Pattern Mining for Label Ranking

by

Cláudio Frederico Pinho Rebelo de Sá

Pattern Mining for Label Ranking

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof.mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op vrijdag 16 december 2016
klokke 11.15 uur

door

Cláudio Frederico Pinho Rebelo de Sá

geboren te Porto, Portugal
in 1984

Promotiecommissie

Promotor:	prof. dr. J. N. Kok (Universiteit Leiden)
Co-promotor:	dr. C. M. Soares (Universidade do Porto)
Co-promotor:	dr. A. J. Knobbe (Universiteit Leiden)
Overige leden:	prof. dr. T. H. W. Bäck (Universiteit Leiden)
	prof. dr. H. J. van den Herik (Universiteit Leiden)
	dr. P. Kralj Novak (Jožef Stefan Institute)
	dr. M. Atzmüller (Universität Kassel)

Front and back cover patterns by Vecree.com, available under a Creative Commons License (CC BY-SA 2.0).
Printing: Ridderprint BV, the Netherlands

Dedicado à minha mãe,
por sempre acreditar em mim

Contents

Chapter 1

Introduction

Preferences are present in many tasks in our daily lives. Buying the right car, choosing a suitable house or even deciding on the food to eat, are trivial examples of decisions that reveal information, explicitly or implicitly, about our preferences. Hence, extracting and modeling preferences can provide us with invaluable information about the choices of a group of persons or individuals. However, this problem is non-trivial because, quite often, preferences depend on different context and options available [83]. Moreover, in areas like e-commerce, which typically deal with decisions from thousands of users, the acquisition of preferences can be a difficult task [57].

For that reason, artificial intelligent methods have been increasingly important for the discovery and automatic learning of preferences [47]. In particular, a subfield of machine learning which focuses on the study and modeling of preferences is *Preference Learning*.

In this thesis, we focus on one subtask of Preference Learning (introduced in Section 1.1), the prediction and analysis of preferences given a predefined set of objects/labels, commonly referred to as *Label Ranking* (Section 1.2).

1.1 Preference Learning

Preference Learning is an emerging subfield of machine learning that focuses on the study and modeling of preferences[1]. Preference learning methods

[1]A comprehensive overview of the state-of-the-art in the field of preference learning can be found in the *Preference Learning* book [57].

are conceptually different from standard machine learning problems such as classification or regression, as it can involve the prediction of more complex structures [7]. Classification and regression problems focus on the prediction of single values, while preference learning methods are designed to predict the order, or ranking, of a set of objects by relative importance.

In this field, the term *preference* is not strictly referring to preferences of individuals, but can also represent more general order relations. In turn, this flexibility gives an important advantage to the paradigm of preference-based learning, like extracting knowledge which, otherwise, would be harder [14]. However, without loss of generality, the discussion will focus on the more traditional type of preferences for easier interpretation.

Preferences can be extracted in an *explicit* way. As an illustrative example, a person who claims to prefer *apples* to *pears*, represented as:

$$apples \succ pears$$

is giving information about an explicit preference. In [81], 5000 Japanese people were asked to order 10 types of sushi by preference.

However, sometimes, information about preference is only *implicitly* given. Going back to the fruit example, if someone picks *bananas* from a basket containing *apples*, *pears* and *bananas*, one can implicitly infer that:

$$bananas \succ apples \wedge bananas \succ pears$$

One real example can be found in [114], where preferences are implicitly taken from clicking behavior of users.

Regardless of how preferences are extracted, they can be given as *relative* or *absolute*. *Relative* preferences cannot be quantifiable (e.g. sorting fruit by taste: $bananas \succ apples \succ pears$) [57]. On the other hand, *absolute* preferences are given in a quantitative form (e.g. the cost of the fruit: *bananas* = 2\$, *pears* = 1\$, *apples* = 3\$). Despite its different nature, in preference learning all types of preferences are combined in the same learning perspective [57].

In terms of modeling the preferences, there are two main approaches, learning *utility functions* and learning *preference relations* [57]. Learning utility functions, is learning to assign a relevance score to each object, which can later be ordered by comparison. Learning preference relations, is to learn the relative order relations between the objects being studied. This type of approach can be difficult to learn in cases where there are many objects to order [42]. For example, consider the ordering of web pages by search engines [78]. In such cases, it is easier to rely on methodologies that learn utility functions.

In short, preference learning, is to learn from empirical data with implicit or explicit preferences. These preferences are explored by preference mining methods [57]. Preference learning is also about predicting preferences in new scenarios, when good generalizations from the given data are possible.

Preference learning can be divided into three main categories [57], *object ranking*, *instance ranking* and *label ranking*.

Object ranking The goal in the *object ranking* task is to output the ranking of a given set of objects, that, in theory, can be infinitely large. It can be considered a regression task whose target variables are orders [82]. A practical example are the lists of ordered web pages generated by search engines [78, 114]. In these case, utility functions are trained to assign a score to each newly given object [57].

Instance ranking In *instance ranking*, the setting is similar to ordinal classification [23], where an instance belongs to a class, among a finite set of classes with a natural order [57]. As an example, consider the assignment of conference papers to categories like: *reject*, *weak reject*, *weak accept* and *accept* [57].

Instance ranking is a generic term for bipartite [89] and multipartite [59] ranking.

In this thesis, we focus on the *label ranking* task (Section 1.2) and its applications.

1.2 Label Ranking

Label ranking is a sub-field of preference learning [57, 26, 123] which studies the problem of learning a mapping from instances to rankings over a finite number of predefined labels. It can be considered a variant of the conventional classification problem [26]. While in classification the goal is to assign examples to a specific class, in label ranking we are interested in assigning a complete preference order of the labels to every example. If this is not possible, incomplete orders can also be assigned to some examples [28].

There are two approaches to tackle label ranking data [6, 24]. *Reduction techniques* (Section 1.2.3), also known as *decomposition methods*, divide the problem into several simpler problems (e.g. ranking by pairwise comparisons [56]). *Direct methods* (Section 1.2.4) treat the rankings without any transformation (e.g. decision trees adapted for the label ranking task [120, 26] or case-based approaches for label ranking [17, 24]).

Label ranking has been used in different applications, mainly for predictive tasks. For example, in meta-learning [16], to predict a ranking of a set of algorithms according to the best expected accuracy on a given dataset. In microarray analysis [74], to find patterns in genes from Yeast on different micro-array experiments. And also in image categorization [58], to predict the relative importance of categories of elements in landscape pictures (e.g. beach, sunset, field, fall foliage, mountain and urban).

1.2.1 Definition

Given an instance x from the instance space $\mathbb{X}$, the goal is to predict the ranking of the labels $\mathcal{L} = \{\lambda_1, \dots, \lambda_k\}$ associated with x [74]. The ranking can be represented as permutation or as an ordered vector.[2] The permutation, denoted as π, contains numbers from 1 to k, where 1 indicates the first position and k the last one (e.g. $\pi = (1, 2, 3, 4)$). The ordered vector represents the objects with an operator indicating the order of the preference (e.g. $\lambda_a \succ \lambda_b \succ \lambda_c \succ \lambda_d$).

The goal in label ranking is to learn the mapping $\mathbb{X} \to \Omega$, where Ω is defined as the permutation space. However, as in classification, we do not assume the existence of a deterministic $\mathbb{X} \to \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω [26]. This means that, for each $x \in \mathbb{X}$, there exists a probability distribution $\mathcal{P}(\cdot|x)$ such that, for every ranking $\pi \in \Omega$, $\mathcal{P}(\pi|x)$ is the probability that π is the ranking associated with x. The training data contains a set of instances $D = \{\langle x_i, \pi_i \rangle\}$, $i = 1, \dots, n$, where x_i is a vector containing the values $x_i^j, j = 1, \dots, m$ of m independent variables, $\mathcal{A}$, describing instance i and π_i is the corresponding target ranking.

Rankings can be either total or partial orders.

[2]Both notations will be used interchangeably in this dissertation.

Total orders A *strict total order* over $\mathcal{L}$ is defined as:[3]

$$\{\forall\,(\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succ \lambda_b \vee \lambda_b \succ \lambda_a\}$$

which represents a *strict ranking* [123], a *complete ranking* [57], or simply a *ranking.* A strict total order can also be represented as a permutation π of the set $\{1, \ldots, k\}$, such that $\pi(a)$ is the position, or *rank*, of λ_a in π. For example, the *strict total order* $\lambda_1 \succ \lambda_2 \succ \lambda_3 \succ \lambda_4$ can be represented as $\pi = (1, 2, 3, 4)$.

However, in real-world ranking data, we do not always have clear and unambiguous preferences, i.e. strict total orders [15]. Hence, sometimes we have to deal with *indifference* ($\sim$) and *incomparability* ($\perp$) [42]. For illustration purposes, let us consider the scenario of elections. If a voter feels that two candidates have identical proposals, then her preference can be expressed as indifferent, so they are assigned the same rank (i.e. a tie). To represent ties, we need a more relaxed setting, called *non-strict total orders*, or simply *total orders*, over $\mathcal{L}$, by replacing the binary strict order relation, $\succ$, with the binary partial order relation, $\succeq$:

$$\{\forall\,(\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succeq \lambda_b \vee \lambda_b \succeq \lambda_a\}$$

These non-strict total orders can represent *partial rankings* (rankings with ties) [123]. For example, the *non-strict total order* $\lambda_1 \succ \lambda_2 \sim \lambda_3 \succ \lambda_4$ can be represented as $\pi = (1, 2, 2, 3)$.

Additionally, real-world data may lack preferences data regarding two or more labels, which is known as *incomparability.* Continuing with the elections example, if the voter is familiar with the proposals of λ_a but not those of λ_b, she is unable to compare them, $\lambda_a \perp \lambda_b$. In other words, the voter cannot decide whether the candidates are equivalent or select one as her favorite. In this case, we can use *partial orders.*

Partial orders Similar to *total orders*, there are *strict* and *non-strict partial orders.* Let us consider the *non-strict partial orders* (which can also be referred to as *partial orders*) over $\mathcal{L}$:

$$\{\forall\,(\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succeq \lambda_b \vee \lambda_b \succeq \lambda_a \ \vee \lambda_a \perp \lambda_b\}$$

We can represent partial orders with *subrankings* [70]. For example, the *partial order* $\lambda_1 \succ \lambda_2 \succ \lambda_4$ can be represented as $\pi = (1, 2, 0, 4)$, where 0 represents that λ_3 is incomparable to the others, i.e. $\lambda_1, \lambda_2, \lambda_4 \perp \lambda_3$.

[3]For convenience, we say *total order* but in fact we mean a *totally ordered set.* Strictly speaking, a *total order* is a binary relation.

1.2.2 Evaluation

Given an instance x_i with label ranking π_i and a ranking $\hat{\pi}_i$ predicted by a label ranking model, several loss functions on Ω can be used to evaluate the accuracy of the prediction. One such function is the number of discordant label pairs:

$$\mathcal{D}(\pi, \hat{\pi}) = \#\{(a, b)\,|\pi(a) > \pi(b) \wedge \hat{\pi}(a) < \hat{\pi}(b)\}$$

If there are no discordant label pairs, the distance $\mathcal{D} = 0$. On the other hand, the function to define the number of concordant pairs is:

$$\mathcal{C}(\pi, \hat{\pi}) = \#\{(a, b)\,|\pi(a) > \pi(b) \wedge \hat{\pi}(a) > \hat{\pi}(b)\}$$

These concepts are used in the definition of several metrics that can be used for evaluation in label ranking:

Kendall Tau Kendall's τ coefficient [85] is the normalized difference between the number of concordant, $\mathcal{C}$, and discordant pairs, $\mathcal{D}$:

$$\tau(\pi, \hat{\pi}) = \frac{\mathcal{C} - \mathcal{D}}{\frac{1}{2}k(k-1)}$$

where $\frac{1}{2}k(k-1)$ is the number of possible pairwise combinations, $\binom{k}{2}$. The values of this coefficient range from $[-1, 1]$, where $\tau(\pi, \pi) = 1$ (i.e. when the rankings are equal) and $\tau(\pi, \pi^{-1}) = -1$ if π^{-1} denotes the inverse order of π (e.g. $\pi = (1, 2, 3, 4)$ and $\pi^{-1} = (4, 3, 2, 1)$). Kendall's τ can also be computed in the presence of ties, using τ_B [5].

Gamma coefficient If we want to measure the correlation between two partial orders (subrankings), or between total and partial orders, we can use the Gamma coefficient [93]:

$$\gamma(\pi, \hat{\pi}) = \frac{\mathcal{C} - \mathcal{D}}{\mathcal{C} + \mathcal{D}}$$

Note that the Gamma coefficient is identical to Kendall's τ coefficient in the presence of strict total orders, because, in this case, $\mathcal{C} + \mathcal{D} = \frac{1}{2}k(k-1)$.

Spearman distance One other commonly used measure is the Spearman's rank correlation coefficient [118]. It is defined as:

$$\rho\left(\pi, \hat{\pi}\right) = 1 - \frac{6 d_S\left(\pi, \hat{\pi}\right)}{k\left(k^2 - 1\right)}$$

where d_S is the squared sum of rank differences, also referred as *Spearman distance* [82]:

$$d_S\left(\pi, \hat{\pi}\right) = \sum_{a=1}^{k}\left(\pi\left(a\right) - \hat{\pi}\left(a\right)\right)^2$$

In other words, the Spearman's rank correlation coefficient is the normalized version of the *Spearman distance* into the interval $[-1, 1]$.

Weighted rank correlation measures Sometimes it is more important to predict the items in the top ranks than the ones ranked lower. For instance, when predicting the ranking of financial analysts to choose which ones to follow [6], it is more important to predict the best ones correctly than the worst ones. That is because it would not be very wise to follow the recommendations of the worst analysts. Thus, labels could be associated with cost and benefit values, which determine the real value of the ranking. For instance, to follow a given analyst, I have to buy the stocks he recommends. On the other hand, following different analysts will likely yield different gains or losses in the market. The empirical evaluation of ranking methods will only be useful in practice if these issues are taken into account.

In these cases, a weighted rank correlation coefficient can be used. They are typically adaptations of existing similarity measures, such as a weighted version of the Spearman's rank coefficient [110].

In terms of evaluation techniques, the usual resampling strategies, such as holdout or cross-validation, can be used to estimate the accuracy of a label ranking algorithm [26]. The accuracy of a label ranker can be estimated by averaging the values of any of the measures explained here, over the rankings predicted for a set of test examples.

To assess the significance of differences between models, using paired tests directly is not advised, since straightforward paired tests on multiple methods might reject the null hypothesis due to random chance [43]. For this reason, two-step statistical tests are usually performed [17, 26]. The first step, consists of a Friedman test, where the null hypothesis is that all learners have equal performance. If this hypothesis is rejected, a two-tailed sign test to compare learners such as the Dunn's Multiple Comparison Procedure [104] is performed.

1.2.3 Reduction techniques

Because label ranking is a relatively new field in machine learning, some methods were basically approaching a reduction to a classification or regression problem [24], i.e. *Reduction techniques*. One great advantage of the reduction is that it makes a label ranking problem viable to be transformed into classification [74] or regression [41] problems. Also, reduction techniques can be quite efficiently implemented and easily applied for distributed systems [124]. On the other hand, there are also some disadvantages.

One option is to reduce the problem to the prediction of the *best label* (multi-label classification). This, however, will come with loss of information [23]. Assume we have the ranking of 3 algorithms in two scenarios: $Alg_1 \succ Alg_2 \succ Alg_3$ and $Alg_2 \succ Alg_1 \succ Alg_3$. A classifier, by focusing on the best one, will struggle to predict the most accurate, while a ranker will conclude that algorithms 1 and 2 perform better than 3.

One most commonly accepted reduction technique is to decompose rankings into binary preference relations, referred to as *pairwise comparisons* [74]. In simple words, it consists into reducing the problem of ranking into several classification problems. Examples of that are: Ranking by Pairwise Comparison (RPC) [74], Likelihood Pairwise Comparisons (LPC) [44] and Rule-based Label Ranking [64]. However, it has been noted that minimizing the classification error on several binary problems is not always equivalent to minimizing a loss function on rankings [23].

Ranking by Pairwise Comparisons

The method Ranking by Pairwise Comparisons (RPC) [74] is a well known reduction technique in the label ranking field. In simple terms, RPC can be divided in two phases, prediction of pairwise preferences and derivation of the rankings [74].

Before the first step, one needs to decomposed rankings into pairwise comparisons for each pair of labels of the form:

$$(\lambda_a, \lambda_b) \in \mathcal{L}, 1 \leq a < b \leq k$$

Considering that $\mathcal{L} = \{\lambda_1, \dots, \lambda_k\}$, there will be $\frac{k(k-1)}{2}$ different pairwise comparisons.

The first step is to learn a classification model from the training data for each pair of labels. This is, considering each pairwise comparison as a class, a separate model, $\mathcal{M}_{ab}$, is called to learn a mapping of the form:

$$x_i \rightarrow \left\{ \begin{array}{ll} 1 & \text{if } \lambda_a \succ \lambda_b \\ 0 & \text{if } \lambda_b \succ \lambda_a \end{array} \right\}, x_i \in D$$

This mapping can be done by any classifier at hand [74].

This approach has the advantage that it can be used with partial rankings. For any instance x_i, where nothing is known about the preference relation of a pair of labels $(\lambda_a, \lambda_b) \in \mathcal{L}$, the model $\mathcal{M}_{ab}$ ignores x_i in the training.

As a matter of choice, this can be easily adapted to deal with the interval $[0, 1]$. This will result in a *valued preference relation*, vpr_x, for every instance $x \in \mathbb{X}$:

$$vpr_x\left(\lambda_a, \lambda_b\right) \left\{ \begin{array}{cl} \mathcal{M}_{ab} & \text{if } a < b \\ 1 - \mathcal{M}_{ab} & \text{if } a > b \end{array} \right.$$

Finally, there is the aggregation step, where the predictions are combined to derive the rankings. Given the predicted pairwise comparisons for each x, the simplest approach is to order the labels, considering the predictions of the model $\mathcal{M}_{ab}$ as weights. Each label λ_a is ranked depending on the sum of the weights:

$$\sum_{\lambda_a \neq \lambda_b} vpr_x\left(\lambda_a, \lambda_b\right)$$

This task may not be trivial as there are possibilities of ties. In this regard, there are some well studied and documented approaches [55, 74]. However, one simple approach is to favor the most common classes according to the class distribution [74].

1.2.4 Direct approaches

Direct methods treat the rankings without any transformation. Hence, avoiding some of the problems of the reduction approaches [23], mentioned in Section 1.2.3. In this section, we outline some direct approaches for label ranking problems which have been proposed in recent years.

The most prominent approaches in the label ranking field are based on probabilistic distribution of rankings, like Mallow's Model [26] or Plackett-Luce [24]. These probabilistic methods estimate the conditional probability $\mathcal{P}(\pi|x)$ from the training data. This gives methods the advantage that, besides predicting a ranking, also provide a reliability score [24].

Case-based methods are also highly competitive direct approaches in label ranking (e.g. k-Nearest Neighbor [17, 26]). In [17] a nearest neighbor approach was proposed to deal with the problem of meta-learning. From a different perspective, in [24], the authors combined case-based with probabilistic models using the Instance-Based Label Ranking method.

A different group of label ranking methods tackle the ranking similarities with distance-based approaches (e.g., [120, 36, 116]). A relatively recent example is a neural networks adaptation proposed with Multilayer Perceptron for Label Ranking [116]. Also, in the naive Bayes for Label Ranking method [6], the prior probabilities of the rankings are similarity-based. In this cases, ranking correlation measures, like Kendall's τ coefficient [85] or the *Spearman distance* [82], are used to calculate the distance between rankings. These so-called distance-based models, make the prediction problem more similar to a regression task, where the difference between two rankings is similar to the error in a regression setting.

Tree-based models are popular in label ranking [120, 115, 26]. Decision trees are known to be competitive methods which are relatively easy to interpret [26]. In [120], Predictive Clustering Trees, successfully combine hierarchical clustering with decision trees for predicting rankings. Probabilistic models are combined in the tree generation to derive the nodes in Label Ranking Trees [26].

