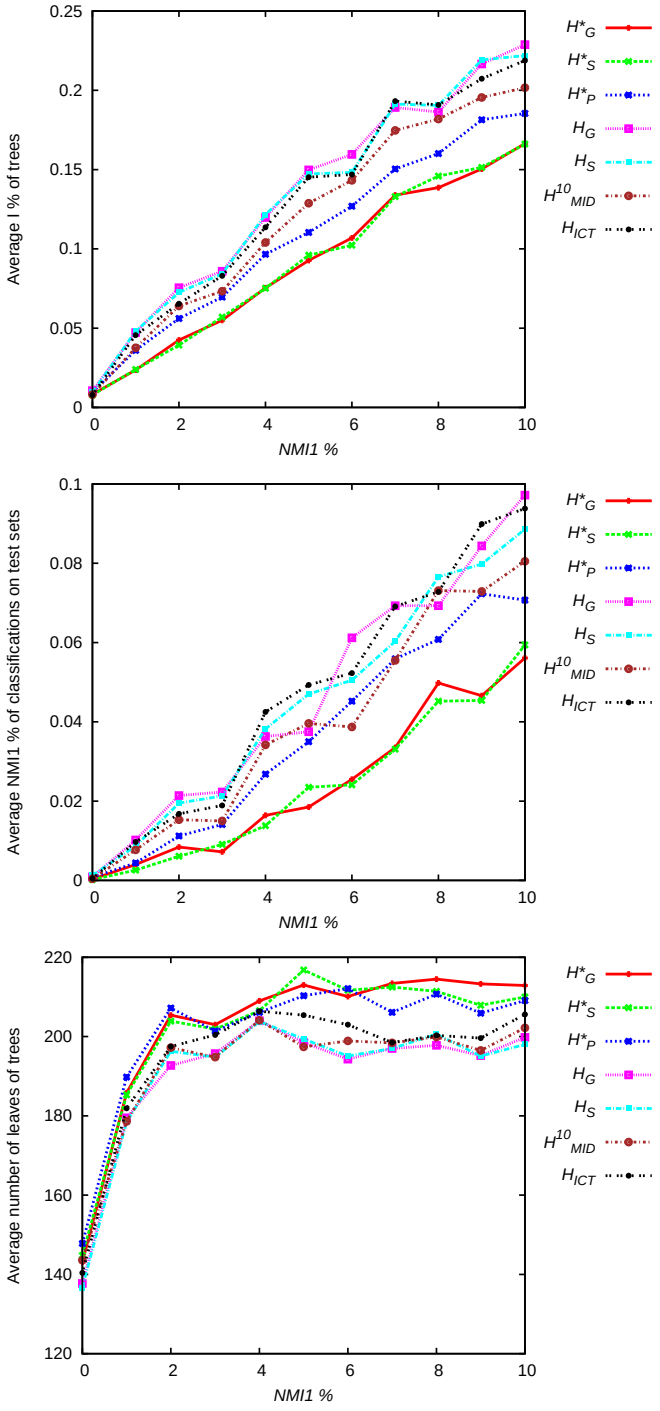


Figure 7: $avgI$, $avgNMI1$ and $avgLeaves$ on artificial data with increasing $NMI1$

been converted to ordinal ones by using WEKA filter `NumericToNominal`, both contained in `weka.filters.unsupervised.attribute`.

We collected 15 datasets with $NMI1$ less than 12%, whose description (taking into account the pre-processing phase) is given in Table 11.

Dataset	Instances	Attributes	Classes	$NMI1$
Breast Cancer	277	9	2	0.6958%
Contraceptive Method Choice (CMC)	1473	9	3	2.5875%
Contact Lenses	24	4	3	11.9565%
CPU	209	6	8	0.2300%
German Credit	1000	20	2	0.4904%
Dermatology	358	34	6	0.1564%
Employee Rejection Acceptance (ERA)	1000	4	9	3.3493%
Employee Selection (ESL)	488	4	9	0.9467%
Haberman	306	3	2	3.6129%
Lectures Evaluation (LEV)	1000	4	5	1.3309%
Lymphography	148	18	4	0.8273%
Monks-1 Train	124	6	2	5.2058%
Monks-3 Train	122	6	2	9.7412%
Postoperative	87	8	3	5.6134%
Social Workers Decisions (SWD)	1000	10	4	0.9491%

Table 11: Tested real datasets

Tables 12 and 13 show the obtained results for tests on real datasets. Our goal is to establish if for each performance index, the tested discrimination measures produce statistically different results.

For this, the performance rank of each discrimination measure with respect to each performance index is reported in square brackets. According to [11], the variance of the value of each performance index (in case the index is an average, as $avgI$, $avgNMI1$ and $avgLeaves$) is not recorded.

In Table 14 are listed the average performance ranks of each tested discrimination measure, with respect to the considered performance indices. As pointed out in [11], the average performance ranks themselves provide a fair comparison of the different tested measures. Indeed, observing the average performance ranks we can see that standard measures such as H_G and H_S perform better for what concerns accuracy, both positioning in the first two places for CCI, K and MAE, while H_P^* occupies the third position for CCI and MAE. Nevertheless, the differences of the average performance ranks for CCI, K and MAE do not largely differ among the tested measures. Hence, we could argue that the tested measures are not significantly different for what concerns accuracy.