1.3 Contributions of this thesis

In this section, we give an overview of the contributions of this thesis, and its motivations. As mentioned in Section 1.2, there are two main approaches to the problem of label ranking [6, 24]. *Decomposition approaches* which divide the problem into several simpler problems and *Direct methods* that treat the rankings as target objects without any transformation. We focus more on direct methods but we also propose decomposition approaches.

The first part of this PhD project extends the work started with the MSc thesis [33] of the candidate. In the latter, Label Ranking Association Rules

(LRAR) were proposed [36]. LRARs are based on traditional Association Rules, redefining the support and confidence measures, in order to take into account the nature of label rankings. However, in the MSc project the empirical study was limited and little information about the behavior of LRARs was obtained. In the PhD project, this work was consolidated, namely to better understand how the rules perform in extreme conditions and in which cases are correctly applied (Section 1.3.1).

In this project we also addressed the lack of pre-processing methods that are specific to label ranking problems. LRARs, like Association Rules, cannot handle numeric data directly, which needs to be discretized beforehand. We proposed two discretization approaches that are specific for label ranking problems (Section 1.3.2). Both approaches are based on a new measure of *ranking entropy* which was developed as part of this work.

The new measure of *ranking entropy* was also the basis for a third contribution. We proposed Entropy Ranking Trees (Section 1.3.3), which is an adaptation to the problem of label ranking of a Top-Down Induction of Decision Trees algorithm. Based on this new algorithm, we made a fourth contribution, which is an ensemble method for label ranking. The algorithm is Label Ranking Forests (Section 1.3.3), which, as the name indicates, is an adaptation of Random Forests for label ranking.

There is not much work on descriptive pattern mining of label rankings and preference data. We address this shortcoming with two additional contributions, *Pairwise Association Rules* and *Exceptional Preferences Mining* (Section 1.3.4), which are two rule-based methods.

Most empirical studies on label rankings are based on a set of benchmark datasets, in the KEBI Data Repository [26]. These were generated from other datasets which were not original label ranking problems. Given the process of transformation used, it is unclear whether these datasets are useful to assess the quality of label ranking methods. Thus, the final contribution of this thesis are two swap randomization techniques for the label ranking task (Section 1.3.5). The proposed methods were used to investigate the usefulness of the available label ranking datasets.

1.3.1 Label Ranking Association Rules

Association Rules mining is used to discover interesting relationships between attributes in large databases [2]. An association rule has the form $A \to B$, meaning that when the set of values A is observed in the data, there is a chance of observing B.

Although association rules were originally developed for descriptive tasks, their success has quickly lead to their adaptation for prediction problems. The motivation for adapting Association Rules (AR) for classification is that, a classification rule model built from such an unrestrained set of rules, can potentially be more accurate than the ones using a greedy search approach [97].

Label Ranking Association Rules [33] were proposed as a predictive approach for label ranking [36]. The main adaptations to the original algorithm were on the *support* and *confidence* measures, which were modified to take into account the similarity between rankings.

The method proposed originally to mine LRAR has a parameter. Such parameter, works as a threshold that determines what should and should not be considered a sufficiently similar pair of rankings, in order to be covered by the same rule. However, the impact of that parameter in the results was not investigated originally. In Chapter 2, we consolidate the original work by discussing results of the analysis on the values of this parameter. The type of questions we investigate is, whether there is a rule of thumb to select its value or it is data-specific.

1.3.2 Discretization

As in any machine learning task, data preparation is essential for the development of accurate label ranking models. For instance, some algorithms are unable to deal with numeric variables, such as the basic versions of Naive Bayes and Association Rules [102, 4], in which case numeric variables should be discretized beforehand.

While there has been a significant development of learning algorithms for label ranking in recent years, there are not many pre-processing methods specifically for this task. Following the adaptation of Association Rules for Label Ranking, the development of a suitable discretization method was paramount. Without such a method, it would not be possible to adequately analyze data with numerical variables.

Discretization, from a general point of view, is the process of partitioning a given interval into a set of discrete sub-intervals. It is normally used to split continuous intervals into two or more sub-intervals which can then be treated as nominal values. When we transform continuous intervals into discrete sub-intervals, regardless of the splits taken, generally leads to a loss of information [60]. In theory, a good discretization should have a good balance between the loss of information and the number of partitions [90].

Discretization methods are typically organized into two groups, *supervised* and *unsupervised*, depending on whether or not they involve the target variable, respectively. In prediction problems, supervised methods usually produce more useful discretizations than unsupervised methods [46].

The difference in nature between the target variable in classification and label ranking problems implies that supervised discretization methods developed for classification are not suitable for LR. For this reason, two methods, based on a well-known supervised discretization approach for classification, were proposed as part of this PhD research. The original method, *Minimum Description Length Partition* (MDLP) [54], uses a measure of entropy from information theory, known as *Shannon entropy* [54].

The first proposed approach, *Minimum Description Length Partition for Ranking* (MDLP-R) [40] (Chapter 3), uses a ranking entropy measure based on the similarities between rankings. This ranking entropy is the equivalent of the Shannon entropy for label ranking problems. A simpler and improved measure of entropy was latter proposed and implemented in a new method, EDiRa (Entropy-based Discretization for Ranking) [39] (Chapter 3).

1.3.3 Tree-based models

Tree-based models are popular for a number of reasons, including how they can clearly express information about the problem, because their structure is relatively easy to interpret even for people without a background in learning algorithms. They have been used in classification [111], regression [20] and also label ranking [120, 26] tasks.

On the other hand, ensemble methods, which use multiple learning algorithms, usually compensate some loss in interpretability with significant accuracy improvements [19]. One of the most popular approaches are ensembles of trees, such as Random Forests [19].

Our contributions concerning the development of tree-based models for label ranking are a new variant of decision trees and the adaptation of the random forests algorithm for this task.

Entropy Ranking Trees Decision trees, like ID3 [111], grow in a top-down recursive partitioning scheme that iteratively splits data into smaller subsets [102]. This splits are performed such that each node divides the data into increasingly more homogeneous subsets, in terms of the target variable. The search for the best split point tries to optimize a given splitting criterion, such as the *information gain* [102]. Information gain measures the difference in entropy between the previous and current state relatively to a target variable.

By implementing the previously proposed *ranking entropy* measure (Section 3) in the splitting process, we proposed a novel ranking tree approach, Entropy Ranking Trees [35] (Chapter 4). The goal is to obtain leaf nodes that contain examples with target rankings as homogeneous as possible.

Label Ranking Forests Adapting Random Forests to label ranking comes in a natural way based on any decision trees approach for label ranking. Motivated by the success of Random Forests in terms of improved accuracy for classification and regression problems [13], we proposed a Random Forest approach for label ranking, Label Ranking Forests [32] (Chapter 4).

1.3.4 Descriptive mining for label ranking

Preference learning approaches can benefit from the analysis of descriptive methods [57]. In label ranking, only recently, a few descriptive approaches for mining label ranking data have been proposed [70, 122]. In [70], the authors suggest an approach using association rules that search for patterns exclusively in rankings (i.e. the independent variables are ignored). In [122], a *ranked tiling* approach to search for patterns in the ranking scores, i.e. ranks, is suggested.

The available label ranking mining approaches focus exclusively on the target ranking, and do not relate its values to the values of the independent variables. However, we believe that much valuable information can be extracted by taking both into account. For example, consider we discover that in 80% of the cases sushi A is preferred to sushi B. By taking independent variables into account, we might actually find that females prefer sushi B to sushi A, but males, which represent 80% of the population, prefer sushi A to sushi B. For that reason, we propose two approaches for mining label ranking data.

Exceptional Preferences Mining In Chapter 5, we propose an approach for finding deviating patterns in label rankings, in the context of Subgroup Discovery [88], referred to as Exceptional Preferences Mining. The aim of Subgroup Discovery is to discover subgroups for which the target shows an unusual distribution, as compared to the overall population in the data [88].

In the context of label ranking, we need to determine to what extent the subgroups show different preferences, and whether any of these preferences are in conflict with the average behavior. To that end, we developed three quality measures, *Pairwise*, *Labelwise* and *Norm*. Each of them strives to find subgroups where the preference relations are exceptional from slightly different perspectives.

The *Pairwise* measure identifies subgroups with strong deviating preferences between pairs of labels. The *Labelwise* measure identifies subgroups where at least one particular label is exceptionally under- or over-appreciated. Finally, the *Norm* quality measure will give more relevance to subgroups where several, or all, labels deviate strongly.

Pairwise Association Rules Association rules use a set of descriptors to represent meaningful subsets of the data [69], hence providing an easy interpretation of the patterns mined. We propose an approach that decomposes rankings into pairwise comparisons and then looks for meaningful associations rules of the form:

$$A \rightarrow \{\lambda_a \succeq \lambda_b \vee \lambda_a \perp \lambda_b \ \vee \lambda_a = \lambda_b | \lambda_a, \lambda_b \in \mathcal{L}\}$$

which we refer as Pairwise Association Rules (Chapter 2). [4]

1.3.5 Label Ranking Data

Due to the lack of benchmark LR datasets, 16 semi-synthetic datasets were adapted from multi-class and regression datasets from the UCI repository and Statlog project [26]. For each multi-class problem, an LR dataset (referred to as *type A* problem) was created by training a Naive Bayes and the target was replaced with a ranking based on the probability score of each

[4]For similar reasons, Label Ranking Association Rules can also be used for mining label ranking data. However, the fact that these search exclusively for complete ranking patterns, can be seen as a limitation.

class. Additionally, for each regression problem, the ranking target was created based on the values of a set of selected numerical attributes (*type B* problems).

This set of 16 datasets has been used by the majority and the contributions in the Label Ranking field [28, 27, 116, 64]. However, it is unclear if the type B datasets contain any meaningful relations between the target rankings and independent variables. Additionally, the rankings in type A problems represent the preferences of an agent, which in this case is the naive Bayes classifier. Therefore, the bias in these algorithms seems too strongly defined and, thus, their ability to represent real world distributions of data is questionable.

In many data mining applications, *swap randomizations* techniques are used together with statistical tests to validate the significance of findings [62]. Using a similar concept, we can investigate the usefulness of type B datasets. For this purpose, we propose two *swap randomization* methods specific for the label ranking datasets, *ranking permutations* and *labelwise permutations.*

Ranking permutations Randomly permuting the rankings is a natural adaptation of the methods used in classification [63]. By doing so, we want to test the *strength* of the relation between independent variables and targets in the data. After the permutation, because we break this relation, we can measure how the label ranking learners behave and compare with the results on the original data. If the differences are not significant, we can conclude that there is no real relation between independent variables and targets.

Labelwise permutations In [19], each attribute was permuted at a time to measure the impact of variables for prediction, in terms of misclassification rate. We propose a similar method by applying the same concept to each individual label (Chapter 6). We define *labelwise permutation* as the process of permuting the ranks of a specific label. This enables us to test if the amount of information in the independent variables about the rank of the selected label is significant. By comparison with the original data (without permutations), statistical significance tests can be used to assess the relevance of each label.

The number of benchmark datasets for label ranking is still relatively small. A final contribution of this project is the adaptation from a multivariate regression problem into a label ranking dataset (Chapter 5). We adapted the dataset from the COIL 1999 Competition Data, taken from the UCI Repository [96], concerning the frequencies of algae populations in different environments, which we refer to as Algae.

1.4 Thesis outline

This thesis is presented as a series of papers in the form of self-contained chapters. These are either papers that have been published or that have been submitted for publication. The dissertation consists of 6 chapters following this introductory chapter.

Chapter 2, Preference Rules [37], presents an empirical study on Label Ranking Association Rules and Pairwise Association Rules. This paper, which has been submitted to the *Information Fusion* journal, is an extension of previous work, Mining Association Rules for Label Ranking [36].

Chapter 3, Entropy-based discretization methods for ranking data [39], presents a supervised approach to discretize datasets with target rankings. This chapter, which is published in the *Information Sciences* journal, is based on preliminary work published in the proceedings of the Discovery Science 2013 conference, Singapore [40].

In Chapter 4, Label Ranking Forests [32], we can find a successful adaption of ensembles of trees for label ranking problems, which has been published in the *Expert Systems* journal. This work is an extension to the preliminary work published in EPIA 2015, in which Entropy Ranking Trees, were proposed [35].

Chapter 5, Exceptional Preferences Mining [34], proposes an approach to look for exceptional behavior in label ranking datasets. This paper is published in the proceedings of the Discovery Science 2016 conference held in Bari, Italy.

Chapter 6, Permutation Tests for Label Ranking [38], presents a smaller contribution where, semi-synthetic datasets used in Label Ranking community, where evaluated with different tests. This chapter is published in the local proceedings of the *BENELUX conference on artificial intelligence 2015.*

Finally, Chapter 7, gives an overview of the main contributions and findings in this PhD dissertation.

Chapter 2

Preference Rules

Cláudio Rebelo de Sá, Paulo Azevedo, Carlos Soares, Alípio Mário Jorge, Arno Knobbe

submitted to Information Fusion Journal, 2016

Abstract

In this paper we investigate two variants of association rules for preference data, Label Ranking Association Rules and Pairwise Association Rules. Label Ranking Association Rules (LRAR) are the equivalent of Class Association Rules (CAR) for the Label Ranking task. In CAR, the consequent is a single class, to which the example is expected to belong to. In LRAR, the consequent is a ranking of the labels. The generation of LRAR requires special support and confidence measures to assess the similarity of rankings. In this work, we carry out a sensitivity analysis of these similarity-based measures. We want to understand which datasets benefit more from such measures and which parameters have more influence in the accuracy of the model. Furthermore, we propose an alternative type of rules, the Pairwise Association Rules (PAR), which are defined as association rules with a set of pairwise preferences in the consequent. While PAR can be used both as descriptive and predictive models, they are essentially descriptive models. Experimental results show the potential of both approaches.

2.1 Introduction

Label ranking is a topic in the machine learning literature [57, 26, 123] that studies the problem of learning a mapping from instances to rankings over a finite number of predefined labels. One characteristic that clearly distinguishes label ranking problems from classification problems is the order relation between the labels. While a classifier aims at finding the true class on a given unclassified example, the label ranker will focus on the relative preferences between a set of labels/classes. These relations represent relevant information from a decision support perspective, with possible applications in various fields such as elections, dominance of certain species over the others, user preferences, etc.

Due to its intuitive representation, Association Rules [4] have become very popular in data mining and machine learning tasks (e.g. Mining rankings [70], Classification [97] and even Label Ranking [36], etc). The adaptation of AR for label ranking, Label Ranking Association Rules (LRAR) [36], are similar to their classification counterpart, Class Association Rules (CAR) [97]. LRAR can be used for predictive or descriptive purposes.

LRAR are relations, like typical association rules, between an antecedent and a consequent ($A \to C$), defined by interest measures. The distinction lies in the fact that the consequent is a complete ranking. Because the degree of similarity between rankings can vary, it lead to several interesting challenges. For instance, how to treat rankings that are very similar but not exactly equal. To tackle this problem, similarity-based interest measures were defined to evaluate LRAR. Such measures can be applied to existing rule generation methods [36] (e.g. APRIORI [4]).

One important issue for the use of LRAR is the threshold that determines what should and should not be considered sufficiently similar. Here we present the results of sensitivity analysis study to show how LRAR behave in different scenarios, to understand the effect of this threshold better. Whether there is a rule of thumb or this threshold is data-specific is the type of questions we investigate here. Ultimately we also want to understand which parameters have more influence in the predictive accuracy of the method.

Another important issue is related to the large number of distinct rankings. Despite the existence of many competitive approaches in Label Ranking, Decision trees [120, 26], k-Nearest Neighbor [17, 26] or LRAR [36], problems with a large number of distinct rankings can be hard to predict. One real-world example with a relatively large number of rankings, is the sushi dataset [81]. This dataset compares demographics of 5000 Japanese citizens with their preferred sushi types. With only 10 labels, it has more than 4900 distinct rankings. Even though it has been known in the preference learning community for a while, no results with high predictive accuracy have been published, to the best of our knowledge. Cases like this have motivated the appearance of new approaches, e.g. to mine ranking data [70], where association rules are used to find patterns within rankings.

We propose a method which combines the two approaches mentioned above [36, 70], because it can could contribute to a better understanding of the datasets mentioned above. We define Pairwise Association Rules (PAR) as association rules with one or more pairwise comparisons in the consequent. In this work we present an approach to identify PAR and analyze the findings in two real world datasets.

By decomposing rankings into the unitary preference relation i.e. *pairwise comparisons*, we can look for sub-ranking patterns. From which, as explained before, we expect to find more frequent patterns than with complete rankings.

LRAR and PARs can be regarded as a specialization of general association rules that are obtained from data containing preferences, which we refer to as *Preference Rules*. These two approaches are complementary in the sense that they can give different insights from preference data. We use LRAR and PAR in this work as predictive and descriptive models, respectively.

The paper is organized as follows: Sections 2.2 and2.3 introduce the task of association rule mining and the label ranking problem, respectively; Section 2.4 describes the Label Ranking Association Rules and Section 2.5 the Pairwise Association Rules proposed here; Section 2.6 presents the experimental setup and discusses the results; finally, Section 2.7 concludes this paper.

2.2 Association Rule Mining

An association rule (AR) is an implication: $A \rightarrow C$ where $A \bigcap C = \emptyset$ and $A, C \subseteq desc\,(\mathbb{X})$, where $desc\,(\mathbb{X})$ is the set of descriptors of instances in the instance space $\mathbb{X}$, typically pairs $\langle attribute, value \rangle$. The training data is represented as $D = \{\langle x_i \rangle\}$, $i = 1, \dots, n$, where x_i is a vector containing the values $x_i^j, j = 1, \dots, m$ of m independent variables, $\mathcal{A}$, describing instance i. We also denote $desc(x_i)$ as the set of descriptors of instance x_i.

2.2.1 Interest measures

There are many interest measures to evaluate association rules [106], but typically they are characterized by *support* and *confidence*. Here, we summarize some of the most common, assuming a rule $A \rightarrow C$ in D.

Support percentage of the instances in D that contain A and C:

$$sup\,(A \rightarrow C) = \frac{\#\{x_i | A \cup C \subseteq desc(x_i), x_i \in D\}}{n}$$

Confidence percentage of instances that contain C from the set of instances that contain A:

$$conf\,(A \rightarrow C) = \frac{sup\,(A \rightarrow C)}{sup\,(A)}$$

Coverage proportion of examples in D that contain the antecedent of a rule: *coverage* [65]:

$$coverage\,(A \rightarrow C) = sup\,(A)$$

We say that a rule $A \rightarrow C$ covers an instance x, if $A \subseteq desc\,(x)$.

Lift measures the independence of the consequent, C, relative to the antecedent, A:

$$lift\,(A \rightarrow C) = \frac{sup(A \rightarrow C)}{sup(A) \cdot sup(C)}$$

Lift values vary from 0 to $+\infty$. If A is independent from C then $lift(A \rightarrow C) \sim 1$.

2.2.2 APRIORI Algorithm

The original method for induction of AR is the APRIORI algorithm, proposed in 1994 [4]. APRIORI identifies all AR that have support and confidence higher than a given minimal support threshold ($minsup$) and a minimal confidence threshold ($minconf$), respectively. Thus, the model generated is a set of AR, $\mathcal{R}$, of the form $A \rightarrow C$, where $A, C \subseteq desc\,(\mathbb{X})$, and $sup(A \rightarrow C) \geq minsup$ and $conf(A \rightarrow C) \geq minconf$. For a more detailed description see [4].

Despite the usefulness and simplicity of APRIORI, it runs a time consuming candidate generation process and needs substantial time and memory space, proportional to the number of possible combinations of the descriptors. Additionally it needs multiple scans of the data and typically generates a very large number of rules. Because of this, many alternative methods were previously proposed, such as hashing [107], dynamic itemset counting [21], parallel and distributed mining [108] and mining integrated into relational database systems [119].

In contrast to itemset-based algorithms, which compute frequent itemsets and rule generation in two steps, there are rule-based approaches such as FP-Growth (Frequent pattern growth method) [67]. This means that, rules are generated at the same time as frequent itemsets are computed.

2.2.3 Pruning

AR algorithms typically generate a large number of rules (possibly tens of thousands), some of which represent only small variations from others. This is known as the rule explosion problem [80] which should be dealt with by pruning mechanisms. Many rules must be discarded for computational and simplicity reasons.

Pruning methods are usually employed to reduce the amount of rules without reducing the quality of the model. For example, an AR algorithm might find rules for which the confidence is only marginally improved by adding further conditions to their antecedent.Another example is when the consequent C of a rule $A \rightarrow C$ has the same distribution independently of the antecedent A. In these cases, we should not consider these rules as meaningful.

Improvement A common pruning method is based on the improvement that a refined rule yields in comparison to the original one [80]. The *improvement* of a rule is defined as the smallest difference between the confidence of a rule and the confidence of all sub-rules sharing the same consequent:

$$imp(A \rightarrow C) = min(\forall A' \subset A, conf(A \rightarrow C) - conf(A' \rightarrow C))$$

As an example, if one defines a minimum improvement $minImp = 1\%$, the rule $A' \rightarrow C$ will be kept if $conf(A' \rightarrow C) - conf(A \rightarrow C) \geq 1\%$, where $A \subset A'$.

If $imp(A \rightarrow C) > 0$ we say that $A \rightarrow C$ is a productive rule.

Significant rules Another way to prune non productive rules is to use statistical tests [125]. A rule is *significant* if the confidence improvement over all its generalizations is statistically significant. The rule $A \rightarrow C$ is significant if $\forall A' \rightarrow C, A' \subset A$ the difference $conf(A \rightarrow C) - conf(A' \rightarrow C)$ is statistically significant for a given significance level (α).

2.3 Label Ranking

In Label Ranking (LR), given an instance x from the instance space $\mathbb{X}$, the goal is to predict the ranking of the labels $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$ associated with x [74]. A ranking can be represented as a *strict total order* over $\mathcal{L}$, defined on the permutation space Ω.

The LR task is similar to the classification task, where instead of a class we want to predict a ranking of labels. As in classification, we do not assume the existence of a deterministic $\mathbb{X} \rightarrow \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω [26]. This means that, for each $x \in \mathbb{X}$, there exists a probability distribution $\mathcal{P}(\cdot|x)$ such that, for every $\pi \in \Omega$, $\mathcal{P}(\pi|x)$ is the probability that π is the ranking associated with x. The goal in LR is to learn the mapping $\mathbb{X} \rightarrow \Omega$. The training data contains a set of instances $D = \{\langle x_i, \pi_i \rangle\}$, $i = 1, \ldots, n$, where x_i is a vector containing the values $x_i^j, j = 1, \ldots, m$ of m independent variables, $\mathcal{A}$, describing instance i and π_i is the corresponding target ranking.

The rankings can be either total or partial orders.

Total orders A *strict total order* over $\mathcal{L}$ is defined as:[1]

$$\{\forall (\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succ \lambda_b \vee \lambda_b \succ \lambda_a\}$$

which represents a *strict ranking* [123], a *complete ranking* [57], or simply a *ranking*. A strict total order can also be represented as a permutation π of the set $\{1, \ldots, k\}$, such that $\pi(a)$ is the position, or *rank*, of λ_a in π. For example, the *strict total order* $\lambda_1 \succ \lambda_2 \succ \lambda_3 \succ \lambda_4$ can be represented as $\pi = (1, 2, 3, 4)$.

However, in real-world ranking data, we do not always have clear and unambiguous preferences, i.e. strict total orders [15]. Hence, sometimes we have

[1]For convenience, we say *total order* but in fact we mean a *totally ordered set*. Strictly speaking, a *total order* is a binary relation.

to deal with *indifference* and *incomparability.* For illustration purposes, let us consider the scenario of elections, where a set of n voters vote on k candidates. If a voter feels that two candidates have identical proposals, then these can be expressed as indifferent so they are assigned the same rank (i.e. a tie).

To represent ties, we need a more relaxed setting, called *non-strict total orders*, or simply *total orders*, over $\mathcal{L}$, by replacing the binary strict order relation, $\succ$, with the binary partial order relation, $\succeq$:

$$\{\forall\,(\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succeq \lambda_b \vee \lambda_b \succeq \lambda_a\}$$

These non-strict total orders can represent *partial rankings* (rankings with ties) [123]. For example, the *non-strict total order* $\lambda_1 \succ \lambda_2 = \lambda_3 \succ \lambda_4$ can be represented as $\pi = (1, 2, 2, 3)$.

Additionally, real-world data may lack preference data regarding two or more labels, which is known as *incomparability.* Continuing with the elections example, the lack of information about one or two of the candidates, λ_a and λ_b, leads to incomparability, $\lambda_a \perp \lambda_b$. In other words, the voter cannot decide whether the candidates are equivalent or select one as the preferred, because he does not know the candidates. Incomparability should not be confused with intrinsic properties of the objects, as if we are comparing apples and oranges. Instead, it is like trying to compare two different types of apple without ever having tried either. In this cases, we can use *partial orders.*

Partial orders Similarly to *total orders*, there are *strict* and *non-strict partial orders.* Let us consider the *non-strict partial orders* (which can also be referred to as *partial orders*) over $\mathcal{L}$:

$$\{\forall\,(\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succeq \lambda_b \vee \lambda_b \succeq \lambda_a \ \vee \lambda_a \perp \lambda_b\}$$

We can represent partial orders with *subrankings* [70]. For example, the *partial order* $\lambda_1 \succ \lambda_2 \succ \lambda_4$ can be represented as $\pi = (1, 2, 0, 4)$, where 0 represents $\lambda_1, \lambda_2, \lambda_4 \perp \lambda_3$.

2.3.1 Methods

Several learning algorithms were proposed for modeling label ranking data in recent years. These can be grouped as decomposition-based or direct.

Decomposition-based methods divide the problem into several simpler problems (e.g., multiple binary problems). An example is ranking by pairwise comparisons [57] and mining rank data [70]. *Direct methods* treat the rankings as target objects without any decomposition. Examples of that include decision trees [120, 26], k-Nearest Neighbors [17, 26] and the linear utility transformation [68, 41]. This second group of algorithms can be divided into two approaches. The first one contains methods that are based on statistical distributions of rankings (e.g. [26]), such as Mallows [91], or Plackett-Luce [24]. The other group of methods are based on measures of similarity or correlation between rankings (e.g. [120, 6]).

LR-specific preprocessing methods have also been proposed, e.g. MDLP-R [40] and EDiRa [39]. Both are *direct methods* and based on measures of similarity. Considering that supervised discretization approaches usually provide better results than unsupervised methods [46], such methods can be of a great importance in the field. In particular, for AR-like algorithms, such as the ones proposed in this work, which are typically not suitable for numerical data.

For more information on label ranking learning methods, more information ca be found in [57].

Label Ranking by Learning Pairwise Preferences

Ranking by pairwise comparisons basically consists of reducing the problem of ranking into several classification problems. In the learning phase, the original problem is formulated as a set of pairwise preferences problem. Each problem is concerned with one pair of labels of the ranking, $(\lambda_i, \lambda_j) \in \mathcal{L}, 1 \leq i < j \leq k$. The target attribute is the relative order between them, $\lambda_i \succ \lambda_j$. Then, a separate model $\mathcal{M}_{ij}$ is obtained for each pair of labels. Considering $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$, there will be $h = \frac{k(k-1)}{2}$ classification problems to model.

In the prediction phase, each model is applied to every pair of labels to obtain a prediction of their relative order. The predictions are then combined to derive rankings, which can be done in several ways. The simplest is to order the labels, for each example, considering the predictions of the models $\mathcal{M}_{ij}$ as votes. This topic has been well studied and documented [55, 74].

2.3.2 Evaluation

Given an instance x_i with label ranking π_i and a ranking $\hat{\pi}_i$ predicted by a LR model, several loss functions on Ω can be used to evaluate the accuracy of the prediction. One such function is the number of discordant label pairs:

$$\mathcal{D}\left(\pi, \hat{\pi}\right) = \#\{(a,b)|\pi(a) > \pi(b) \wedge \hat{\pi}(a) < \hat{\pi}(b)\}$$

If there are no discordant label pairs, the distance $\mathcal{D} = 0$. Alternatively, the function to define the number of concordant pairs is:

$$\mathcal{C}\left(\pi, \hat{\pi}\right) = \#\{(a,b)|\pi(a) > \pi(b) \wedge \hat{\pi}(a) > \hat{\pi}(b)\}$$

Kendall Tau Kendall's τ coefficient [85] is the normalized difference between the number of concordant, $\mathcal{C}$, and discordant pairs, $\mathcal{D}$:

$$\tau\left(\pi, \hat{\pi}\right) = \frac{\mathcal{C} - \mathcal{D}}{\frac{1}{2}k\left(k-1\right)}$$

where $\frac{1}{2}k\left(k-1\right)$ is the number of possible pairwise combinations, $\binom{k}{2}$. The values of this coefficient range from $[-1, 1]$, where $\tau\left(\pi, \pi\right) = 1$ if the rankings are equal and $\tau(\pi, \pi^{-1}) = -1$ if π^{-1} denotes the inverse order of π (e.g. $\pi = (1,2,3,4)$ and $\pi^{-1} = (4,3,2,1)$). Kendall's τ can also be computed in the presence of ties, using tau-b [5].

An alternative measure is the Spearman's rank correlation coefficient [118].

Gamma coefficient If we want to measure the correlation between two partial orders (subrankings), or between total and partial orders, we can use the Gamma coefficient [93]:

$$\gamma\left(\pi, \hat{\pi}\right) = \frac{\mathcal{C} - \mathcal{D}}{\mathcal{C} + \mathcal{D}}$$

Which is identical to Kendall's τ coefficient in the presence of strict total orders, because $\mathcal{C} + \mathcal{D} = \frac{1}{2}k\left(k-1\right)$.

Weighted rank correlation measures When it is important to give more relevance to higher ranks, a weighted rank correlation coefficient can be used. They are typically adaptations of existing similarity measures, such as ρ_w [110], which is based on Spearman's coefficient.

These correlation measures are not only used for evaluation estimation, they can be used within learning [36] or preprocessing [39] models. Since Kendall's τ has been used for evaluation in many recent LR studies [26, 40], we use it here as well.

The accuracy of a label ranker can be estimated by averaging the values of any of the measures explained here, over the rankings predicted for a set of test examples. Given a dataset, $D = \{\langle x_i, \pi_i \rangle\}$, $i = 1, \ldots, n$, the usual resampling strategies, such as holdout or cross-validation, can be used to estimate the accuracy of a LR algorithm.

2.4 Label Ranking Association Rules

Association rules were originally proposed for descriptive purposes. However, they have been adapted for predictive tasks such as classification (e.g., [97]). Given that label ranking is a predictive task, the adaptation of AR for label ranking comes in a natural way. A *Label Ranking Association Rule* (LRAR) [36] is defined as:

$$A \rightarrow \pi$$

where $A \subseteq desc\,(\mathbb{X})$ and $\pi \in \Omega$. Let $\mathcal{R}_\pi$ be the set of *label ranking association rules* generated from a given dataset. When an instance x is covered by the rule $A \rightarrow \pi$, the predicted ranking is π. A rule $r_\pi : A \rightarrow \pi, r_\pi \in \mathcal{R}_\pi$, covers an instance x, if $A \subseteq desc(x)$.

We can use the CAR framework[97] for LRAR. However this approach has two important problems. First, the number of classes can be extremely large, up to a maximum of $k!$, where k is the size of the set of labels, $\mathcal{L}$. This means that the amount of data required to learn a reasonable mapping $\mathbb{X} \rightarrow \Omega$ is unreasonably large.

The second disadvantage is that this approach does not take into account the differences in nature between label rankings and classes. In classification, two examples either have the same class or not. In this regard, label ranking is more similar to regression than to classification. In regression, a large number of observations with a given target value, say 5.3, increases the probability of observing similar values, say 5.4 or 5.2, but not so much for very different values, say -3.1 or 100.2. This property must be taken into account in the induction of prediction models. A similar reasoning can be made in label ranking. Let us consider the case of a data set in which ranking $\pi_a = (1,2,3,4)$ occurs in 1% of the examples. Treating rankings as classes would mean that $P(\pi_a) = 0.01$. Let us further consider that the rankings $\pi_b = (1,2,4,3)\,, \pi_c = (1,3,2,4)$ and $\pi_d = (2,1,3,4)$, which are obtained from π_a by swapping a single pair of adjacent labels, occur in 50% of the examples. Taking into account the stochastic nature of these rankings [26], $P(\pi_a) = 0.01$ seems to underestimate the probability of observing π_a. In other words it is expected that the observation of π_b, π_c and π_d increases the probability of observing π_a and vice-versa, because they are similar to each other.

This affects even rankings which are not observed in the available data. For example, even though a ranking is not present in the dataset it would not be entirely unexpected to see it in future data. This also means that it is possible to compute the probability of unseen rankings.

To take all this into account, similarity-based interestingness measures were proposed to deal with rankings [36].

2.4.1 Interestingness measures in Label Ranking

As mentioned before, because the degree of similarity between rankings can vary, similarity-based measures can be used to evaluate LRAR. These measures are able to distinguish rankings that are *very similar* from rankings that are very *very distinct*. In practice, the measures described below can be applied to existing rule generation methods [36] (e.g. APRIORI [4]).

Support The support of a ranking π should increase with the observation of similar rankings and that variation should be proportional to the similarity. Given a measure of similarity between rankings $s(\pi_a, \pi_b)$, we can adapt the concept of support of the rule $A \rightarrow \pi$ as follows:

$$sup_{lr}(A \rightarrow \pi) = \frac{\displaystyle\sum_{i:A\subseteq desc(x_i)} s(\pi_i, \pi)}{n}$$

Essentially, what we are doing is assigning a weight to each target ranking π_i in the training data that represents its contribution to the probability that π may be observed. Some instances $x_i \in \mathbb{X}$ give a strong contribution to the support count (i.e., 1), while others will give a weaker or even no contribution at all.

Table 2.1: An example of a label ranking dataset.

		π_1	π_2	π_3
TID	$\mathcal{A}_1$	$(1,3,2)$	$(2,1,3)$	$(2,3,1)$
1	L	0.33	0.00	1.00
2	L	0.00	1.00	0.00
3	L	1.00	0.00	0.33

Any function that measures the similarity between two rankings or permutations can be used, such as Kendall's τ [85] or Spearman's ρ [118]. The function used here is of the form:

$$s(\pi_a, \pi_b) = \begin{cases} s'(\pi_a, \pi_b) & \text{if } s'(\pi_a, \pi_b) \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where s' is a similarity function. This general form assumes that below a given threshold, θ, is not useful to discriminate between different rankings, as they are so different from π_a. This means that, the support sup_{lr} of $A \to \pi_a$ will be based only on the items of the form $\langle A, \pi_b \rangle$, for all π_b where $s'(\pi_a, \pi_b) > \theta$).

Many functions can be used as s'. However, given that the loss function we aim to minimize is known beforehand, it makes sense to use it to measure the similarity between rankings. Therefore, we use Kendall's τ as s'.

Concerning the threshold, given that anti-monotonicity can only be guaranteed with non-negative values [109], it implies that $\theta \geq 0$. Therefore we think that $\theta = 0$ is a reasonable default value, because it separates between the positive and negative correlation between rankings.

Table 2.1 shows an example of a label ranking dataset represented according to this approach. Instance $(\{\mathcal{A}_1 = L, \pi_3\})$ (TID=1) contributes to the support count of ruleitem $\langle \{\mathcal{A}_1 = L\}, \pi_3 \rangle$ with 1, as expected. However, that same instance, will also give a contribution of 0.33 to the support count of ruleitem $\langle \{\mathcal{A}_1 = L\}, \pi_1 \rangle$, given their ranking similarity. On the other hand, no contribution to the support of ruleitem $\langle \{\mathcal{A}_1 = L\}, \pi_2 \rangle$ is given, because these rankings are clearly different. This means that $sup_{lr}\left(\langle \{\mathcal{A}_1 = L\}, \pi_3 \rangle\right) = \frac{1+0.33}{3}$.

Confidence The confidence of a rule $A \to \pi$ comes in a natural way if we replace the classical measure of support with the similarity-based sup_{lr}.

$$conf_{lr}\left(A \to \pi\right) = \frac{sup_{lr}\left(A \to \pi\right)}{sup\left(A\right)}$$

Improvement Improvement in association rule mining is defined as the smallest difference between the confidence of a rule and the confidence of all sub-rules sharing the same consequent [80]. In LR it is not suitable to compare targets simply as equal or different (Section 2.4). Therefore, to implement pruning based on improvement for LR, some adaptation is required as well. Given that the relation between target values is different from classification, as discussed in Section 2.4.1, we have to limit the comparison between rules with different consequents, if $S'(\pi, \pi') \geq \theta$.

Improvement for Label Ranking is defined as:

$$imp_{lr}(A \to \pi) = min(conf_{lr}(A \to \pi) - conf_{lr}(A' \to \pi'))$$

for $\forall A' \subset A$, and $\forall (\pi, \pi')$ where $S'(\pi', \pi) \geq \theta$. As an illustrative example, consider the two rules $r_1 : A_1 \to (1,2,3,4)$ and $r_2 : A_2 \to (1,2,4,3)$, where A_2 is a superset of A_1, $A_1 \subset A_2$. If $S'((1,2,3,4), (1,2,4,3)) \geq \theta$ then r_2 will only be kept if, and only if, $conf(r_1) - conf(r_2) \geq minImp$.

Lift The *lift* measures the independence between the consequent and the antecedent of the rule [9]. The adaptation of lift for LRAR is straightforward since it only depends the concept of support, for which a version for LRAR already exists:

$$lift_{lr}(A \to \pi) = \frac{sup_{lr}(A \to \pi)}{sup(A) \cdot sup_{lr}(\pi)}$$

2.4.2 Generation of LRAR

Given the adaptations of the interestingness measures proposed, the task of learning LRAR can be defined essentially in the same way as the task of learning AR, i.e. to identify the set of LRAR that has a support and a confidence higher than the thresholds defined by the user. More formally, given a training set $D = \{\langle x_i, \pi_i \rangle\}, i = 1, \ldots, n$, the algorithm aims to create a set of high accuracy rules $\mathcal{R}_\pi = \{r_\pi : A \to \pi\}$ to cover a test set $T = \{\langle x_j \rangle\}, j = 1, \ldots, s$. If $\mathcal{R}_\pi$ does not cover some $x_j \in T$, a *DefaultRanking* (Section 2.4.3) is assigned to it.

Implementation of LRAR in CAREN

The association rule generator we are using is CAREN [10]. [2] CAREN implements an association rule algorithm to derive rule-based prediction models, like CAR and LRAR. For Label Ranking datasets, CAREN derives association rules where the consequent is a complete ranking.

CAREN is specialized in generating association rules for predictive models and employs a bitwise depth-first frequent pattern mining algorithm. Rule pruning is performed using a Fisher exact test [10]. Like CMAR [95], CAREN is a rule-based algorithm rather than itemset-based. This means that, frequent itemsets are derived at the same time as rules are generated, whereas itemset-based algorithms carry out the two tasks in two separated steps.

Rule-based approaches allow for different pruning methods. For example, let us consider the rule $A \rightarrow \lambda$, where λ is the most frequent class in the examples covering A. If $sup\,(A \rightarrow \lambda) < minsup$ then there is no need to search for a superset of A, A^*, since any rule of the form $A^* \rightarrow \lambda, A \subset A^*$ cannot have a support higher than *minsup*.

CAREN generates significant rules [125]. Statistical significance of a rule is evaluated using a Fisher Exact Test by comparing its support to the support of its direct generalizations. The direct generalizations of a rule $A \rightarrow C$ are $\emptyset \rightarrow C$ and $(A \setminus \{a\}) \rightarrow C$ where a is a single item.