On the other hand, in the performance indices related to monotonicity, our measures H_S^*, H_G^* and H_P^* rank in first, second and fourth positions for $avgI$, and in the first three positions for $avgNMI1$. We also observe that both for $avgI$ and $avgNMI1$ the differences in the average performance ranks is quite large among the tested measures so, in this case, we could argue that our measures

Dataset		H^*_G	H^*_S	H^*_P	H_G	H_S	H^{10}_{MID}	H_{ICT}
Breast Cancer	CCI	67.148% [2.5]	67.509% [1]	67.148% [2.5]	66.4259% [4]	65.7039% [6]	62.8158% [7]	66.0649% [5]
	K	0.2032 [3]	0.2148 [1]	0.1974 [4]	0.2087 [2]	0.1621 [6]	0.1358 [7]	0.1973 [5]
	MAE	0.3285 [2.5]	0.3249 [1]	0.3285 [2.5]	0.3357 [4]	0.3429 [6]	0.3718 [7]	0.3393 [5]
	$avgI$	0.0575% [1]	0.0609% [2]	0.0732% [3]	0.1311% [7]	0.1211% [5]	0.0936% [4]	0.1302% [6]
	$avgNMI1$	0.0018% [2.5]	0.0018% [2.5]	0.0013% [1]	0.004% [7]	0.0023% [4]	0.0026% [5]	0.0037% [6]
	$avgLeaves$	123.6 [7]	121.1 [6]	107.7 [5]	90.4 [3]	88.1 [1]	97 [4]	90.3 [2]
CMC	CCI	46.2321% [7]	46.3% [6]	46.6395% [5]	51.7311% [1]	51.5274% [2]	49.7623% [4]	49.8302% [3]
	K	0.1914 [7]	0.1957 [5]	0.1922 [6]	0.2642 [1]	0.2603 [2]	0.2381 [3]	0.2372 [4]
	MAE	0.3584 [7]	0.3579 [6]	0.3557 [5]	0.3217 [1]	0.3231 [2]	0.3349 [4]	0.3344 [3]
	$avgI$	0.0758% [1]	0.0782% [2]	0.1109% [3]	0.1956% [5]	0.1984% [6]	0.1129% [4]	0.2005% [7]
	$avgNMI1$	0.0087% [2]	0.0086% [1]	0.0125% [3]	0.0235% [6]	0.0242% [7]	0.0191% [4]	0.0233% [5]
	$avgLeaves$	234.6 [5]	242.9 [7]	237.6 [6]	187.2 [2]	185.2 [1]	192.9 [4]	188 [3]
Contact Lenses	CCI	62.5% [5]	62.5% [5]	62.5% [5]	83.3333% [2]	83.3333% [2]	58.3333% [7]	83.3333% [2]
	K	0.4016 [4.5]	0.4016 [4.5]	0.325 [6]	0.6903 [2]	0.6903 [2]	0.2258 [7]	0.6903 [2]
	MAE	0.25 [5]	0.25 [5]	0.25 [5]	0.1111 [2]	0.1111 [2]	0.2777 [7]	0.1111 [2]
	$avgI$	0.1732% [1]	0.1763% [2]	0.1853% [4]	0.2899% [6]	0.2899% [6]	0.1838% [3]	0.2899% [6]
	$avgNMI1$	0.1666% [2.5]	0.1666% [2.5]	0.1333% [1]	0.2666% [6]	0.2666% [6]	0.2333% [4]	0.2666% [6]
	$avgLeaves$	17.1 [6]	17.2 [7]	14.8 [4]	6.6 [2]	6.6 [2]	14.9 [5]	6.6 [2]
CPU	CCI	83.732% [2]	82.7751% [4.5]	84.6889% [1]	82.7751% [4.5]	81.8181% [7]	82.7751% [4.5]	82.7751% [4.5]
	K	0.5793 [2]	0.5512 [6]	0.611 [1]	0.5652 [4]	0.5413 [7]	0.5654 [3]	0.561 [5]
	MAE	0.0325 [2]	0.0344 [4.5]	0.0306 [1]	0.0344 [4.5]	0.0363 [7]	0.0344 [4.5]	0.0344 [4.5]