The final set of rules obtained define the label ranking prediction model, which we can also refer as the *label ranker*.

CAREN also employs prediction for strict rankings using *consensus ranking* (Section 2.4.3), best rule, among others.

2.4.3 Prediction

A very straightforward method to generate predictions using a label ranker is used. The set of rules $\mathcal{R}_\pi$ can be represented as an ordered list of rules, by some user defined measure of relevance:

$$< r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_t} >$$

As mentioned before, a rule $r^*_\pi : A^* \rightarrow \pi^*$ covers (or matches) an instance $x_i \in T$, if $A^* \subseteq desc(x_i)$. If only one rule, r^*_π, matches x_i, the predicted ranking

[2] `http://www4.di.uminho.pt/~pja/class/caren.html`

for x_i is π^*. However, in practice, it is quite common to have more than one rule covering the same instance x_i, $\mathcal{R}^*_\pi(x_j) \subseteq \mathcal{R}_\pi$. In $\mathcal{R}^*_\pi(x_j)$ there can be rules with conflicting ranking recommendations. There are several methods to address those conflicts, such as selecting the best rule, calculating the majority ranking, etc. However, it has been shown that a ranking obtained by ordering the average ranks of the labels across all rankings minimizes the euclidean distance to all those rankings [84]. In other words, it maximizes the similarity according to Spearman's ρ [118]. This can be referred to as the *average ranking* [17].

Given any set of rankings $\{\pi_i\}$ $(i = 1, \ldots, s)$ with k labels, we compute the *average ranking* as:

$$\overline{\pi}(j) = \frac{\sum_{i=1}^{s} \pi_i(j)}{s}, j = 1, \ldots, k \tag{2.2}$$

The average ranking $\overline{\pi}$ can be obtained if we rank the values of $\overline{\pi}(j), j = 1, \ldots, k$. A weighted version of this method can be obtained by using the *confidence* or *support* of the rules in $\mathcal{R}^*_\pi(x_j)$ as weights.

Default rules

As in classification, in some cases, the label ranker might not find any rule that covers a given instance x_j, so $\mathcal{R}^*_\pi(x_j) = \emptyset$. To avoid this, we need to define a *default rule*, $r_\emptyset$, which can be used in such cases:

$$\{\emptyset\} \rightarrow \textit{default ranking}$$

A *default class* is also often used in classification tasks [66], which is usually the majority class of the training set D. In a similar way, we could define the majority ranking as our *default ranking*. However, some label ranking datasets have as many rankings as instances, making the majority ranking not so representative.

As mentioned before, the *average ranking* (Equation 2.2) of a set of rankings, minimizes the distance to all rankings in that set [84]. Hence we can use the *average ranking* as the *default ranking*.

2.4.4 Parameter tuning

Due to the intrinsic nature of each different dataset, or even of the preprocessing methods used to prepare the data (e.g., the discretization method), the maximum *minsup*/*minconf* needed to obtain a rule set $\mathcal{R}_\pi$, that covers all the examples, may vary significantly [98]. The trivial solution would be, for example, to set *minconf* $= 0$ which would generate many rules, hence increasing the coverage. However, this rule would probably lead to a lot of uninteresting rules as well, as the model would overfit the data. Then, our goal is to obtain a rule set $\mathcal{R}_\pi$ which gives maximal coverage while keeping high confidence rules.

Let us define M as the coverage of the model i.e. the coverage of the set of rules $\mathcal{R}_\pi$. Algorithm 1 represents a simple, heuristic method to determine the *minconf* that obtains the rule set such that a certain minimal coverage is guaranteed $minM$.

Algorithm 1 Confidence tuning algorithm

```
Given $minsup$ and $step$
$minconf = 100\%$
while $M < minM$ do
    $minconf = minconf - step$
    Run CAREN with ($minsup$,$minconf$) and determine $M$
end while
    return $minconf$
```

This procedure has the important advantage that it does not take into account the accuracy of the rule sets generated, thus reducing the risk of overfitting.

2.5 Pairwise Association Rules

Association rules use a sets of descriptors to represent meaningful subsets of the data [69], hence providing an easy interpretation of the patterns mined. Due to the intuitive representation, since its first application in the market basket analysis [2], they have become very popular in data mining and machine learning tasks (Mining rankings [70], Classification [97], Label Ranking [36], etc).

LRAR proved to be an effective predictive model, however they are designed to find complete rankings. Despite its similarity measures, which take into account possible ranking noise, it does not capture subranking patterns because it will always try to infer complete rankings. On the other hand, association rules were used to find patterns within rankings [70], however, they do not relate it with the independent variables. Besides, in [70], the consequent is limited to one pairwise comparison.

In this work, we propose a decomposition method to look for meaningful associations between independent variables and preferences (in the form of pairwise comparisons), the Pairwise Association Rules (PAR), which can be regarded as predictive or descriptive model. We define PAR as:

$$A \rightarrow \{\lambda_a \succeq \lambda_b \vee \lambda_a \perp \lambda_b \ \vee \lambda_a = \lambda_b | \lambda_a, \lambda_b \in \mathcal{L}\}$$

where, as in the original AR paper [4], we allow rules with multiple items, not only in the antecedent but also in the consequent, i.e. PAR can have multiple sets of pairwise comparisons in the consequent.

Similarly to RPC (Section 2.3.1), we decompose the target rankings into pairwise comparisons. Therefore, PAR can be obtained from data with strict rankings, partial rankings and subrankings. [3]

Contrary to LRAR, we use the same interestingness measures that are also used in typical AR approaches, instead of the similarity-based versions defined for LR problems, i.e. *sup*, *conf*, etc. This allows PAR to filter out non-frequent/interesting patterns and makes it more difficult to derive strict rankings. When methods cannot find interesting rules with enough pairwise comparisons to define a strict ranking, partial rankings, subrankings or even with sets of disjoint pairwise comparisons can be found. This is, interest measures are defining the borders between what the model will keep or abstain.

Abstention is used in machine learning to describe the option to not make a prediction when the confidence in the output of a model is insufficient. The simplest case is classification, where the model can abstain itself to make a decision [11]. In the label ranking task, a method that makes partial abstentions was proposed in [28]. A similar reasoning is used here both for predictive and descriptive models.

More formally, let us define $D = \{\langle x_i, \pi_i \rangle\}, i = 1, \ldots, n$ where π_i can be a *complete ranking*, *partial ranking* or a *sub-ranking*. For each π of size k we

[3]To derive the PAR, we added a pairwise decomposition method to the CAREN [10] software.

can extract up to h pairwise comparisons. We consider 4 possible outcomes for each pairwise comparison:

- $\lambda_a \succeq \lambda_b$
- $\lambda_b \succeq \lambda_a$
- $\lambda_a = \lambda_b$ (indifference)
- $\lambda_a \perp \lambda_b$ (incomparability)

As an example, a PAR can be of the form:

$$A \rightarrow \lambda_1 \succ \lambda_4 \wedge \lambda_3 \succ \lambda_1 \wedge \lambda_1 \perp \lambda_2$$

The consequent can be simplified into $\lambda_3 \succ \lambda_1 \succ \lambda_4$ or represented as a subranking $\pi = (2, 0, 1, 3)$.

2.6 Experimental Results

In this section we start by describing the datasets used in the experiments, then we introduce the experimental setup and finally present the results obtained.

2.6.1 Datasets

The data sets in this work were taken from KEBI Data Repository in the Philipps University of Marburg [26] (Table 2.2).

To illustrate domain-specific interpretations of the results, we experiment with two additional datasets. We use an adapted dataset from the 1999 COIL Competition [96], Algae [34], concerning the frequencies of algae populations in different environments. The original dataset consisted of 340 examples, each representing measurements of a sample of water from different European rivers on different periods. The measurements include concentrations of chemical substances like nitrogen (in the form of nitrates, nitrites and ammonia), oxygen and chlorine. Also the pH, season, river size and its flow velocity were registered. For each sample, the frequencies of 7 types of algae were also measured. In this work, we considered the algae concentrations as preference relations by ordering them from larger to smaller concentrations. Those with 0 frequency are placed in last position and equal frequencies are

Table 2.2: Summary of the datasets

Datasets	type	#examples	#labels	#attributes	U_π
bodyfat	B	252	7	7	94%
calhousing	B	20,640	4	4	0.1%
cpu-small	B	8,192	5	6	1%
elevators	B	16,599	9	9	1%
fried	B	40,769	5	9	0.3%
glass	A	214	6	9	14%
housing	B	506	6	6	22%
iris	A	150	3	4	3%
segment	A	2310	7	18	6%
stock	B	950	5	5	5%
vehicle	A	846	4	18	2%
vowel	A	528	11	10	56%
wine	A	178	3	13	3%
wisconsin	B	194	16	16	100%
Algae (COIL)		316	7	10	72%
Sushi		5000	10	10	98%

represented with ties. Missing values in the independent variables were set to 0.

Finally, the Sushi preference dataset [81], which is composed of demographic data about 5000 people and sushi preferences is also used. Each person sorted a set of 10 different sushi types by preference. The 10 types of sushi, are a) shrimp, b) sea eel, c) tuna, d) squid, e) sea urchin, f) salmon roe, g) egg h) fatty tuna, i) tuna roll and j) cucumber roll. Since the attribute names were not transformed in this dataset, we can make a richer analysis of it.

Table 2.2 presents a simple measure of the diversity of the target rankings, the *Unique Ranking's Proportion*, U_π. U_π is the proportion of distinct target rankings for a given dataset. As a practical example, the *iris* dataset has 5 distinct rankings for 150 instances, which results in $U_\pi = \frac{5}{150} \approx 3\%$.

2.6.2 Experimental setup

Continuous variables were discretized with two distinct methods: (1) *Entropy-based Discretization for Ranking data (EDiRa)* ([39]) and (2) *equal width* bins. *EDiRa* is the state of the art supervised discretization method in Label Ranking, while *equal width* is a simple, general method that serves as baseline.

The evaluation measure used in all experiments is Kendall's τ. A ten-fold cross-validation was used to estimate the value for each experiment. The generation of Label Ranking Association Rules (LRAR) and PAR was performed with CAREN [10] which uses a depth-first based approach.

The confidence tuning Algorithm 1 was used to set parameters. We consider that 5% seems a reasonable step value because the *minconf* can be found in, at most, 20 iterations. Given that a common value for the *minsup* in Association Rules (AR) mining is 1%, we use it as default for all datasets. We define the *minM* as 95% to get a reasonable coverage, and $minImp = 1\%$ to avoid rule explosion.

In terms of similarity functions, we use a normalized Kendall τ between the interval $[0, 1]$ as our similarity function s (Equation 2.1).

2.6.3 Results with LRAR

In the experiments described in this section we analyze the performance from different perspectives, *accuracy*, *number of rules* and *average confidence* as the similarity threshold θ varies. We expect to understand the impact of using similarity measures in the generation of LRAR and provide some insights about its usage.

LRAR, despite being based on similarity measures, are consistent with the classical concepts underlying association rules. A special case is when $\theta = 1$, where, as in CAR, only equal rankings are considered. Therefore, by varying the threshold θ we also understand how similarity-based interest measures $(0 \leq \theta < 1)$ contribute to the accuracy of the model, in comparison to frequency-based approaches $(\theta = 1)$.

We would also like to understand how some properties of the data relate the sensitivity to θ. We can extract two simple measures of ranking diversity from the datasets, the *Unique Ranking's Proportion* (U_π), mentioned before, and the *ranking entropy* [39].

Sensitivity analysis

Here we analyze how the similarity threshold θ affects the accuracy, number and quality (in terms of confidence) of LRAR.

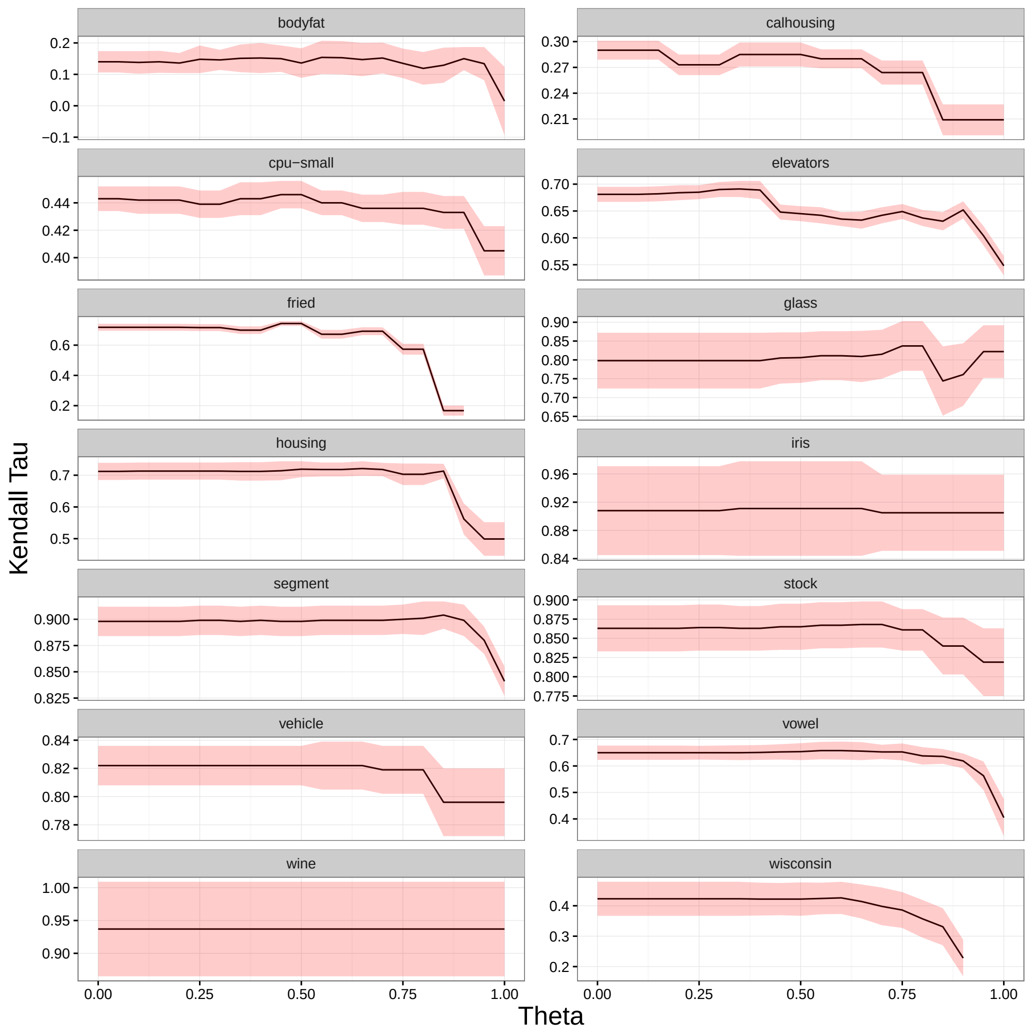


Figure 2.1: Average accuracy (Kendall τ) of CAREN as the θ varies

Accuracy In Figure 2.1 we can see the behavior of the accuracy of CAREN in terms of θ. It shows that, in general, there is a tendency for the accuracy to decrease as θ gets closer to 1. This happens in 12 out of the 14 datasets analyzed. On the other hand, in 9 out of 14 datasets, the accuracy is rather stable in the range $\theta \in [0, 0.6]$.

If we take into consideration that the model ignores all similarities between rankings for $\theta = 1$, the observed behavior seems to favor the similarity-based approach. In line with that, two extreme cases can be seen with *fried* and *wisconsin* datasets, where CAREN was not able to find any LRAR for

$\theta = 1$. [4]

Let us consider the *accuracy range*, the maximum accuracy minus the minimum accuracy. To find out which datasets are more likely to be affected by the choice of θ, we can compare their ranking entropy with the measured *accuracy range* from Figure 2.1. In Figure 2.2 we compare the accuracy range with the *ranking entropy* [39]. We can see that, the higher the entropy, the more the accuracy can be affected by the choice of θ.

Results seem to indicate that, when mining LRAR in datasets with low ranking entropy, the choice of θ is not so relevant. On the other hand, as the entropy gets bigger, a reasonable value should be $0 \leq \theta \leq 0.6$.

One interesting behavior can be found in the dataset *fried*. Despite the fact that it has a very low proportion of unique rankings, U_π (*fried*) $= 0.3\%$ (Table 2.2) its entropy is quite high (Figure 2.2). For this reason, it makes it more sensitive to θ, as seen in Figure 2.1. On the other hand, *iris* and *wine*, with very low entropy, seem unaffected by θ.

Number of rules Ideally, we would like to obtain a small number of rules with high accuracy. However, such a balance is not expected to happen frequently. Ultimately, as accuracy is the most important evaluation criterion, if a reduction in the number of rules comes with a high cost in accuracy, it is better to have more rules. Thus, it is important to understand how the number of LRAR varies with the similarity threshold θ, while taking the impact in the accuracy of the model into account as well.

In Figure 2.3 we see how many LRAR are generated per dataset as θ varies. The majority of the plots, 10 out of 14, show a decrease in the number of rules as θ gets closer to 1. As discussed before, the accuracy in general also decreases as $\theta \geq 0.6$, so let us focus on $\theta \in [0, 0.6]$.

In the interval $\theta \in [0, 0.6]$, the number of rules generated is quite stable in 9 out of 14 datasets. In the first half of this interval, $\theta \in [0, 0.3]$, it is even more remarkable for 13 datasets.

We expect the number of rules to decrease as θ increases, however, results show that the number of rules does not decrease so much, especially for values up to 0.3. This is due to the fact that θ is also used in the pruning step (Section 2.4.1), reducing the number of rules against which the improvement of an extension is measured and, thus, increasing the probability of an ex-

[4]The *default rule* was not used in these experiments because it is not related with θ.

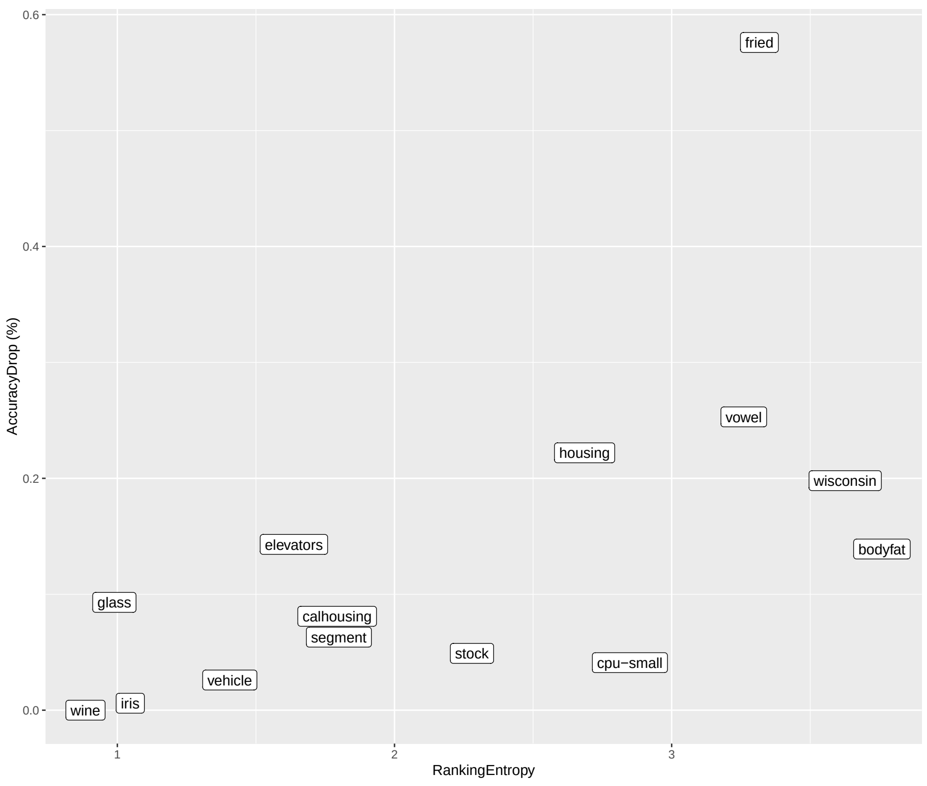


Figure 2.2: Measured *accuracy range* (Kendall τ) of CAREN in comparison to ranking entropy.

tension not being kept in the model. This means that, $minImp_{lr}$ is being effective in the reduction of LRAR.

As mentioned before, $imp_{lr}\left(A \rightarrow \pi\right)$ not only compares rules $A' \rightarrow \pi$ where $A' \subset A$, but also rules $A \rightarrow \pi'$ where $S'\left(\pi', \pi\right) \geq \theta$. In other words, with the $minImp_{lr}$ we are pruning LRAR with similar rankings too.

These results do not lead to any strong conclusions about the ideal value for θ regarding the number of rules. However, they are in line with the previous analysis of *accuracy*.

Minimum Confidence As mentioned before, we use a greedy algorithm to automatically adjust the minimum confidence in order to reduce the number of examples that are not covered by any rule. This means that the method has to adapt the value of *minconf* per dataset per θ, as seen in Figure 2.4.

In general, the *minconf* decreases in a monotonic way as θ increases. As

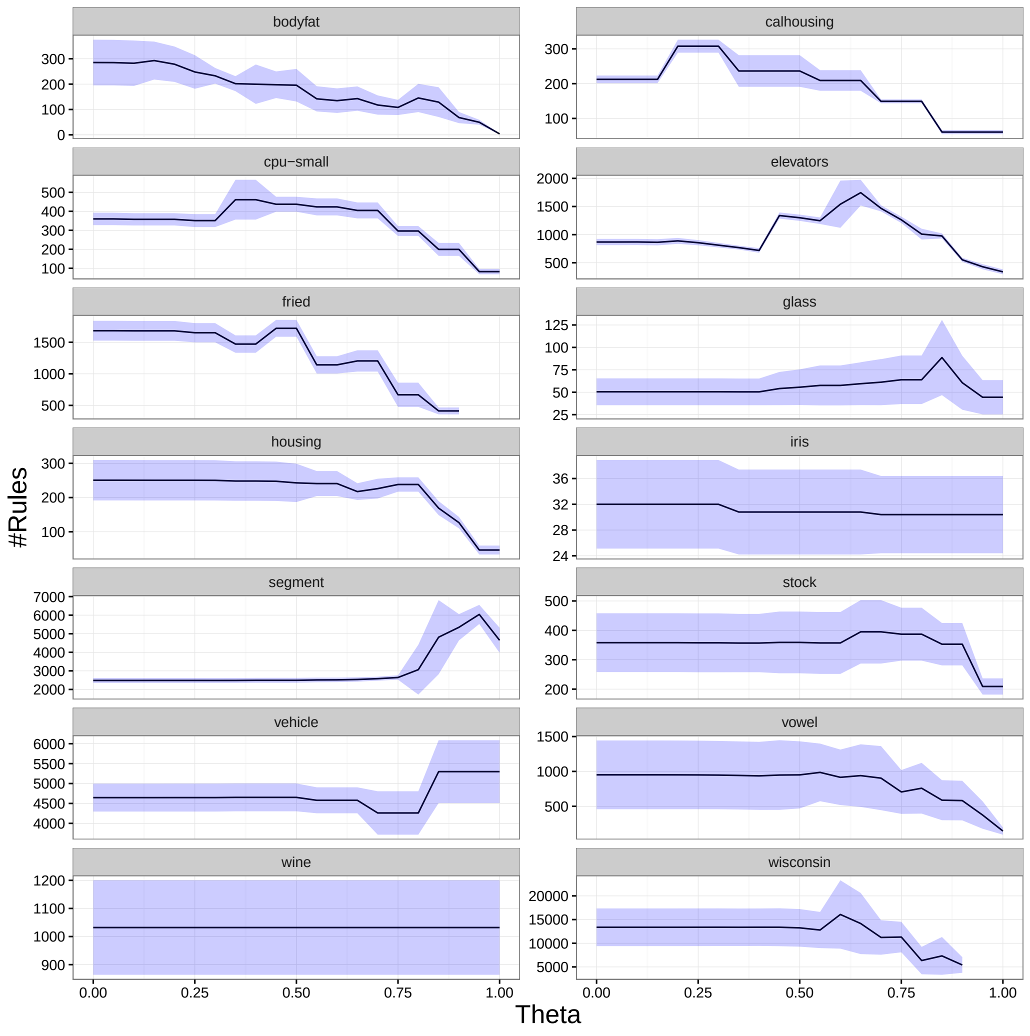


Figure 2.3: Number of Label Ranking Association Rules generated by CAREN as the θ varies

$\theta \approx 1$ the *minconf* gets to its minimum with 13 out of 14 datasets, which is consistent with the accuracy plots (Figure 2.1). This means that, if we want to generate rules with as much confidence as possible, we should use the minimum θ, i.e. $\theta = 0$.

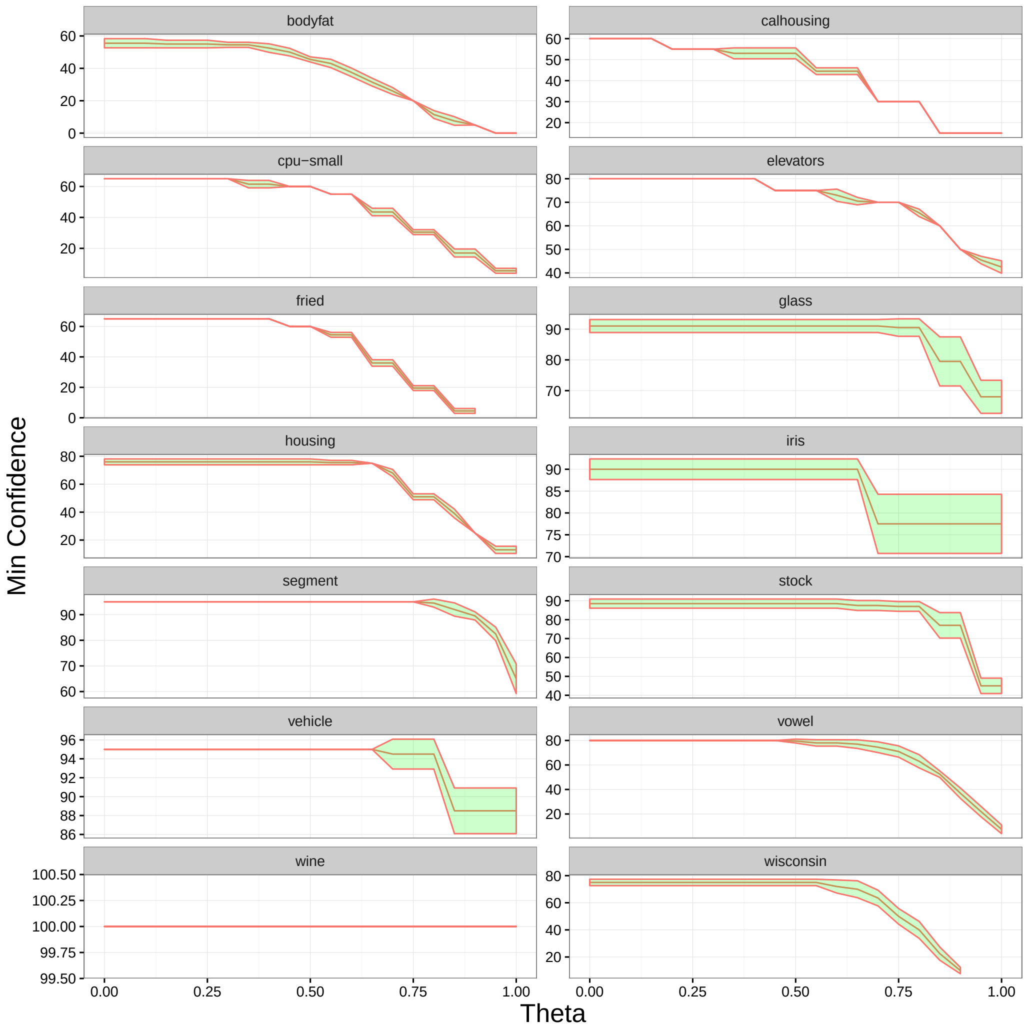


Figure 2.4: Mininum confidence adjusted to CAREN as the θ varies

Support versus accuracy We vary the minimum support threshold, *minsup*, to test how it affects the accuracy of our learner. A similar study has been carried out on CBA [75]. Specifically, we vary the *minsup* from 0.1% to 10%, using a step size of 0.1%. Due to the complexity of these experiments, we only considered the six smallest datasets.

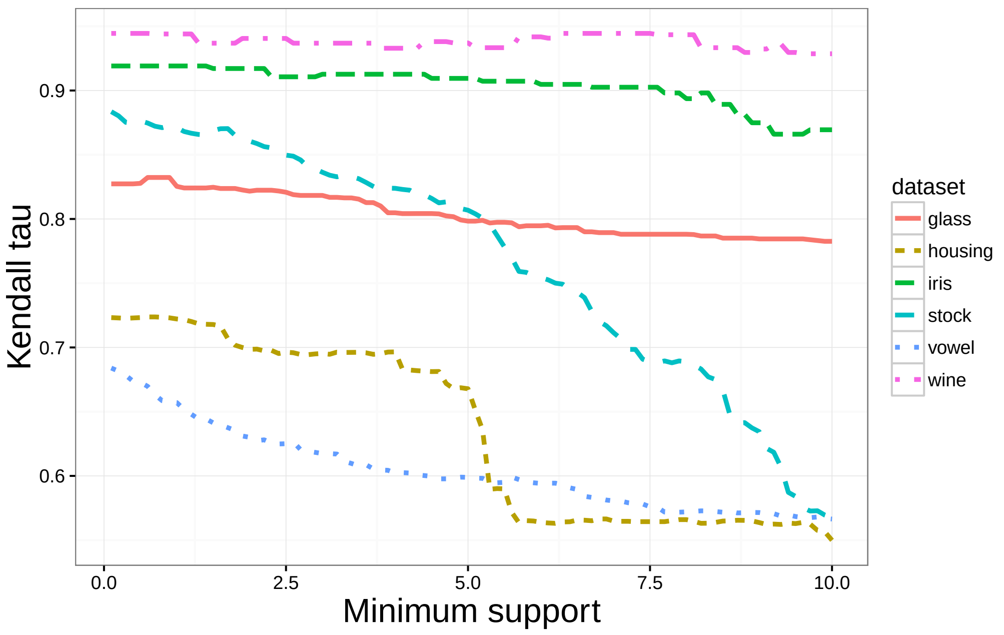


Figure 2.5: Average accuracy (Kendall τ) of CAREN as the *minsup* varies.

In general, as we increase *minsup* the accuracy decreases, which is a strong indicator that the support should be small (Figure 2.5). All lines are monotonically decreasing, i.e. either the values remain constant or they decrease as *minsup* increases.

From a different perspective, the changes are generally very small for $minsup \in [0.1\%, 1.0\%]$. Considering that lower *minsup* generate potentially more rules, we recommend $minsup = 1\%$ as a reasonable value to start experiments with.

Discretization techniques To test the influence of the discretization method used, we performed the same analysis using a non-supervised discretization method, *equal width*. In general, the accuracy had the same behavior, as a function of θ, as with *EDiRa*, i.e. the results are highly correlated (Figure 2.6). However, the supervised approach is consistently better. These results add further evidence that *EDiRa* is a suitable discretization method for label ranking [39].

Similar behavior was observed concerning the number of rules generated and the minimum confidence.

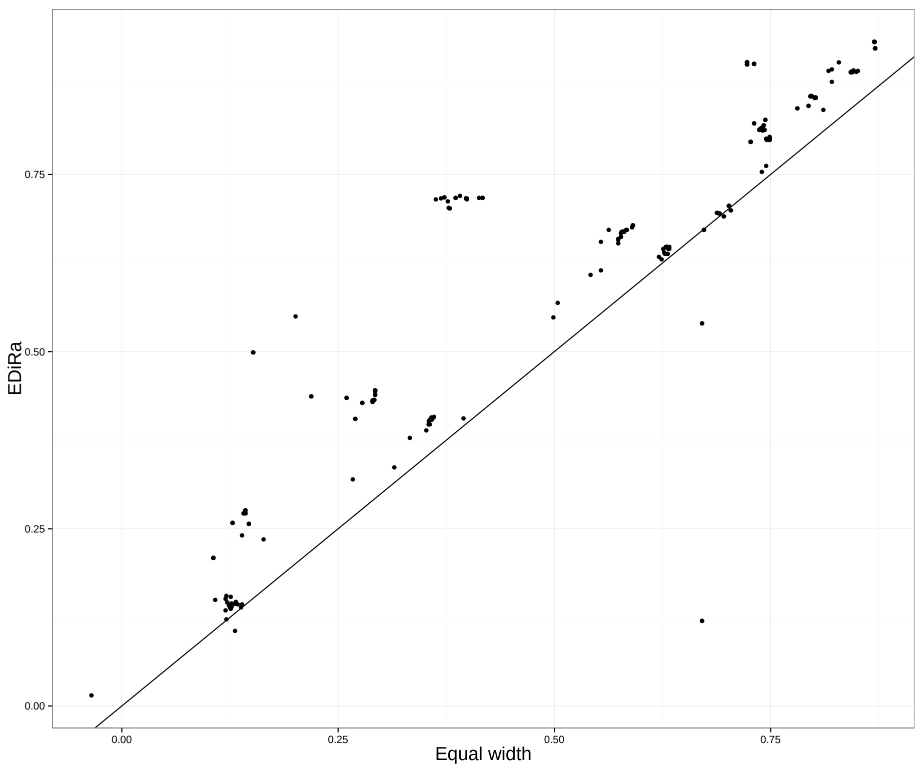


Figure 2.6: Ranking accuracy (Kendall τ) of CAREN after the discretization of data using *equal width* and *EDiRa*. This plot aggregates all the experiments carried out, concerning different issues, which means that each dataset is represented multiple times, with different parameter settings.

Summary It is well known that general, simple rules to set parameters of machine learning algorithms do not exist. Nevertheless it is good to know where reasonable values lie. Hence, we think that $\theta \in [0.5, 0.6]$ and *minsup* $= 1\%$ are good default values for LRAR with CAREN. In terms of the discretization methods, our results confirm that a supervised approach, such as *EDiRa*, is a good choice.

2.6.4 Results with PAR

In this work we use PAR, as a descriptive model, to find patterns concerning subsets of labels. We focus in the descriptive task for two reasons. One is to make the approach more simple and the other one is because this complements with the predictive LRAR approach.

The minimum support and confidence presented here are defining the abstention level of the model. The *minsup* and *minconf* were adjusted manually to generate a small set of rules between 150 to 200.

In the generation of PAR, we set the minimum lift to 1.5. Despite that many interesting rules were found, due to space limitations we only present the most relevant.

Algae data Using the Algae dataset, we found 179 PARs with $minsup = 2$ and $minconf = 90$. With $sup = 2.2\%$ and $conf = 100\%$ the rule with the highest lift (approx. 6) was:

$$\begin{aligned}&\texttt{Riversize} = small \wedge \texttt{pH} \geq 37.9 \wedge \texttt{Flowvelocity} = high \wedge\\&\texttt{Chloride} \geq 3.4 \wedge \texttt{Nitrates\&Ammonia} \geq 18.5\\&\rightarrow L6 \succ L2 \wedge L5 \succ L7 \wedge L2 \succ L7\end{aligned}$$

The consequent of this rule can be represented as $L6 \succ L2 \succ L7 \wedge L5 \succ L7$. Considering that the labels represent algae populations, this rule states that it is always true that, under these conditions, type 6 is more prevalent than type 2. It also states that type 7 is less prevalent than types 2, 5 and 6.

The second rule with highest lift, with $sup = 3.1\%$ and $conf = 91\%$ is:

$$\begin{aligned}&\texttt{Flowvelocity} = medium \wedge \texttt{Nitrates\&Ammonia} < 18.5 \wedge\\&\texttt{Nitrogenasnitrates} < 7.9\\&\rightarrow L1 \succ L7 \wedge L7 \succ L3\end{aligned}$$

The target of this rule is the partial ranking $L1 \succ L7 \succ L3$.

If this PAR was used for prediction, the subranking $\pi = (1, 0, 3, 0, 0, 0, 2)$ would have been the prediction.

Sushi data When analyzing the sushi dataset we got 166 rules with $minconf = 70\%$ and the $minsup = 1\%$. With a lift of 1.95 the following rule was found:

$$\begin{aligned}&\texttt{Ageinterval} = 15 - 19 \wedge \texttt{Sex} = Male \wedge \texttt{Livedin} = Eastern\ Japan\\&\rightarrow \texttt{egg} \succ \texttt{seaurchin} \wedge \texttt{shrimp} \succ \texttt{seaurchin}\end{aligned}$$

In the whole dataset, 37% of the people show this relative preferences $\texttt{egg} \succ \texttt{seaurchin} \wedge \texttt{shrimp} \succ \texttt{seaurchin}$. This PAR shows that this number almost double (72%), if we consider males from Eastern Japan, aged between $15-19$.

A related rule was also found concerning a different group of people, with different age and from a different region ($sup = 1.1\%$, $conf = 71.6\%$ and $lift = 1.65$):

$$\begin{aligned}
&\texttt{Ageinterval} = 30-39 \wedge \texttt{Sex} = \textit{Male} \wedge\\
&\texttt{Livesin} = \textit{Western Japan} \wedge \texttt{Changedcity} = \textit{Yes}\\
&\rightarrow \texttt{seaurchin} \succ \texttt{egg} \wedge\\
&\texttt{fattytuna} \succ \texttt{tunaroll} \wedge\\
&\texttt{tunaroll} \succ \texttt{cucumberroll} \wedge\\
&\texttt{fattytuna} \succ \texttt{egg}
\end{aligned}$$

This rule includes one relative preference found in this group, `seaurchin` $\succ$ `egg`, which is the opposite to what was observed in the previous rule. Based on this information, we analyzed the data and found out that 75% of people that live in Eastern Japan prefer `egg` to `seaurchin` while 84% of people from Western Japan prefer `seaurchin` to `egg`.

2.7 Conclusions

In this paper we address the problem of finding association patterns in label rankings. We present an extensive empirical analysis on the behavior of a label ranking method, the CAREN implementation of Label Ranking Association Rules. The performance was analyzed from different perspectives, *accuracy*, *number of rules* and *average confidence*. The results show that, similarity-based interest measures contribute positively to the accuracy of the model, in comparison to frequency-based approaches, i.e. when $\theta = 1$. The results confirm that LRAR are a viable label ranking tool which helps solving complex label ranking problems (i.e. problems with high ranking entropy). The results also enabled the identification of some values for the parameters of the algorithm that are good candidates to be used as default values.

Results also seem to indicate that, the higher the entropy, the more the accuracy can be affected by the choice of θ. An user can measure the ranking entropy of a dataset beforehand and adjust θ accordingly.

Additionally, we propose Preference Association Rules (PAR), which are association rules where the consequent represents multiple pairwise preferences.

We illustrated the usefulness of this approach to identify interesting patterns in label ranking datasets, which cannot be obtained with LRAR.

In future work, we will use PAR for predictive tasks.

Chapter 3

Entropy-based discretization methods for ranking data

Cláudio Rebelo de Sá, Carlos Soares, Arno Knobbe

in Information Sciences Journal, 2016

Abstract

Label Ranking problems are becoming increasingly important in Machine Learning. While there has been a significant amount of work on the development of learning algorithms for LR in recent years, there are not many pre-processing methods for LR. Some methods, like Naive Bayes for LR and APRIORI-LR, cannot handle real-valued data directly. Conventional discretization methods used in classification are not suitable for LR problems, due to the different target variable. In this work, we make an extensive analysis of the existing methods using simple approaches. We also propose a new method called EDiRa *for the discretization of ranking data. We illustrate the advantages of the method using synthetic data and also on several benchmark datasets. The results clearly indicate that the discretization is performing as expected and also improves the results and efficiency of the learning algorithms.*

3.1 Introduction

Research in Label Ranking (LR) has been increasing over the last few years [116, 36, 27, 28, 123, 127]. Label Ranking (LR) studies the problem of learning a mapping from instances to rankings over a finite number of predefined labels. An example of an LR problem is the ranking of a set of restaurants according to the preferences of a given person. It can be considered as a variant of the conventional classification problem [26]. However, in contrast to a classification setting, where the objective is to assign examples to a specific class, in LR we are interested in assigning a complete preference order of the labels to every example. An additional difference is that the true (possibly partial) ranking of the labels is available for the training examples.