	$avgI$	0.0196% [2]	0.0162% [1]	0.0328% [4]	0.0407% [7]	0.0361% [5]	0.0253% [3]	0.0396% [6]
	$avgNMI1$	0.0004% [3.5]	0.0004% [3.5]	0% [1.5]	0.001% [7]	0.0009% [5.5]	0% [1.5]	0.0009% [5.5]
	$avgLeaves$	36.6 [6]	37 [7]	34.4 [1]	35.6 [4]	34.8 [2]	35.7 [5]	34.9 [3]
German Credit	CCI	62.7% [6]	63.3% [5]	60.6% [7]	71.5% [1]	70.1% [3]	67% [4]	71.3% [2]
	K	0.0937 [6]	0.1083 [5]	0.0794 [7]	0.3271 [1]	0.2968 [3]	0.2216 [4]	0.3224 [2]
	MAE	0.373 [6]	0.367 [5]	0.394 [7]	0.285 [1]	0.299 [3]	0.33 [4]	0.287 [2]
	$avgI$	0.0833% [1]	0.0934% [2]	0.1207% [3]	0.2251% [7]	0.2146% [5]	0.1415% [4]	0.2248% [6]
	$avgNMI1$	0.0003% [2]	0.0001% [1]	0.0019% [3]	0.0038% [6]	0.0042% [7]	0.0033% [4]	0.0038% [5]
	$avgLeaves$	188.6 [7]	186.3 [6]	158.7 [5]	120.8 [1]	124.6 [3]	136.5 [4]	120.9 [2]
Dermatology	CCI	89.3854% [6]	84.6368% [7]	93.5754% [3]	94.4134% [1]	93.8547% [2]	92.1787% [5]	93.296% [4]
	K	0.867 [6]	0.8069 [7]	0.9192 [3]	0.9298 [1]	0.923 [2]	0.9017 [5]	0.9159 [4]
	MAE	0.0353 [6]	0.0512 [7]	0.0214 [3]	0.0186 [1]	0.0204 [2]	0.026 [5]	0.0223 [4]
	$avgI$	0.172% [3]	0.1612% [1]	0.2238% [4]	0.3532% [6]	0.3127% [5]	0.1699% [2]	0.3663% [7]
	$avgNMI1$	0.0009% [3]	0.0003% [1]	0.0009% [3]	0.0011% [6]	0.0011% [6]	0.0009% [3]	0.0011% [6]
	$avgLeaves$	26 [5]	56.2 [7]	25.4 [4]	16.3 [1]	18.1 [2]	30.9 [6]	18.5 [3]
ERA	CCI	23% [4]	23% [4]	23% [4]	23% [4]	23% [4]	23% [4]	23% [4]
	K	0.0904 [4]	0.0904 [4]	0.0904 [4]	0.0904 [4]	0.0904 [4]	0.0904 [4]	0.0904 [4]
	MAE	0.1711 [4]	0.1711 [4]	0.1711 [4]	0.1711 [4]	0.1711 [4]	0.1711 [4]	0.1711 [4]
	$avgI$	0.0068% [1]	0.0103% [2]	0.061% [7]	0.0223% [4]	0.0223% [5]	0.0192% [3]	0.026% [6]
	$avgNMI1$	0.0007% [4]	0.0007% [4]	0.0007% [4]	0.0007% [4]	0.0007% [4]	0.0007% [4]	0.0007% [4]
	$avgLeaves$	44 [4]	44 [4]	44 [4]	44 [4]	44 [4]	44 [4]	44 [4]
ESL	CCI	68.2377% [1]	66.3934% [4.5]	65.9836% [6]	65.7786% [7]	67.0081% [2]	66.8032% [3]	66.3934% [4.5]
	K	0.6011 [1]	0.5777 [5]	0.5727 [6]	0.5705 [7]	0.5859 [2]	0.5828 [3]	0.5786 [4]
	MAE	0.0705 [1]	0.0746 [4.5]	0.0755 [6]	0.076 [7]	0.0733 [2]	0.0737 [3]	0.0746 [4.5]
	$avgI$	0.0092% [2]	0.0085% [1]	0.0102% [4]	0.0139% [7]	0.0128% [5]	0.0093% [3]	0.0129% [6]
	$avgNMI1$	0.0017% [3]	0.0014% [1]	0.0014% [2]	0.0033% [7]	0.0032% [6]	0.0029% [5]	0.0021% [4]
	$avgLeaves$	131 [6]	131.7 [7]	129.9 [5]	122.4 [1.5]	122.4 [1.5]	122.8 [3]	123.3 [4]