As in any machine learning task, data preparation is essential for the development of accurate LR models. For instance, some algorithms are unable to deal with numeric variables, such as the basic versions of Naive Bayes and Association Rules [102, 4], in which case numeric variables should be discretized beforehand. Discretization, from a general point of view, is the process of partitioning a given interval into a set of discrete sub-intervals. It is normally used to split continuous intervals into two or more sub-intervals which can then be treated as nominal values. In theory, a good discretization should have a good balance between the loss of information and the number of partitions [90]. While there has been a significant amount of work on the development of learning algorithms for LR in recent years, there are not many pre-processing methods specifically for this task.

Discretization methods are typically organized in two groups, depending on whether or not they involve target variable information. These are usually referred to as *supervised* and *unsupervised* discretization, respectively. Previous research found that the supervised methods produce more useful discretizations than unsupervised methods [46]. The difference in nature between the target variable in classification and in LR problems implies that supervised discretization methods developed for the former are not suitable for the latter. In fact, in classification, two target values (i.e., classes) are either equal or different, while in LR, the difference between two rankings is closer to a continuous function, similar to the error in a regression setting. In this work, we make an extensive empirical analysis of the existing methods. We also propose a new method based on Minimum Description Length Principle (MDLP) [54] for the discretization of ranking data. The new method of supervised discretization for ranking data, which we refer to as EDiRa (Entropy-based Discretization for Ranking), follows the line of work in [40].

Based on MDLP for classification, it adapts the concept of entropy to LR based on the distance between rankings.

We also make an extensive study of the *Minimum Description Length Principle for Ranking data (MDLP-R)* method proposed in [40], which is also based on MDLP [54]. This analysis includes varying its parameter to assess how it affects the performance of the learner.

Finally we present a comparison between the newly proposed approach EDiRa and MDLP-R, along with the original MDLP (i.e. for classification). The results observed show that EDiRa behaves better in many scenarios and is also more robust.

The paper is organized as follows: Section 3.2 introduces the LR problem and the learning algorithms used in this paper. Section 3.3 introduces discretization and Section 3.4 describes the method proposed here. Section 3.5 presents the experimental setup and discusses the results. Finally, Section 3.6 concludes this paper.

3.2 Label Ranking

The LR task is similar to classification. In classification, given an instance x from the instance space $\mathbb{X}$, the goal is to predict the label (or class) λ to which x belongs, from a predefined set $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$. In LR, the goal is to predict the ranking of the labels in $\mathcal{L}$ that are associated with x [74]. A ranking can be represented as a total order over $\mathcal{L}$ defined on the permutation space Ω. In other words, a total order can be seen as a permutation π of the set $\{1, \ldots, k\}$, such that $\pi(a)$ is the position of λ_a in π.

As in classification, we do not assume the existence of a deterministic $\mathbb{X} \rightarrow \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω [26]. This means that, for each $x \in \mathbb{X}$, there exists a probability distribution $\mathcal{P}(\cdot|x)$ such that, for every $\pi \in \Omega$, $\mathcal{P}(\pi|x)$ is the probability that π is the ranking associated with x. The goal in LR is to learn the mapping $\mathbb{X} \rightarrow \Omega$. The training data contains a set of instances $D = \{\langle x_i, \pi_i \rangle\}, i = 1, \ldots, n$, where x_i is a vector containing the values $x_i^j, j = 1, \ldots, m$ of m independent variables describing instance i and π_i is the corresponding target ranking.

Given an instance x_i with label ranking π_i, and the ranking $\hat{\pi}_i$ predicted by an LR model, we evaluate the accuracy of the prediction with a loss function on Ω. One such function is the number of discordant label pairs,

$$\mathcal{D}(\pi, \hat{\pi}) = \#\{(a, b) | \pi(a) > \pi(b) \wedge \hat{\pi}(a) < \hat{\pi}(b)\}$$

If normalized to the interval $[-1, 1]$, this function is equivalent to Kendall's τ coefficient [85], which is a correlation measure where $\mathcal{D}(\pi, \pi) = 1$ and $\mathcal{D}(\pi, \pi^{-1}) = -1$ (π^{-1} denotes the inverse order of π).

The accuracy of a model can be estimated by averaging this function over a set of examples. This measure has been used for evaluation in recent LR studies [26, 40] and, thus, we will use it here as well. However, other correlation measures, like Spearman's rank correlation coefficient [118], can also be used.

Given the similarities between LR and classification, one could consider workarounds that treat the label ranking problem essentially as a classification problem. One such workaround is *Ranking As Class (RAC)* [40], which replaces the rankings with classes:

$$\forall \pi_i \in \Omega, \pi_i \rightarrow \lambda_i.$$

This approach allows the use of all pre-processing and prediction methods for classification in LR problems.

3.2.1 Association Rules for Label Ranking

Label Ranking Association Rules (LRAR) [36] are a straightforward adaptation of Class Association Rules (CAR):

$$A \rightarrow \pi$$

where $A \subseteq desc\,(\mathbb{X})$ and $\pi \in \Omega$. Where $desc\,(\mathbb{X})$ is the set of descriptors of instances in $\mathbb{X}$, typically pairs $\langle attribute, value \rangle$. Similar to how predictions are made with CARs in CBA (Classification Based on Associations) [97], when an example matches the antecedent of the rule, $A \rightarrow \pi$, the predicted ranking is π.

If the RAC approach is used, the number of classes can be extremely large, up to a maximum of $k!$, where k is the number of labels in $\mathcal{L}$. This means that the amount of data required to learn a reasonable mapping $\mathbb{X} \rightarrow \Omega$ can be very large.

Alternatively, mining of LRAR uses similarity-based support and confidence measures [36].

Similarity-based Support and Confidence

Given a measure of similarity $s(\pi_a, \pi_b)$, the *support* of the rule $A \rightarrow \pi$ is defined as follows:

$$sup_{lr}(A \rightarrow \pi) = \frac{\sum\limits_{i:A \subseteq desc(x_i)} s(\pi_i, \pi)}{n} \tag{3.1}$$

This essentially assigns a weight to each target π_i in the training data, that represents its contribution to the probability that π may be observed.

The similarity function is of the form:

$$s(\pi_a, \pi_b) = \begin{cases} s'(\pi_a, \pi_b) & \text{if } s'(\pi_a, \pi_b) \geq \theta_{sup} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where s' is itself a similarity function between rankings. Any function that measures ranking similarity, such as Kendall's τ or Spearman's ρ, can be used as s'. This general form assumes that below a given threshold, θ_{sup}, it is not useful to discriminate between different similarity values, as they are too different from π_a anyway.

The *confidence* of a rule $A \rightarrow \pi$ is obtained simply by replacing the measure of support with the new one:

$$conf_{lr}(A \rightarrow \pi) = \frac{sup_{lr}(A \rightarrow \pi)}{sup(A)}$$

In a similar way to $conf$ in classical Association Rule Mining [71], $conf_{lr}(A \rightarrow \pi)$, can be interpreted as the conditional probability of finding π given A, $\mathcal{P}(\pi|A)$.

As in [36], we use Kendall τ to measure the similarity between rankings in our experiments.

3.2.2 Naive Bayes for Label Ranking

Naive Bayes for Label Ranking (NBLR) [6] is an LR method based on the naive Bayes Classifier. It uses a measure of probability adapted for rankings, based on similar reasoning to the one underlying APRIORI-LR. This adapted probability measure is plugged directly into the naive Bayes algorithm to generate a LR model.

The prior probability of a ranking π is defined in [6] as the mean similarity between π and all the others:

$$\mathcal{P}_{LR}(\pi) = \frac{\sum_{i=1}^{n} \rho(\pi, \pi_i)}{n}$$

where ρ is the Spearman rank correlation coefficient [118]. This assumes that the larger the number of rankings similar to π there are, the higher the probability to observe π. Similarly, the conditional probability of the value i of attribute j, v_i^j given ranking π is defined as:

$$\mathcal{P}_{LR}(v_i^j|\pi) = \frac{\sum_{i:x_i^j=v_i^j} \rho(\pi, \pi_i)}{|\{i : x_i^j = v_i^j\}|}$$

Given an observation x_i, the Naive Bayes for LR outputs the ranking $\hat{\pi}$ with the highest $\mathcal{P}_{LR}(\pi|x_i)$ value:

$$\hat{\pi} = \underset{\pi \in \Pi_{\mathcal{L}}}{\arg\max}\, \mathcal{P}_{LR}(\pi|x_i)$$

where $\mathcal{P}_{LR}(\pi|x_i)$ is the estimated posterior probability of ranking π:

$$\mathcal{P}_{LR}(\pi|x_i) = \mathcal{P}_{LR}(\pi) \prod_{j=1}^{m} \mathcal{P}_{LR}(x_i^j|\pi)$$

3.3 Discretization

Discretization methods define intervals or ranges in continuous variables which allows them to be used as nominal variables by learning algorithms. Discretization is of great relevance since several algorithms can improve their performance by using discretized data [53], even those that can discretize variables on-the-fly [54], such as the ID3 discretizer [112].

The main issue in discretization is the choice of the intervals, because a continuous variable can be discretized in an infinite number of ways. An ideal discretization method finds a reasonable number of cut points that split the data into meaningful intervals. For classification datasets, a meaningful interval should be coherent with the class distribution along the variable.

Discretization approaches can be divided along several dimensions:

Top-down/Bottom-up Discretization methods with a Top-down or Bottom-up approach start by sorting the dataset with respect to the variable which will be discretized. In the Top-down approach, the method starts with an interval containing all points. Then, it recursively splits the intervals into sub-intervals, until a stopping criterion is satisfied. One example is MDLP method [54].

In the Bottom-up approach, the method starts with the maximal number of intervals (i.e., one cut point between each pair of adjacent values) and then iteratively merges them until a stopping criterion is satisfied. One well-known Bottom-up method is ChiMerge [86].

Static/Dynamic A dynamic discretization method acts on-the-fly, while the learner is building the model. Static methods discretize the data before the learning method starts to run. The latter are independent from the learning methods whereas the dynamic methods only have access to data as it is provided by the learner. Most of the discretization methods are static, such as ChiMerge [86] or MDLP [54]. An example of a dynamic method is how the ID3 algorithm deals with numeric variables [112].

Univariate/Multivariate Univariate methods, like MDLP [54], discretize one attribute at a time while multivariate ones, such as MVD [12] or SMDNS [76], can discretize two or more variables simultaneously. The latter can be useful when there are high levels of interaction between attributes [60].

Supervised/Unsupervised The discretization methods can use the values of the target variable, when available, or not. These options are referred to as *supervised* and *unsupervised* respectively. The *unsupervised* methods ignore the classes of the objects and divide the interval into a user-defined number of bins. Examples of the latter are the EqualWidth and EqualFrequency discretizations [60]. The *supervised* methods, like MDLP [54] or [73], take into account the distribution of the class labels in the discretization process. Previous research shows that the supervised methods tend to produce better discretizations than the unsupervised ones [46].

It is not an easy task to determine which discretization technique is the best because several criteria can be used to evaluate their performance [60]. These include direct measures, such as the number of intervals generated, the processing time and inconsistency [99], and indirect ones, such as measurement of the accuracy of classification algorithms on the discretized data. However, some tests have been done with the most well-known algorithms and the results indicate that ChiMerge [86], MDLP [54], Zeta [72], Distance [22], and Chi2 [99] are among the best ones [60]. Based on these results and on the fact that it is one of the most commonly used methods, we decided to adapt MDLP to discretize ranking data. A first adaptation of this method for LR was already introduced, named MDLP-R [40] (Section 3.4). However, as shown below, this method can be improved.

3.3.1 Entropy-based methods

Several methods perform discretization by optimizing entropy [29, 54]. In classification, class entropy is a measure of uncertainty in a finite interval of classes and it can be used as an evaluation metric.

The entropy of classes used in the original MDLP method [54], which derives from the Shannon entropy, is defined as:

$$Ent\left(S\right) = -\sum_{i=1}^{K} P\left(C_i, S\right) log\left(P\left(C_i, S\right)\right) \tag{3.3}$$

where $P\left(C_i, S\right)$ stands for the proportion of examples with class C_i in a subset S, and K is the number of distinct classes in S and

$$P\left(C_i, S\right) = \frac{\#C_i}{n_S}$$

where n_S is the number of instances in subset S.

A good partition is such that it minimizes the overall entropy in its subsets. Likewise, in discretization, a good partition of the continuous variable minimizes the class entropy in the subsets of examples it creates. It is well known that the optimal cut points must be between instances of distinct classes [54]. In practical terms, the class information entropy is calculated for all possible partitions and compared with the entropy without partitions. This can be done recursively until some stopping criterion is satisfied. The stopping criteria can be defined by a user or by a heuristic method like MDLP.

Minimum Description Length Principle MDLP [54] is a well-known method used to discretize continuous attributes in classification tasks. It measures the information gain of a given split point by comparing the values of entropy before and after the partition. For each split point considered, the entropy of the initial interval is compared with the weighted sum of the entropy of the two resulting intervals. Given an interval S:

$$Gain\left(A,T;S\right)=Ent\left(S\right)-\frac{|S_1|}{n_S}Ent\left(S_1\right)-\frac{|S_2|}{n_S}Ent\left(S_2\right)$$

where $|S_1|$ and $|S_2|$ are the number of instances on the left side (S_1) and the number of instances on the right side (S_2), respectively, of the cut point T in attribute A. The decision criterion for accepting or rejecting a new partition by MDLP is given by the Minimum Description Length Principle Cut (MDLPC) [54].

MDLPC Criterion The partition induced by a cut point T for a set S of n_S examples is accepted *iff*

$$Gain\left(A,T;S\right)>\frac{log_2\left(n_S-1\right)}{n_S}+\frac{\Delta\left(A,T;S\right)}{n_S}$$

where $\Delta\left(A,T;S\right)$ is equal to:

$$log_2\left(3^K-2\right)-\left[KEnt\left(S\right)-K_1Ent\left(S_1\right)-K_2Ent\left(S_2\right)\right]$$

and K, K_1, K_2 is the number of distinct target values in S, S_1, S_2 respectively.

3.4 Discretization for Label Ranking

A supervised discretization method for LR should take into account the specificities of its type of target, namely rankings. Two properties, in particular, are important: how many different rankings are present in the subset and how similar they are to each other. To adapt MLDP for LR, an entropy measure should be used that accounts for these two properties [40].

In this work, we compare two different adaptations of the Shannon entropy for rankings with the regular MDLP after an RAC transformation. These entropy measures use MDLPC as a stopping criterion, in the same way as it is used for classification. First we describe the adaptations of the entropy for rankings and then we show how to integrate it with MDLP.

Table 3.1: Example dataset D_{ex} - Small artificial dataset with noise in the rankings.

TID	x^1	π	λ^{RAC}
1	0.1	(1,2,4,3,5)	a
2	0.2	(1,2,3,4,5)	b
3	0.3	(2,1,3,4,5)	c
4	0.4	(1,3,2,4,5)	d
5	0.5	(1,2,3,5,4)	e
6	0.6	(5,4,3,1,2)	f
7	0.7	(4,5,3,2,1)	g
8	0.8	(5,3,4,2,1)	h

3.4.1 Adapting the concept of entropy for rankings

In this section, we explain how the adapted versions of entropy for LR can be used. We start by a motivation of the approach with a discussion of the use of the concept of entropy in LR. We then show in detail how the heuristic adaptation of entropy for rankings behaves.

Let us consider a very simple synthetic dataset D_{ex}, presented in Table 3.1. In this dataset, we have eight distinct rankings in the target column π. Even though they are all distinct, the first five rankings are very similar (the label ranks are mostly in ascending order), the last three are also very similar to each other (descending order), but the first group is very different from the second. Without any further considerations, it is natural to assume that an optimal split point for x^1 should lie between values 0.5 and 0.6 (instances 5 and 6).

In the RAC approach, the rankings are transformed into eight distinct classes as shown in column λ^{RAC}. The natural split point identified earlier is completely undetectable in column λ^{RAC}. As shown in Equation 3.3, the entropy of a set of classes depends on the relative proportion of a class. If we measure the ranking proportion in the same way, we get:

$$P\left(\lambda_i^{RAC}, D_{ex}\right) = 1/8, \forall \lambda_i^{RAC} \in D_{ex}$$

This example illustrates why the concept of entropy cannot be applied directly to rankings. In fact, the problem we are facing here is the same as in the adaptation of the concept of support for LRAR in APRIORI-LR [36] (Equation 3.4). Hence, a similar line of reasoning as the one in Section 3.2.1 can be followed here. The uncertainty associated with a certain ranking decreases in the presence of similar – although not equal – rankings. Furthermore, this decrease is proportional to that distance. To take this into account, we can use the distance-based ranking proportion of ranking π_i in set S [40]:

$$P_\pi(\pi_i, S) = \frac{\sum_{j=1}^{n_S} s(\pi_i, \pi_j)}{\sum_{i=1}^{K} \sum_{j=1}^{n_S} s(\pi_i, \pi_j)} \tag{3.4}$$

where

$$s(\pi_a, \pi_b) = \begin{cases} s'(\pi_a, \pi_b) & \text{if } s'(\pi_a, \pi_b) \geq \theta_{disc} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

and θ_{disc} is the threshold parameter of the similarity measure, equivalent to the threshold for similarity support, θ_{sup}, in Equation 3.2. As in [40], we use Kendall τ as s', by default, and the negative correlations are ignored (Section 3.2.1), i.e. $\theta_{disc} \geq 0$.

However, this approach alone is not enough to give a fair measure for the entropy of rankings. The entropy of the set of classes $\{\lambda_1, \lambda_2\}$ is the same as $\{\lambda_1, \lambda_3\}$ or $\{\lambda_2, \lambda_3\}$. This happens because, λ_1 is as different from λ_2 as λ_2 is from λ_3. However, in LR, the difference between two rankings is closer to a continuous function. Considering these two sets:

$$1)\mathcal{S}_1 = \{\pi_1 = (1,2,3,4,5), \pi_2 = (1,2,3,5,4)\}$$

$$2)\mathcal{S}_2 = \{\pi_1 = (1,2,3,4,5), \pi_3 = (5,4,3,2,1)\}$$

the distance-based ranking proportion of π_1 relative to sets S_1 and S_2, using Kendall τ as a similarity measure, for S_1 and S_2 is, respectively:

$$P_\pi(\pi_1, \mathcal{S}_1) = \frac{s(\pi_1, \pi_1) + s(\pi_1, \pi_2)}{s(\pi_1, \pi_1) + s(\pi_1, \pi_2) + s(\pi_2, \pi_1) + s(\pi_2, \pi_2)} = \\ = \frac{1 + 0.8}{1 + 0.8 + 0.8 + 1} = 0.5 \tag{3.6}$$

and

$$P_\pi(\pi_1, \mathcal{S}_2) = \frac{s(\pi_1, \pi_1) + s(\pi_1, \pi_3)}{s(\pi_1, \pi_1) + s(\pi_1, \pi_3) + s(\pi_3, \pi_1) + s(\pi_3, \pi_3)} = \\ = \frac{1 + 0}{1 + 0 + 0 + 1} = 0.5. \tag{3.7}$$

Since the ranking proportions will be the same in both cases, the entropy will also be the same. If we decompose these rankings into pairwise-comparisons,

Table 3.2: Pairwise-comparisons perspective.