Table 12: Results concerning CCI, K, MAE, $avgI$, $avgNMI1$ and $avgLeaves$ of tests on first eight real datasets (performance ranks are reported in square brackets, averaging the ties)

Dataset		H^*_G	H^*_S	H^*_P	H_G	H_S	H^{10}_{MLD}	H_{ICT}
Haberman	CCI	70.915% [1]	70.5882% [2]	68.9542% [3]	64.7058% [4.5]	64.379% [6]	63.7254% [7]	64.7058% [4.5]
	K	0.2377 [2]	0.2384 [1]	0.173 [3]	0.1211 [4.5]	0.0956 [7]	0.1001 [6]	0.1211 [4.5]
	MAE	0.2908 [1]	0.2941 [2]	0.3104 [3]	0.3529 [4.5]	0.3562 [6]	0.3627 [7]	0.3529 [4.5]
	$avgI$	0.0783% [4]	0.0741% [3]	0.0648% [1]	0.0875% [6.5]	0.0856% [5]	0.0675% [2]	0.0875% [6.5]
	$avgNMI1$	0.0216% [2]	0.0203% [1]	0.0278% [4]	0.0318% [6.5]	0.0303% [5]	0.0258% [3]	0.0318% [6.5]
	$avgLeaves$	117.6 [6]	117.8 [7]	108.4 [5]	90.2 [1.5]	90.3 [3]	97.6 [4]	90.2 [1.5]
LEV	CCI	63.1% [3.5]	62.9% [6]	62.8% [7]	63.1% [3.5]	63.2% [2]	63.3% [1]	63% [5]
	K	0.4694 [3]	0.466 [6]	0.4649 [7]	0.4689 [4]	0.4702 [2]	0.4717 [1]	0.4675 [5]
	MAE	0.1476 [3.5]	0.1484 [6]	0.1488 [7]	0.1476 [3.5]	0.1472 [2]	0.1468 [1]	0.148 [5]
	$avgI$	0.005% [2]	0.0045% [1]	0.0141% [7]	0.0096% [5]	0.011% [6]	0.0055% [3]	0.0091% [4]
	$avgNMI1$	0.0003% [3]	0.0003% [6.5]	0.0003% [6.5]	0.0003% [3]	0.0003% [3]	0.0003% [3]	0.0003% [3]
	$avgLeaves$	91.6 [6]	91.4 [4]	91.9 [7]	90.8 [1]	91 [3]	91.5 [5]	90.9 [2]
Lymphography	CCI	72.9729% [7]	76.3513% [4]	80.4054% [1]	75.6756% [5.5]	78.3783% [2]	77.027% [3]	75.6756% [5.5]
	K	0.4896 [7]	0.5473 [4]	0.6236 [1]	0.5436 [5]	0.5833 [2]	0.5634 [3]	0.5271 [6]
	MAE	0.1351 [7]	0.1182 [4]	0.0979 [1]	0.1216 [5.5]	0.1081 [2]	0.1148 [3]	0.1216 [5.5]
	$avgI$	0.1003% [2]	0.0889% [1]	0.1293% [4]	0.2105% [7]	0.1587% [5]	0.1241% [3]	0.1725% [6]
	$avgNMI1$	0.001% [1.5]	0.001% [1.5]	0.002% [3]	0.0093% [5]	0.0166% [7]	0.008% [4]	0.0109% [6]
	$avgLeaves$	51.3 [6]	52.6 [7]	41.5 [5]	28.1 [2]	27.5 [1]	33.8 [4]	29.4 [3]
Monks-1 Train	CCI	70.9677% [6.5]	70.9677% [6.5]	77.4193% [5]	92.7419% [2]	92.7419% [2]	84.6774% [4]	92.7419% [2]
	K	0.4193 [6.5]	0.4193 [6.5]	0.5483 [5]	0.8548 [2]	0.8548 [2]	0.6935 [4]	0.8548 [2]
	MAE	0.2903 [6.5]	0.2903 [6.5]	0.2258 [5]	0.0725 [2]	0.0725 [2]	0.1532 [4]	0.0725 [2]
	$avgI$	0.1368% [1]	0.1375% [2]	0.1644% [3]	0.2404% [5.5]	0.2407% [7]	0.2034% [4]	0.2404% [5.5]
	$avgNMI1$	0.0177% [1.5]	0.0177% [1.5]	0.0277% [4]	0.0379% [6.5]	0.0354% [5]	0.0268% [3]	0.0379% [6.5]
	$avgLeaves$	54.2 [7]	53.3 [6]	40 [5]	19.6 [2.5]	17.7 [1]	27.7 [4]	19.6 [2.5]
Monks-3 Train	CCI	60.6557% [6]	59.836% [7]	81.1475% [4]	90.1639% [2.5]	91.8032% [1]	78.6885% [5]	90.1639% [2.5]
	K	0.2107 [6]	0.1941 [7]	0.6223 [4]	0.8031 [2.5]	0.8359 [1]	0.5734 [5]	0.8031 [2.5]
	MAE	0.3934 [6]	0.4016 [7]	0.1885 [4]	0.0983 [2.5]	0.0819 [1]	0.2131 [5]	0.0983 [2.5]
	$avgI$	0.1424% [2]	0.1416% [1]	0.2145% [4]	0.3146% [6.5]	0.3142% [5]	0.2102% [3]	0.3146% [6.5]
	$avgNMI1$	0.0341% [2]	0.0296% [1]	0.078% [3]	0.1104% [5.5]	0.112% [7]	0.1072% [4]	0.1104% [5.5]
	$avgLeaves$	61.4 [6.5]	61.4 [6.5]	36.3 [5]	17.1 [2.5]	17 [1]	34.6 [4]	17.1 [2.5]
Postoperative	CCI	57.4712% [3]	58.6206% [2]	59.7701% [1]	55.1724% [4]	50.5747% [7]	52.8735% [5.5]	52.8735% [5.5]
	K	−0.0882 [4]	−0.0726 [3]	−0.1153 [7]	−0.0353 [1]	−0.0957 [5]	−0.0565 [2]	−0.1143 [6]
	MAE	0.2835 [3]	0.2758 [2]	0.2681 [1]	0.2988 [4]	0.3295 [7]	0.3141 [5.5]	0.3141 [5.5]
	$avgI$	0.1176% [1]	0.1191% [2]	0.1573% [4]	0.1831% [6]	0.2092% [7]	0.1517% [3]	0.1815% [5]
	$avgNMI1$	0.0472% [6]	0.0416% [4]	0.0273% [1]	0.0365% [2.5]	0.0551% [7]	0.0428% [5]	0.0365% [2.5]
	$avgLeaves$	46.9 [7]	46.6 [6]	41.5 [5]	35.5 [2]	34.5 [1]	39.3 [4]	36.2 [3]
SWD	CCI	58.8% [2]	58.3% [6]	58.5% [3.5]	58.4% [5]	58.5% [3.5]	59.5% [1]	58.2% [7]
	K	0.3698 [2]	0.3624 [6]	0.3656 [3]	0.3633 [5]	0.3637 [4]	0.3804 [1]	0.36 [7]
	MAE	0.206 [2]	0.2085 [6]	0.2075 [3.5]	0.208 [5]	0.2075 [3.5]	0.2025 [1]	0.209 [7]
	$avgI$	0.013% [3]	0.0114% [2]	0.0195% [6]	0.0215% [7]	0.0179% [4]	0.0114% [1]	0.0193% [5]
	$avgNMI1$	0.0006% [2]	0.0007% [3.5]	0.0007% [5.5]	0.0008% [7]	0.0007% [3.5]	0.0004% [1]	0.0007% [5.5]
	$avgLeaves$	114.8 [7]	114.2 [6]	113.6 [5]	112.3 [2]	112.4 [3]	111.9 [1]	112.9 [4]

Table 13: Results concerning CCI, K, MAE, $avgI$, $avgNMI1$ and $avgLeaves$ of tests on last seven real datasets (performance ranks are reported in square brackets, averaging the ties)

behave better than the others for what concern monotonicity.

Quite large differences are also observed for the average performance ranks of $avgLeaves$, where our measures H_S^*, H_G^* and H_P^* occupy the last three positions of the related ranking. This comes from the fact that our discrimination measures generally produce trees with a larger number of leaves, as already observed in tests on artificial datasets.

CCI		K		MAE	
H_G	[3.4333]	H_G	[3.0666]	H_G	[3.4333]
H_S	[3.4333]	H_S	[3.4000]	H_S	[3.4333]
H_P^*	[3.8666]	H_{MID}^{10}	[3.8666]	H_P^*	[3.9333]
H_{ICT}	[4.0666]	H_{ICT}	[4.2000]	H_{ICT}	[4.0666]
H_G^*	[4.1666]	H_G^*	[4.2333]	H_G^*	[4.1333]
H_{MID}^{10}	[4.3333]	H_P^*	[4.4000]	H_{MID}^{10}	[4.3333]
H_S^*	[4.7000]	H_S^*	[4.8333]	H_S^*	[4.6666]
$avgI$		$avgNMI1$		$avgLeaves$	
H_S^*	[1.6666]	H_S^*	[2.3666]	H_S	[1.9666]
H_G^*	[1.8000]	H_G^*	[2.7000]	H_G	[2.1333]
H_{MID}^{10}	[3.0000]	H_P^*	[3.0333]	H_{ICT}	[2.7666]
H_P^*	[4.0666]	H_{MID}^{10}	[3.5666]	H_{MID}^{10}	[4.0666]
H_S	[5.4000]	H_{ICT}	[5.1333]	H_P^*	[4.7333]
H_{ICT}	[5.9000]	H_S	[5.5333]	H_G^*	[6.1000]
H_G	[6.1666]	H_G	[5.6666]	H_S^*	[6.2333]

Table 14: Rankings for tests on real datasets according to average performance ranks (reported in square brackets)

In order to give a statistical relevance of our previous observations, we followed the procedure described in [11] selecting the *Friedman test* and the *Nemenyi post-hoc test* for pairwise comparisons of the average performance ranks of each discrimination measure, with respect to each performance index.

For each performance index, the null-hypothesis of Friedman test is that all the tested discrimination measures are equivalent in performance. Table 15 shows the values of the Friedman statistic χ_F^2 and the Iman and Davenport correction F_F, for each performance index.

Since the tests are related to 15 datasets and 7 algorithms (each corresponding to a discrimination measure used for splitting), the F_F statistic is distributed according to the F-distribution with 6 and 84 degrees of freedom, whose critical value for $\alpha = 0.05$ is 2.2085.

For CCI, K and MAE since the F_F statistic is less than 2.2085 we are not able to reject the null hypothesis, i.e., we cannot say that there are differences in performance among the tested discrimination measures.

For $avgI$, $avgNMI1$ and $avgLeaves$, instead, the F_F statistic is greater than 2.2085 thus we reject the null hypothesis, i.e., we can say that there are differences in performance among the tested measures.

For these performance indices we can proceed with the Nemenyi post-hoc

	Friedman statistic χ_F^2	Iman and Davenport correction F_F
CCI	4.1571	0.6779
K	7.0642	1.1924
MAE	3.9357	0.6402
$avgI$	69.2786	46.8066
$avgNMI1$	38.2286	10.3377
$avgLeaves$	61.3286	29.9462

Table 15: Friedman statistic χ_F^2 and Iman and Davenport correction F_F for average performance rank comparisons on real datasets

test, for which the critical difference is $CD = 2.3262$. Tables 16, 17 and 18 show the pairwise differences (in absolute value) of average performance ranks, for $avgI$, $avgNMI1$ and $avgLeaves$.

	H_G^*	H_S^*	H_P^*	H_G	H_S	H_{MID}^{10}	H_{ICT}
H_G^*	—	—	—	—	—	—	—
H_S^*	0.1333	—	—	—	—	—	—
H_P^*	2.2666	2.4000	—	—	—	—	—
H_G	4.3666	4.5000	2.1000	—	—	—	—
H_S	3.6000	3.7333	1.3333	0.7666	—	—	—
H_{MID}^{10}	1.2000	1.3333	1.0666	3.1666	2.4000	—	—
H_{ICT}	4.1000	4.2333	1.8333	0.2666	0.5000	2.9000	—

Table 16: Pairwise differences of average performance ranks for $avgI$

	H_G^*	H_S^*	H_P^*	H_G	H_S	H_{MID}^{10}	H_{ICT}
H_G^*	—	—	—	—	—	—	—
H_S^*	0.3333	—	—	—	—	—	—
H_P^*	0.3333	0.6666	—	—	—	—	—
H_G	2.9666	3.3000	2.6333	—	—	—	—
H_S	2.8333	3.1666	2.5000	0.1333	—	—	—
H_{MID}^{10}	0.8666	1.2000	0.5333	2.1000	1.9666	—	—
H_{ICT}	2.4333	2.7666	2.1000	0.5333	0.4000	1.5666	—

Table 17: Pairwise differences of average performance ranks for $avgNMI1$

The underlined values listed in Tables 16, 17 and 18, are the pairwise differences greater than the critical difference CD, i.e., those implying a statistically significant difference in performance.