Pairwise	π_1(1,2,3,4,5)	π_2(1,2,3,5,4)	π_3(5,4,3,2,1)
$\lambda_1 \succ \lambda_2$	true	true	false
$\lambda_1 \succ \lambda_3$	true	true	false
$\lambda_1 \succ \lambda_4$	true	true	false
$\lambda_1 \succ \lambda_5$	true	true	false
$\lambda_2 \succ \lambda_3$	true	true	false
$\lambda_2 \succ \lambda_4$	true	true	false
$\lambda_2 \succ \lambda_5$	true	true	false
$\lambda_3 \succ \lambda_4$	true	true	false
$\lambda_3 \succ \lambda_5$	true	true	false
$\lambda_4 \succ \lambda_5$	true	false	false

we obtain the 10 label comparisons presented in Table 3.2. π_1 matches 9 pairs with π_2, but it does not match any with π_3.

Another issue that must be taken into account when adapting entropy for rankings is that, as in any probabilistic phenomenon, ranking data is expected to contain some noise. Noise in rankings may be caused by different reasons. For example, if a total ranking results from the combination of a set of incomplete pairwise preferences, it may not be an entirely accurate representation of the true ranking. Or, give a set of items (e.g. products) associated with an utility function (e.g. price), when asked to rank those items according to the utility function, different experts might provide slightly different rankings. The differences can arise due to imperfect or incomplete access to information [92]. As an example, instances 6, 7 and 8 in D_{ex} could represent the same "real" ranking, say $(5, 4, 3, 2, 1)$, but perceived by different experts. Additionally, as the number of labels increases, we expect that the probability of being affected by noise is also higher. For simplicity, in this work, we assume all different sources of noise have similar manifestations and, thus, are treated in the same way.

Considering that entropy is a measure of disorder, we believe that a measure of entropy for rankings should generate lower values for sets with similar rankings (low noise) and higher values for sets with different rankings (high noise).

MDLP-R

As discussed in [40], MDLP-R addresses the issues discussed earlier. It is based on an adaptation of entropy for rankings Ent_{LR}, which is defined as:

$$Ent_{LR}(S) = \sum_{i=1}^{K} P_{\pi}(\pi_i, S)\, log\,(P_{\pi}(\pi_i, S)) \times log\,(Q(\pi_i, S)) \tag{3.8}$$

where K is the number of distinct rankings in S and $Q(\pi_i, S)$ is the average similarity of the ranking π_i with the rankings in the subset S:

$$Q(\pi_i, S) = \frac{\sum_{j=1}^{n_S} s(\pi_i, \pi_j)}{n_S}$$

where s is a similarity measure (Equation 3.5). Additionally, $abs\,(log\,(Q(\pi_i, S)))$ can be seen as a dispersion measure around π_i. If a lot of rankings in S are similar to π_i, $Q(\pi_i, S)$ will be close to 1. On the other hand, low values are obtained when there are no rankings in S that are similar to π_i. As a practical example, using this measure on the sets mentioned before, $\mathcal{S}_1$ and $\mathcal{S}_2$, we get, $Q(\pi_1, \mathcal{S}_1) = 0.90$ and $Q(\pi_1, \mathcal{S}_2) = 0.50$. Which will result in $abs\,(log\,(0.90)) = 0.105$ and $abs\,(log\,(0.5)) = 0.693$, as expected.

The value of Ent_{LR} depends strongly on the threshold θ_{disc}. If the similarity measure, $s(\pi_i, \pi_j)$, generates negative correlations, $Q(\pi_i, S)$ may, under certain conditions, be negative. Since the *log* of negative values is not defined, the domain of the parameter θ_{disc} is defined as: $0 \leq \theta_{disc} \leq 1$. Alternatively, this limitation is not required if the value of s is rescaled to the interval $[0, 1]$ (Equation 3.5).

In our example, the value of Ent_{LR} on the sets $\mathcal{S}_1$ and $\mathcal{S}_2$ is $Ent_{LR}(\mathcal{S}_1) \approx 0.073$ and $Ent_{LR}(\mathcal{S}_2) \approx 0.480$, respectively. Intuitively, this makes more sense than the values obtained using MDLP with RAC, which is the same for the two sets $Ent_{RAC}(\mathcal{S}_1) = Ent_{RAC}(\mathcal{S}_2) \approx 0.693$.

EDiRa

Here, we propose a new entropy measure for rankings which is more simple and intuitive than MDLP-R and has no parameters. This new measure can be divided into two parts. The first part is the Shannon entropy as defined in [54] (Equation 3.3). The second part is a dispersion measure which makes the entropy measure more sensitive to overall similarity between the rankings in the set S. It is expressed as an average of the similarity measure, s', normalized between 0 and 1.

While, in MDLP-R, the proportion in entropy is similarity-based, the new measure uses the standard proportion as in classification, $P(\pi_i, S)$. In fact, some tests indicated that, in this particular approach, the two types of proportions yielded equivalent results. [1] This means that the second part of the formula has a stronger impact on the similarity level. For this reason we decided to keep the simplest approach, both from a theoretical and a computational perspective, which is the standard proportion.

In the second part of the expression, which represents the homogeneity of the rankings in the subset S, we use $log\left(\overline{kt}(S)\right)$. Where $\overline{kt}(S)$ is the average normalized [2] Kendall τ distance in the subset S:

$$\overline{kt}(S) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{n_S}\frac{\tau(\pi_i,\pi_j)+1}{2}}{K \times n_S}$$

As an example, $\overline{kt}(\mathcal{S}_1) = 0.95$ and $\overline{kt}(\mathcal{S}_2) = 0.50$

This leads to the new expression to compute the entropy of rankings:

$$Ent_{LR2}(S) = \sum_{i=1}^{K} P(\pi_i, S)\, log\left(P(\pi_i, S)\right) log\left(\overline{kt}(S)\right) \tag{3.9}$$

where K is the number of distinct rankings in S. This measure makes the discretization method more robust to noise, as shown in Section 3.5. The values of this new measure on the example sets $\mathcal{S}_1$ and $\mathcal{S}_2$ are $Ent_{LR2}(\mathcal{S}_1) \approx 0.036$ and $Ent_{LR2}(\mathcal{S}_2) \approx 0.480$. These values show that this new entropy is consistent with the previous one.

Given that Kendall τ is a measure of the proportion of the concordant pairs of labels, this entropy measure can still work with partial orders, as long as there is at least one pairwise comparison per instance. However, in this work, we focus on total orders.

Two different discretization methods can be created simply by replacing the standard entropy measure by each of the two new measures in the entropy of rankings in the MDLP presented in [54]. In terms of taxonomy (Section 3.3), MDLP-R and EDiRa are, thus, in the same category as MDLP, namely *Top-down*, *Static*, *Univariate* and *Supervised*.

[1] In the interest of space, we opted to omit these results.

[2] Since similarity measures for rankings, such as Kendall τ and Spearman ρ, are defined in the interval $[-1, 1]$, we rescale their to the interval $[0, 1]$ by adding 1 and dividing by 2.

3.5 Experimental Results

In this paper, we are investigating discretization methods, which are hard to evaluate directly. Thus, they are evaluated here as pre-processing methods to the APRIOR-LR [36] and NBLR [6] algorithms. The experimental study is divided into three parts. In the first part, we perform experiments on benchmark datasets to gain some understanding about how the parameter θ_{disc} affects the performance of MDLP-R. The second part consists of experiments on controlled artificial datasets to investigate whether the methods are performing as expected. The third part tests the discretization methods with the APRIORI-LR algorithm and NBLR on datasets from the KEBI Data Repository [26] (Table 3.3).

For these experiments in particular, it is useful to define a simple measure of the diversity of the target rankings, which we refer to as *Unique Ranking's Proportion*, U_π. U_π is the proportion of distinct target rankings for a given dataset (Table 3.3). As a practical example, the *iris* dataset has 5 distinct rankings for 150 instances, which will result in a $U_\pi = \frac{5}{150} \approx 3\%$. This means that all the 150 rankings are duplicates of these 5.

We believe that datasets with high U_π should be more difficult to discretize using the RAC approach because the number of classes is very high. The experiments performed in the artificial datasets (Section 3.5.2) provide evidence that support this observation.

Table 3.3: Summary of the datasets.

Datasets	type	#examples	#labels	#attributes	U_π
bodyfat	B	252	7	7	94%
calhousing	B	20,640	4	4	0.1%
cpu-small	B	8,192	5	6	1%
elevators	B	16,599	9	9	1%
fried	B	40,769	5	9	0.3%
glass	A	214	6	9	14%
housing	B	506	6	6	22%
iris	A	150	3	4	3%
segment	A	2310	7	18	6%
stock	B	950	5	5	5%
vehicle	A	846	4	18	2%
vowel	A	528	11	10	56%
wine	A	178	3	13	3%
wisconsin	B	194	16	16	100%

The evaluation measure is Kendall's τ and the performance of the methods was estimated using ten-fold cross-validation. In Section 3.5.2 the experiments were repeated ten times, due to the random nature of the changes made to the data. For the generation of Label Ranking Association Rules (LRAR), we used an extension of CAREN [10] for LR.

3.5.1 Sensitivity to the θ_{disc} parameter

The entropy of a set of rankings varies depending on the value of the θ_{disc} threshold (Equation 3.8), sometimes significantly affecting its value. The first set of experiments investigates how that affects the accuracy of the learning methods. We did experiments with APRIORI-LR on KEBI datasets for different θ_{disc} thresholds, varying θ_{disc} from 0 to 1 by steps of 0.1.

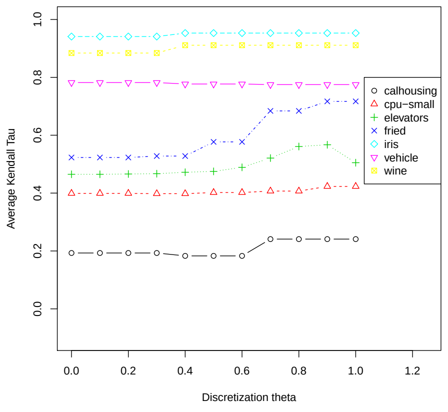


Figure 3.1: Accuracy of APRIORI-LR (expressed in terms of Kendall τ) as θ_{disc} varies, in datasets where the distinct rankings represent less than 5% of the data. ($U_\pi < 5\%$)

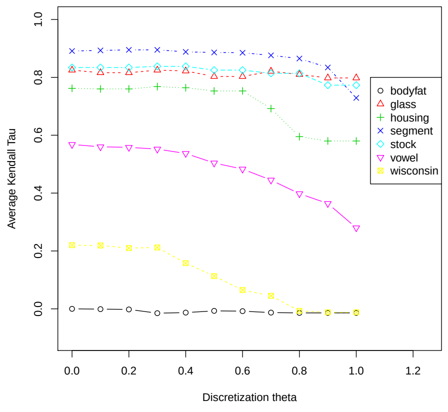


Figure 3.2: Accuracy of APRIORI-LR (expressed in terms of Kendall τ) as θ_{disc} varies, in datasets where the distinct rankings represent more than 5% of the data. ($U_\pi \geq 5\%$)

The results indicate that θ_{disc} plays an important role in the behavior of MDLP-R. Which, on the other hand, will influence the accuracy of APRIORI-LR. To better understand how to adjust θ_{disc} for any given dataset, it is useful to divide the datasets into two distinct groups: 1) $U_\pi < 5\%$ (Figure 3.1) and 2) $U_\pi \geq 5\%$ (Figure 3.2).

The most interesting impact of splitting the datasets by high and low U_π is that they seem to behave differently. For the first group (Figure 3.1) as θ_{disc}

increases, the accuracy of APRIORI-LR increases for most of the datasets and very rarely decreases the accuracy. On the other hand, in Figure 3.2 we can see that increasing θ_{disc} has the opposite effect on the second group. This means that when there are only a few distinct rankings in the data, the method can be less sensitive to the ranking similarities. As the value of the parameter increases, the method tends to fit every distinct ranking into a different bin. This should work as long as there is a reasonable small number of distinct rankings. A lower θ_{disc} threshold allows the method to group larger ranges into each bin. In datasets with higher U_π, as the method is more robust to noise, it should create better partitions, i.e. by grouping the "closer" rankings together in the same bins.

This analysis shows that θ_{disc} plays an important role in the effectiveness of the partitions made by MDLP-R. Also, by measuring U_π, we can get some clues about a reasonable value for θ_{disc}.

3.5.2 Results on Artificial Datasets

Table 3.4: Discretization results using the MDLP, MDLP-R and EDiRa methods.

			Partitions	
TID	x^1	π	MDLP-R/EDiRa	MDLP
1	0.1	(1,2,4,3,5)	1	1
2	0.2	(1,2,3,4,5)	1	2
3	0.3	(2,1,3,4,5)	1	3
4	0.4	(1,3,2,4,5)	1	4
5	0.5	(1,2,3,5,4)	1	5
6	0.6	(5,4,3,1,2)	2	6
7	0.7	(4,5,3,2,1)	2	7
8	0.8	(5,3,4,2,1)	2	8

Results obtained with artificial datasets can give more insight about how the discretization methods perform. Table 3.4 compares the intervals discretized by the MDLP-R and MDLP on the very simple dataset presented in Table 3.1. As expected, since there are eight distinct rankings, the RAC approach with MDLP for classification will see eight distinct classes and break the dataset into eight intervals. MDLP-R and EDiRa, however, can identify the similarities of rankings, and split the dataset into two intervals.

For a more thorough analysis, we follow the experimental setup used in [40] with some variations. The synthetic datasets are based on a simple set with

100 examples, containing one independent variable with value 1 for the first example, 2 for the second, and so on, and the target rankings are distributed in the following order:

- Examples 1 to 38: variations of $\pi_1 = (10, 2, 3, 4, 5, 6, 7, 8, 9, 1)$
- Examples 39 to 75: variations of $\pi_2 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$
- Examples 76 to 100: variations of $\pi_3 = (10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$

The natural breakpoints for this dataset are $T_1 = 38.5$ and $T_2 = 75.5$ which were intentionally chosen to avoid trivial partitions. However, considering that π_1 is much closer to π_2 than to π_3, T_2 should have a bigger impact in the total entropy than T_1. In order to test the advantages of our method in comparison with the RAC approach, we intentionally introduced noise in the target rankings, by performing several swaps. Each swap is an inversion of two consecutive ranks in every ranking of the data. For each ranking, the choice of the pairs to invert is random. Swaps will be done repeatedly, to obtain different levels of noise.

We performed experiments which vary the number of swaps from 0 to 150. As the number of random swaps increases, the proportion of unique rankings U_π should also increase. Therefore it becomes harder to learn an accurate model. In the following experiments, U_π grows very rapidly, as any number of swaps bigger than 5 produce a $U_\pi \geq 99\%$.

In [40], *minconf* was fixed to 50% in all APRIORI-LR runs and *minsup* = 0.1%. When APRIORI-LR cannot find at least one LRAR to rank a new instance it predicts a default ranking. Here, we use a different approach. As the default rule is only used as a last resort, for a fair comparison of the methods, the minimum confidence (*minconf*) is adjusted with a simple greedy method (Algorithm 2), so that $Cov \geq 95\%$, where Cov is the proportion of test examples covered by the model, i.e. with a prediction not generated by the default rule. For each run, a different *minconf* can be found, so the values presented in Table 3.5 are the average of the 10 runs.

Figure 3.3 (top graph) shows the effect of varying the number of swaps on the ranking accuracy obtained by APRIORI-LR with the three different discretization methods, MDLP, MDLP-R and EDiRa. The graph, clearly indicates that the discretization with EDiRa (orange line) leads to better results for APRIORI-LR, than with the other two. While for lower number of swaps, the difference is not so evident, as the noise increases, the other methods are increasingly more affected by it than EDiRa.

The two ranking discretization methods, MDLP-R and EDiRa, behave simi-

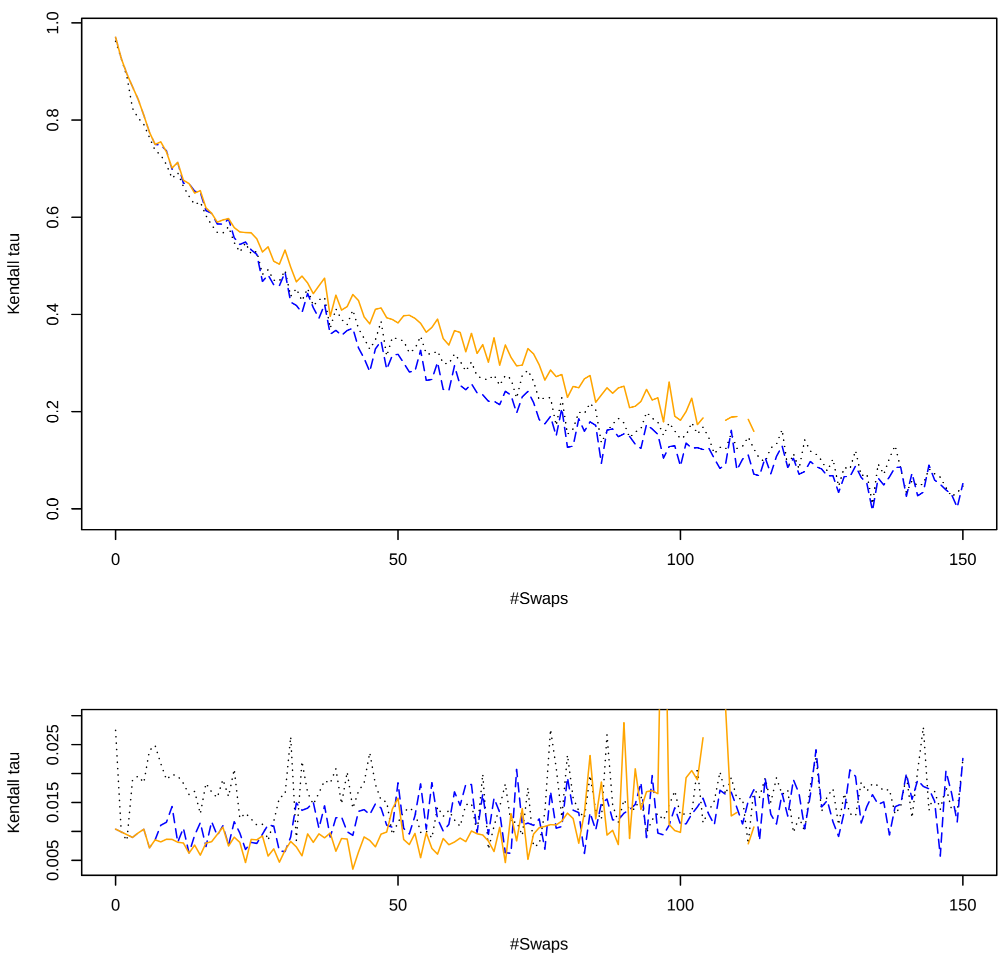


Figure 3.3: Accuracy (Top) and its Standard Deviation (Bottom) of the APRIORI-LR (expressed in terms of Kendall τ) as a function of the number of swaps and its standard deviation, for MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line).