For what concerns $avgI$ the Nemenyi test highlights that:

- H_G^* performs better than H_G, H_S and H_{ICT};
- H_S^* performs better than H_P^*, H_G, H_S and H_{ICT};

	H^*_G	H^*_S	H^*_P	H_G	H_S	H^{10}_{MID}	H_{ICT}
H^*_G	—	—	—	—	—	—	—
H^*_S	0.1333	—	—	—	—	—	—
H^*_P	1.3666	1.5000	—	—	—	—	—
H_G	3.9666	4.1000	2.6000	—	—	—	—
H_S	4.1333	4.2666	2.7666	0.1666	—	—	—
H^{10}_{MID}	2.0333	2.1666	0.6666	1.9333	2.1000	—	—
H_{ICT}	3.3333	3.4666	1.9666	0.6333	0,8000	1.3000	—

Table 18: Pairwise differences of average performance ranks for $avgLeaves$

- H^{10}_{MID} performs better than H_G, H_S and H_{ICT};

thus measures H^*_G, H^*_S and H^{10}_{MID} form a group having better $avgI$ performance, even if we are not able to detect a statistical significant difference among them.

For what concerns $avgNMI1$, instead, the Nemenyi test highlights that:

- H^*_G performs better than H_G, H_S and H_{ICT};
- H^*_S performs better than H_G, H_S and H_{ICT};
- H^*_P performs better than H_G and H_S;

thus measures H^*_G, H^*_S and H^*_P form a group having better $avgNMI1$ performance, even if we are not able to detect a statistical significant difference among them.

Finally, for what concerns $avgLeaves$ the Nemenyi test highlights that:

- H_G performs better than H^*_G, H^*_S and H^*_P;
- H_S performs better than H^*_G, H^*_S and H^*_P;
- H_{ICT} performs better than H^*_G and H^*_S;

thus measures H_G, H_S and H_{ICT} form a group having better $avgLeaves$ performance, even if we are not able to detect a statistical significant difference among them.

The executed experimental analysis on real datasets does not allow to deduce a statistical significant difference among the tested measures in terms of indices of accuracy CCI, K and MAE. On the other hand, measures H^*_S and H^*_G have significantly better results than H_G, H_S and H_{ICT} in terms of $avgI$ and $avgNMI1$, while H^*_P has significantly better results than H_G and H_S in terms $avgNMI1$.

We notice that also H^{10}_{MID} has significantly better results than H_G, H_S and H_{ICT} in terms of $avgI$, but the Nemenyi test does not allow to conclude that it is worse or better than our measures H^*_S, H^*_G and H^*_P, even if H^{10}_{MID} follows H^*_S and H^*_G in the $avgI$ ranking.

As already pointed out, the better performance of our measures in terms of $avgI$ and $avgNMI1$ comes with a higher average number of leaves of the

constructed trees which, in turn, implies a finer partition of the description space.

Measures H_S^*, H_G^* and H_P^* perform better in terms of monotonicity because they are able to produce a finer partition of the description space and at the same time they tend to minimize the non-monotone labels. This last fact has a particular impact on the index $avgI$ whose value strictly depends on the number of leaves.

From a statistical point of view, the lager size of trees built with our measures could be a cause of overfitting that must be properly treated. Actually, such trees are not thought to be directly used in classification tasks but they should serve as input of suitable post-processing algorithms.

Indeed, several algorithms for building globally monotone decision trees proposed in the literature, essentially rely on a post-processing of an input tree, usually grown with standard methods [6, 31]. For example, the ICT algorithm [34] executes a recursive relabelling of the leaves with possible local pruning of equal leaves, while the pruning-based algorithm described in [13] starts from a large overfitted tree and applies repeatedly a suitable post-pruning rule: in both cases the process stops once a globally monotone decision tree is reached.

Algorithms like those described above can surely benefit from an input tree built with our measures, as the tree is from the beginning better in terms of monotonicity than trees built with other measures. This is particularly relevant in the ICT algorithm which alters the information extracted from the dataset with the only purpose of achieving global monotonicity. Indeed, starting from a better input tree in terms of monotonicity reduces the quantity of exogenous information introduced by ICT during the post-processing.

In the case the globally monotone decision tree resulting from the chosen post-processing is still too large (this is the case, for example, when the input tree grown with our measures is globally monotone from the beginning and thus the post-processing for enforcing global monotonicity does not take place) then the post-pruning rule for globally monotone decision trees given in [5] can be applied.

11. Conclusions and future works

In this paper, following the approach given in [18], we proposed a rank generalization of Gini discrimination measure and Yuan and Shaw discrimination measure, moreover we directly introduced a third function inspired to the functional structure of the second generalized measure.

Since the introduced measures share a common functional structure, a hierarchical construction model for rank discrimination measures has been developed and the properties a function must satisfy to be a *rank discrimination measure* (according to our conception) have been isolated. This hierarchical construction model can serve also as a basis for creating new rank discrimination measures.

Computational aspects concerning splitting criteria based on rank discrimination measures have also been investigated.

We presented a binary tree classifier RDMT(H) parametrized by a discrimination measure H used for splitting and other three pre-pruning parameters. Even if we cannot expect from it a globally monotone classifier in general, RDMT(H) guarantees a weak form of monotonicity on the resulting tree, namely *rule monotonicity*, in the case the dataset is monotone consistent and H is a rank discrimination measure.

The present paper mainly focused on the effect of different discrimination measures on the properties of the final trees with the goal of empirically showing the effectiveness of rank discrimination measures.

RDMT(H) has been used to compare measures H^*_G, H^*_S, H^*_P, H_G, H_S, H^{10}_{MID} and H_{ICT}, in terms of classification accuracy, monotonicity and size of the constructed trees.

We executed tests on artificial datasets having an increasing degree of non-monotone noise [28]. From these tests no significant difference in performance concerning classification accuracy has emerged, while H^*_G, H^*_S and H^*_P could be shown to produce trees with a higher degree of monotonicity especially for increasing non-monotone noise, despite their larger number of leaves.

Tests on real datasets have been also executed, selecting 15 datasets from UCI [33] and WEKA [37] repositories. The results of tests on real datasets have been statistically validated through the Friedman test and the post-hoc Nemenyi test [11]. Also in these tests the compared measures resulted not to be significantly different in accuracy, while H^*_G, H^*_S and H^*_P (especially H^*_G and H^*_S) showed to behave better with respect to monotonicity, despite, again, a larger number of leaves in the built trees.

The trees grown with our measures are thought to serve as input to post-processing algorithms for enforcing global monotonicity like ICT [34] or the pruning-based algorithm described in [13]. In particular, algorithms like ICT or any other post-processing algorithm altering the information extracted from the dataset, could surely benefit from an input tree built using our measures. Indeed, since such a tree has "naturally" a higher degree of monotonicity and is solely based on the dataset information, the exogenous information introduced during the post-processing of the tree seems to be surely less than starting from a tree built using other measures.

The comparative analysis presented in this paper essentially focused on the discrimination measures adopted by most known monotone decision tree induction algorithms. Of course, the chosen measures do not exhaust the plethora of alternatives and we plan to enlarge the family of splitting criteria implemented in RDMT for future comparisons. Among the possibilities we quote the modification of Gini index given in [38]. Another promising alternative is represented by deriving a discrimination measure based on stochastic monotonicity [32].

Other aspects of splitting with rank discrimination measures deserve future investigations. First of all, in this paper no assumption has been made on the class distribution, so it is important to study the behaviour of different measures in case of unbalanced classes and also their response to sampling on the initial dataset (see [10] for an analogous issue on standard decision trees).

Secondly, it is known [6] that standard discrimination measures like Gini in-

dex or the conditional Shannon entropy suffer from the so-called selection bias, i.e., they tend to select attributes with more values. Of course such circumstance can affect also monotone classification problems, as they are indistinguishable to standard discrimination measures from ordinary classification problems. Nevertheless, discrimination measures sensitive to monotonicity are intentionally biased in a way to select at each split the attribute "enforcing the most" a form of monotonicity: the distinguishing feature of each measure is exactly the way it realizes its bias toward monotonicity. For this we believe selection bias needs a proper definition in case of monotone classification problems since here we do not cope with only classification accuracy but also monotonicity enters in the play.

As a further quite natural future development we mention the fuzzification of the proposed rank discrimination measures in a way to deal with fuzzy decision tree classifiers [20, 39].

Acknowledgements

Both the authors are grateful to the anonymous referees for valuable suggestions, especially concerning the computational and experimental analysis. The results presented in this paper have been achieved during second author's stay at the Laboratoire d'Informatique de Paris 6 (LIP6), Université Pierre et Marie Curie, Paris. The second author would like to thank all the members of the department of Données et Apprentissage Artificiel for their kind willingness and their precious scientific support.

References

[1] J. Aczél and Z. Daróczy. *On measures of information and their characterizations*, volume 115 of *Mathematics in science and engineering*. Academic Press New York / San Francisco / London, 1975.

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

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

[4] A. Ben-David, L. Sterling, and T. Tran. Adding monotonicity to learning algorithms may impair their accuracy. *Expert Systems with Applications*, 36(3, Part 2):6627–6634, 2009.

[5] J.C. Bioch and V. Popova. Monotone decision trees and noisy data. Technical Report ERS-2002-53-LIS, ERIM, Rotterdam School of Management, Erasmus Universiteit Rotterdam, 2002.

[6] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, Boca Raton, 1984.

[7] K. Cao-Van. *Supervised ranking, from semantics to algorithms*. PhD thesis, Universiteit Gent, 2003.

[8] K. Cao-Van and B. De Baets. Consistent representation of rankings. In H. de Swart, E. Orlowska, G. Schmidt, and M. Roubens, editors, *Theory and Applications of Relational Structures as Knowledge Instruments*, volume 2929 of *Lecture Notes in Computer Science*, pages 1966–1967. Springer Berlin / Heidelberg, 2003.

[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] D.A. Cieslak and N.V. Chawla. Learning decision trees for unbalanced data. In W. Daelemans, B. Goethals, and K. Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 5211 of *Lecture Notes in Computer Science*, pages 241–256. Springer Berlin Heidelberg, 2008.

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

[12] A. Feelders. Monotone Relabeling in Ordinal Classification. In *IEEE International Conference on Data Mining 2010 (ICDM 2010)*, pages 803–808, 2010.

[13] A. Feelders and M. Pardoel. Pruning for monotone classification trees. In M.R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, editors, *Advances in Intelligent Data Analysis V*, volume 2810 of *Lecture Notes in Computer Science*, pages 1–12. 2003.

[14] S. Greco, B. Matarazzo, and R. Slowinski. Rough approximation by dominance relations. *International Journal of Intelligent Systems*, 17(2):153–171, 2002.

[15] S. Greco, B. Matarazzo, and R. Slowinski. Rough sets methodology for sorting problems in presence of multiple attributes and criteria. *European Journal of Operational Research*, 138(2):247–259, 2002.

[16] S. Greco, B. Matarazzo, and R. Slowinski. Customer satisfaction analysis based on rough set approach. *Zeitschrift für Betriebswirtschaft*, 77:325–339, 2007.

[17] Q. Hu, X. Che, L. Zhang, D. Zhang, M. Guo, and D. Yu. Rank entropy based decision trees for monotonic classification. *Knowledge and Data Engineering, IEEE Transactions on*, 24(11):2052–2064, 2012.

[18] Q. Hu, M. Guo, D. Yu, and J. Liu. Information entropy for ordinal classification. *SCIENCE CHINA Information Sciences*, 53:1188–1200, 2010.

[19] Q. Hu, W. Pan, L. Zhang, D. Zhang, Y. Song, M. Guo, and D. Yu. Feature selection for monotonic classification. *Fuzzy Systems, IEEE Transactions on*, 20(1):69–81, 2012.

[20] E. Hüllermeier and S. Vanderlooy. Why fuzzy decision trees are good rankers. *Fuzzy Systems, IEEE Transactions on*, 17(6):1233–1244, 2009.

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

[22] S. Lievens and B. De Baets. Supervised ranking in the WEKA environment. *Information Sciences*, 180(24):4763–4771, 2010.

[23] K. Makino, T. Suda, H. Ono, and T. Ibaraki. Data analysis by positive decision trees. *IEICE Transactions on Information and Systems*, E82-D(1):76–88.

[24] C. Marsala. Gradual fuzzy decision trees to help medical diagnosis. In *IEEE International Conference on Fuzzy Systems 2012 (FUZZ-IEEE 2012)*, page (accepted), 2012.

[25] C. Marsala and B. Bouchon-Meunier. Ranking attributes to build fuzzy decision trees: a comparative study of measures. In *IEEE International Conference on Fuzzy Systems 2006 (FUZZ-IEEE 2006)*, pages 1777–1783, 2006.

[26] C. Marsala and B. Bouchon-Meunier. Quality of measures for attribute selection in fuzzy decision trees. In *IEEE International Conference on Fuzzy Systems 2010 (FUZZ-IEEE 2010)*, pages 1–8, 2010.

[27] C. Marsala and D. Petturiti. Hierarchical model for rank discrimination measures. In L.C. van der Gaag, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 7958 of *Lecture Notes in Computer Science*, pages 412–423. Springer Berlin Heidelberg, 2013.

[28] I. Milstein, A. Ben-David, and R. Potharst. Generating noisy ordinal monotone datasets. *Artificial Intelligence Research*, 3(1), 2014.

[29] R. Potharst and J.C. Bioch. Decision trees for ordinal classification. *Intelligent Data Analysis*, 4(2):97–111, 2000.

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

[31] J.R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[32] M. Rademaker and B. De Baets. Optimal restoration of stochastic monotonicity with respect to cumulative label frequency loss functions. *Information Sciences*, 181(4):747–757, 2011.

[33] UCI. UC Irvine Machine Learning Repository. `http://archive.ics.uci.edu/ml/`.

[34] R van der Kamp, A. Feelders, and N. Barile. Isotonic classification trees. In N.M. Adams, C. Robardet, A. Siebes, and J.-F. Boulicaut, editors, *Advances in Intelligent Data Analysis VIII*, volume 5772 of *Lecture Notes in Computer Science*, pages 405–416. Springer Berlin Heidelberg, 2009.

[35] M. Velikova and H. Daniels. Decision trees for monotone price models. *Computational Management Science*, 1:231–244, 2004.

[36] M. Velikova, H. Daniels, and M. Samulski. Partially Monotone Networks Applied to Breast Cancer Detection on Mammograms. In V. Kurková, R. Neruda, and J. Koutník, editors, *Artificial Neural Networks - ICANN 2008*, volume 5163 of *Lecture Notes in Computer Science*, pages 917–926. Springer Berlin / Heidelberg, 2008.

[37] WEKA. Machine Learning Group at University of Waikato. `http://www.cs.waikato.ac.nz/ml/weka/`.

[38] F. Xia, W. Zhang, and F. Li Y. Yang. Ranking with decision tree. *Knowledge and Information Systems*, 17(3):381–395, 2008.

[39] Y. Yuan and M.J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69(2):125–139, 1995.