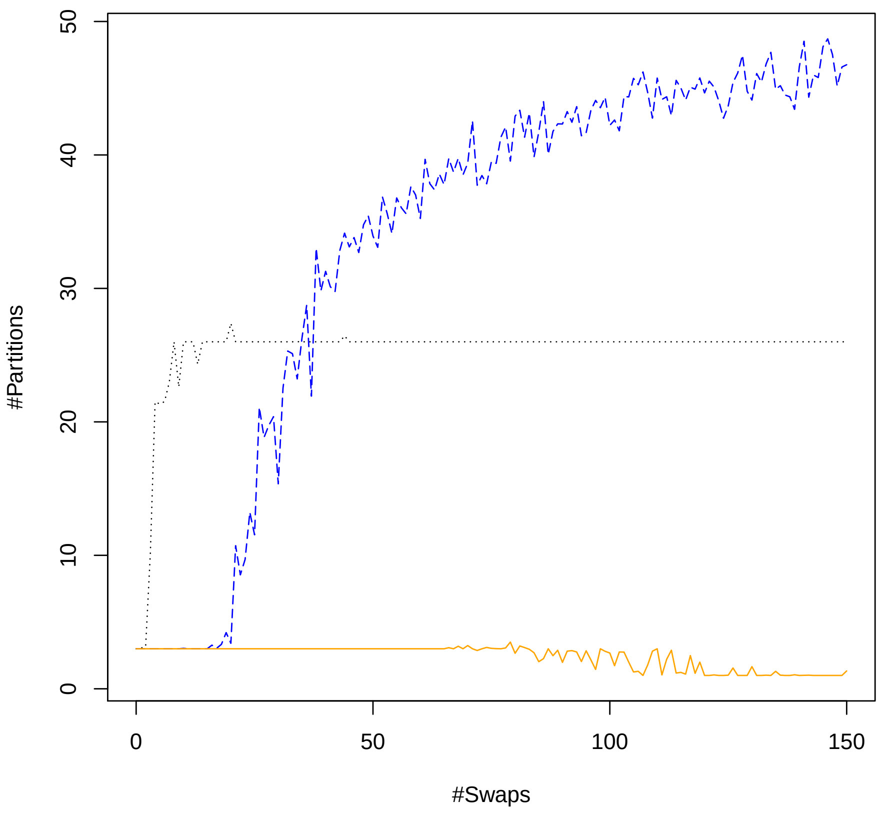


Figure 3.4: Comparison of the average number of partitions generated by MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line).

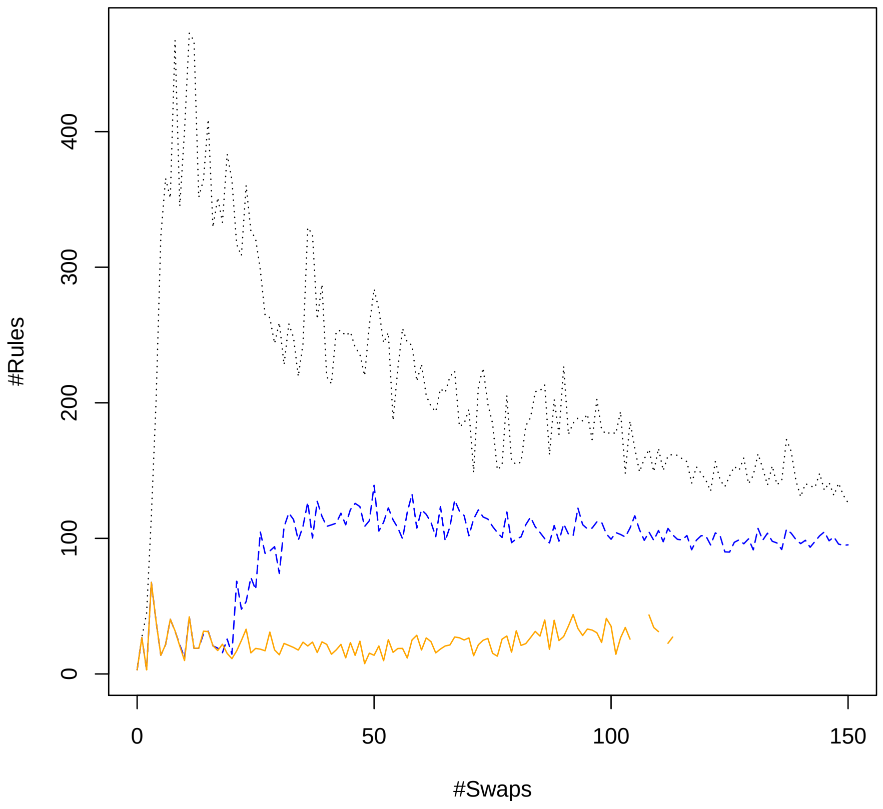


Figure 3.5: Comparison of the number of rules generated by APRIORI-LR after discretization with MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line).

Algorithm 2 Parameter tuning method.

```
$minconf \leftarrow 100\%$
$minsup \leftarrow 1\%$
$Cov \leftarrow 0\%$
**while** $Cov < 95\%$ **do**
    **function** APRIORI-LR($minconf, minsup$)
        **return** $Cov$
    **end function**
    **if** $Cov < 95\%$ **then**
        $mconf \leftarrow mconf - 5$
    **end if**
**end while**
**return** $minconf$
```

larly between 0 to 20 swaps, with equivalent accuracies obtained by APRIORI-LR (as it can be seen in the top graph in Figure 3.3 by the overlapping lines). However, from that point on, MDLP-R starts to behave worst, in terms of APRIORI-LR accuracy, than EDiRa and even MDLP.

If we analyze Figure 3.3 (bottom graph) representing the standard deviation over the 10 repetitions of the results presented in the top graph, there is additional information in favor of MDLP-R and EDiRa. The standard deviation of the results of these methods is smaller in the presence of small amounts of noise (until approximately 30 swaps for MDLP-R and 80 swaps for EDiRa). This means that EDiRa is the most reliable method in this scenario.

One great advantage of using EDiRa can be seen in Figure 3.4, which represents the number of partitions made by the methods for different values of the number of swaps. For any number of swaps up to 80, approximately, EDiRa makes two partitions, which means that the split point choice is invariant to greater amounts of noise than MDLP-R and MDLP. This will result in a smaller number of rules generated by APRIORI-LR, as supported by the graph in Figure 3.5. In this scenario, EDiRa makes APRIORI-LR much more efficient because it will use less than 10% of the rules, relatively to MDLP, and even gets slightly better accuracy. Furthermore, fewer rules means that the model is easier to interpret by humans, which is an important requirement in many applications [94].

In Figure 3.6, we can see how the average *minconf* (determined by Algorithm 2) evolves as the number of swaps increase. Intuitively, we expect that fewer partitions will produce rules with lower confidence as we increase the

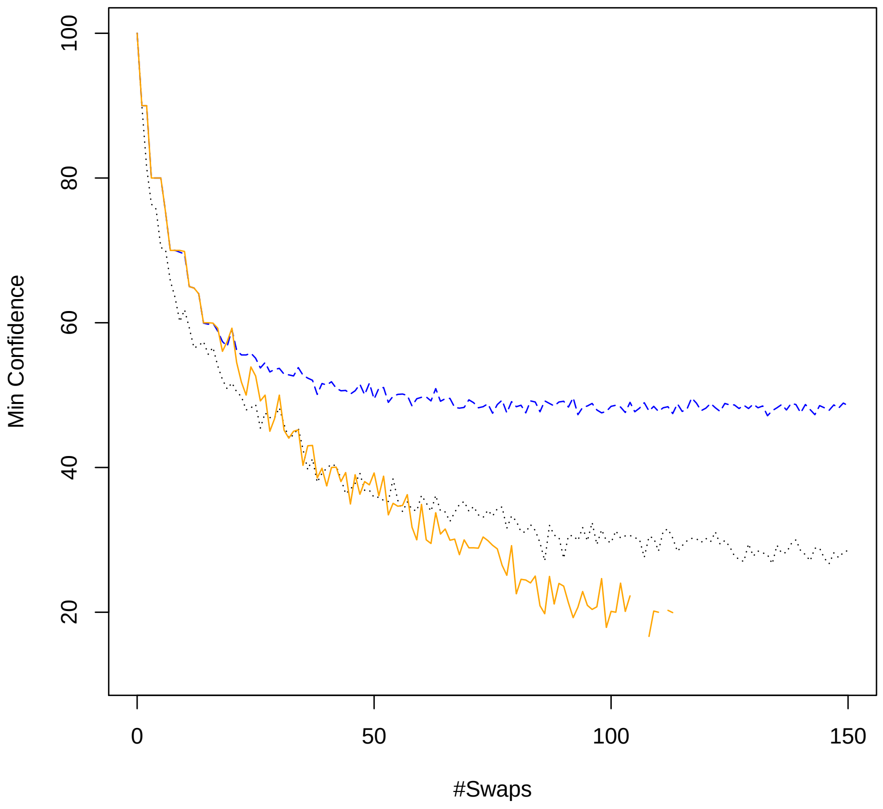


Figure 3.6: Comparison of the average *minconf* used by APRIORI-LR with the three discretization methods. MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line).

noise. These values are in agreement with the observed graphs, in particular with Figure 3.4.

Remember that we aim to decrease the entropy of the system by making partitions. However, as the level of noise increases the relationship between the independent variable and the target becomes weaker and, at some point, random. Ideally, a supervised discretization method should be able to detect this phenomenon. This is exactly what we observe in Figure 3.4 for EDiRa, when the number of swaps is greater than 100, which in a ranking of 10 labels will probably lead to a random target ranking, it stops making partitions. On the other hand, from around 20 swaps, MDLP-R starts to make more and more partitions, resulting in worst accuracy for APRIORI-LR.

Finally, in Figure 3.7 we are able to see how the different methods measure entropy for the same data. It is interesting to realize that, in the graph of 0 swaps, the methods behave similarly. As we increase the number of swaps, we start to see how the behaviors diverge. For 6 swaps or more MDLP cannot identify any obvious partitions, which explains the flat line of the method in Figure 3.4. On the other hand, for MDLP-R and EDiRa the partitions are still very clear up to 50 swaps.

Taking into account that the two rankings used to generate the target rankings for examples 1 to 75 are more similar to each other than to the ranking used in the remaining examples, it makes more sense to observe lower values near T_2 than near T_1. The graphs for 1 to 10 swaps in Figure 3.7 show that MDLP-R is giving very similar values for this two cut points, while EDiRa clearly indicates that T_2 is the most important cut.

In previous work, [40], MDLP-R was performing better than MDLP in most of the results but this is not observed in Figure 3.3. This is due to the use in these experiments of a different setup and different parameters from the ones used in [40]. The tuning of *minconf* leads to an increase of accuracy for APRIORI-LR with an MDLP discretization, which outperforms the results obtained with MDLP-R. We believe that the main reason is that from a certain level of noise (more than 20 swaps) MDLP-R is overfitting. This observation is supported by Figure 3.4, where the number of partitions grows up to almost 50 partitions for very noisy scenarios. This number of partitions represents almost %50 of the number of instances in these experiments. Still, for smaller levels of noise, MDLP-R is a better choice in comparison to MDLP, as the accuracy of APRIORI-LR is higher, even though, using fewer rules Figure 3.5.

All of these results are good indicators that EDiRa creates more meaningful

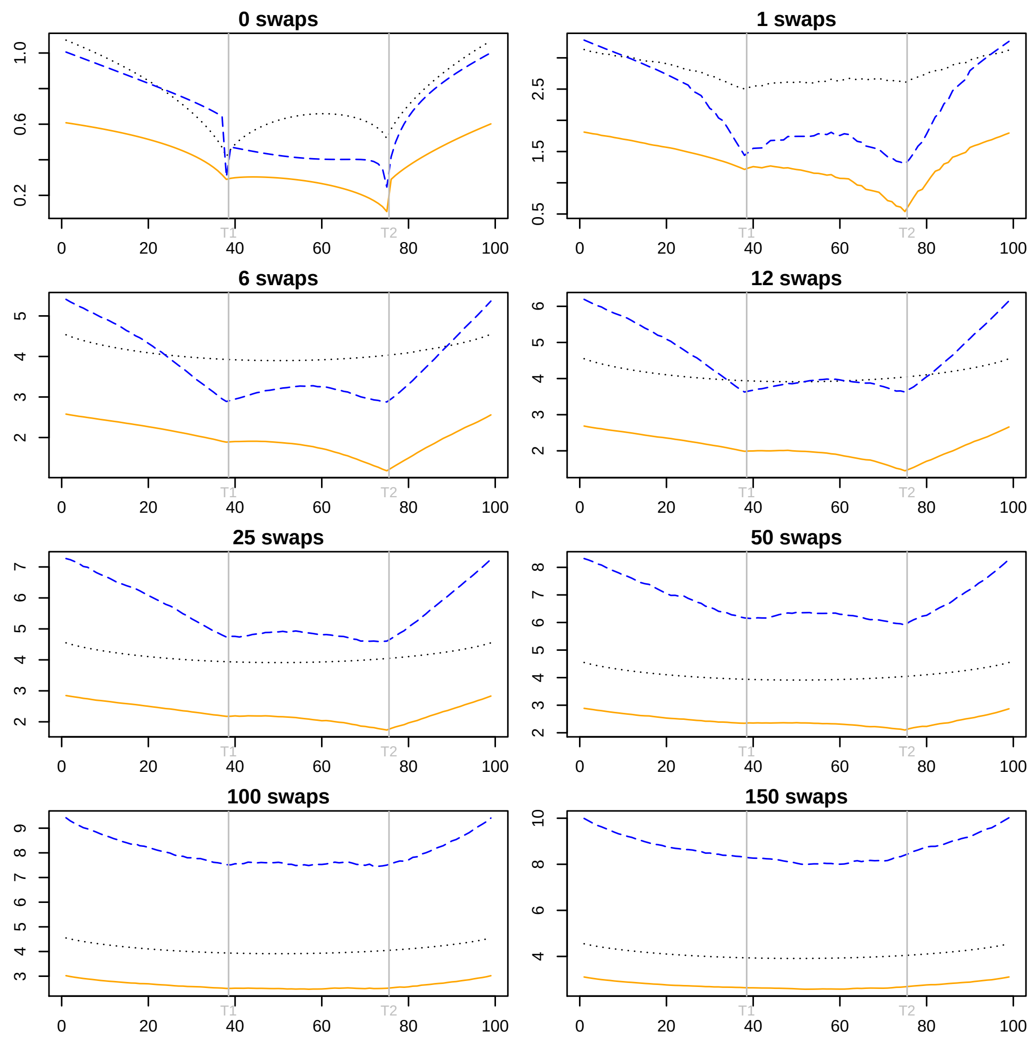


Figure 3.7: Comparison of the entropy values for MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line) for each candidate cut point for different number of swaps. Where the vertical axis is the entropy and the horizontal axis the candidate cut points.

intervals for ranking data.

3.5.3 Results on Benchmark Datasets

In this section, we describe experiments carried out with two algorithms which are more suitable for nominal rather than continuous variables, APRIORI-LR [36] and NBLR [6]. Finally, we also briefly discuss how the methods behave using different similarity measures.

Results with APRIORI-LR

In these experiments, we used a similar experimental setup to the one in [40]. Given that the majority of the datasets have less than 1000 instances and we want to avoid overfitting, the minimum support (*minsup*) was set to 1% instead of 0.1%. For the *minconf*, we use the method proposed in Section 3.5.2 (Algorithm 2).

Table 3.5: Results obtained by APRIORI-LR with MDLP, MDLP-R and EDiRa discretization on benchmark datasets. The mean accuracy is represented in terms of Kendall's tau, τ.

	MDLP				MDLP-R				EDiRa			
	τ	mconf	#rules	#part	τ	mconf	#rules	#part	τ	mconf	#rules	#part
bodyfat	.087	28	748	59	.000	5	744	80	.139	24	144	2
calhousing	.291	35	113	7	.193	26	89	106	.272	35	107	9
cpu-small	.414	37	209	3	.399	38	302	37	.429	35	332	4
elevators	.646	60	206	3	.465	45	678	116	.669	60	714	4
fried	.749	35	1,733	6	.523	24	1,019	20	.706	25	1,281	12
glass	.815	90	52	3	.825	99	510	10	.800	87	43	2
housing	.720	57	373	10	.762	66	465	22	.715	56	210	5
iris	.944	93	24	3	.941	85	31	4	.906	83	31	3
segment	.891	90	3,415	13	.891	85	1,887	36	.895	90	3467	7
stock	.868	81	324	11	.834	78	340	19	.858	80	315	8
vehicle	.827	94	4,506	4	.782	87	2,282	14	.812	94	4,664	4
vowel	.668	74	4,013	14	.568	59	1,881	112	.648	63	794	3
wine	.937	100	617	2	.884	100	1,549	6	.937	100	1,028	2
wisconsin	.268	41	1,058	44	.220	38	1149	76	.404	52	10,550	2
average τ	.651	-	-	-	.591	-	-	-	.656	-	-	-
standard dev τ	.276	-	-	-	.301	-	-	-	.251	-	-	-

Table 3.5 shows that EDiRa improves the APRIORI-LR accuracy in the benchmark datasets when compared to MDLP-R. With this new method, the average number of partitions (#part) is drastically reduced in all datasets.

Comparing EDiRa with MDLP, we observe that both methods lead to very similar accuracy. We would like to give particular attention to the two datasets where U_π is very big namely *bodyfat* and *wisconsin*. The datasets have $U_\pi = 94\%$ and $U_\pi = 100\%$ respectively. The average number of partitions with EDiRa in these two datasets is much smaller than with MDLP while the accuracy of APRIORI-LR has a major increase.

Another important fact that can be observed in Table 3.5 is that every time that APRIORI-LR generated more rules with EDiRa than with MDLP-R, there is an increase in the accuracy. This is true for datasets *calhousing*, *cpu-small*, *elevators*, *fried*, *segment*, *vehicle* and *wisconsin*.

Results with NBLR

The second LR algorithm tested was an adaptation of the simple naive Bayes algorithm for Label Ranking [6]. This adaptation of the algorithm cannot be used with numeric variables.

Table 3.6: Results obtained for naive Bayes for Label Ranking with MDLP, MDLP-R and EDiRa discretization on benchmark datasets. (The mean accuracy is represented in terms of Kendall's tau, τ).

	MDLP		MDLP-R		EDiRa	
	τ	#part	τ	#part	τ	#part
bodyfat	.060	59	.081	80	.175	2
calhousing	.293	7	.322	106	.286	9
cpu-small	.397	3	.408	37	.400	4
elevators	.611	3	.580	116	.602	4
fried	.823	6	.897	20	.896	12
glass	.759	3	.675	10	.717	2
housing	.742	10	.777	22	.684	5
iris	.889	3	.876	4	.836	3
segment	.711	13	.712	36	.702	7
stock	.736	11	.742	19	.719	8
vehicle	.657	4	.682	14	.657	4
vowel	.686	14	.497	112	.616	3
wine	.786	2	.794	6	.786	2
wisconsin	.346	44	.268	76	.394	2
average τ	.607	-	.593	-	.605	-
standard dev τ	.240	-	.245	-	.213	-

Table 3.6 shows the accuracy of this algorithm for the benchmark datasets. In this case, there is no clear winner among any of the three discretization methods available. However, the results obtained with EDiRa seem to be

more consistent as the standard deviation of the accuracy shows. This indicates that EDiRa is more reliable than the other methods.

Additionally, if we observe the two datasets with the highest U_π, which are expected to be the hardest for the methods, the best accuracy is obtained with EDiRa.

Using a different similarity measure

As mentioned in Section 3.4.1, any ranking similarity measure can be used for MDLP-R or EDiRa. Similar results to the ones presented in Table 3.5 and Table 3.6, were obtained using Spearman ρ as similarity measure. In the interest of space, we do not present those results. However, to illustrate them, we present in Figure 3.8 how the accuracy obtained by APRIORI-LR follows the same behavior for both discretization methods using the two similarity measures.

3.6 Conclusions

In this paper, we presented an extensive study of discretization for LR problems. Despite the increase in research on LR, most papers focus on the development of new algorithms and, thus, little attention has been paid to pre-processing methods. We carried out a detailed analysis on MDLP-R. We also introduced a new method for supervised discretization in LR problems, EDiRa, based on an improved measure of entropy. Both methods use different entropy measures which were adapted to take into account the similarity of rankings.

An analysis of MDLP-R in terms of the similarity threshold parameter θ_{disc} was performed to better understand its behavior. It was clear that, in simple scenarios, MDLP-R deals with noisy ranking data according to expectation and that θ_{disc} plays a major role in it. However, in more complex situations, MDLP-R tends to overfit the data.

The new method, EDiRa, was motivated by the need for increased sensitivity to the homogeneity of rankings in a set. The results show that EDiRa is a viable LR discretization method which clearly outperforms MDLP-R.

We believe that the measure of entropy for rankings proposed here, despite its heuristic nature, makes sense and may be more generally useful in LR. This

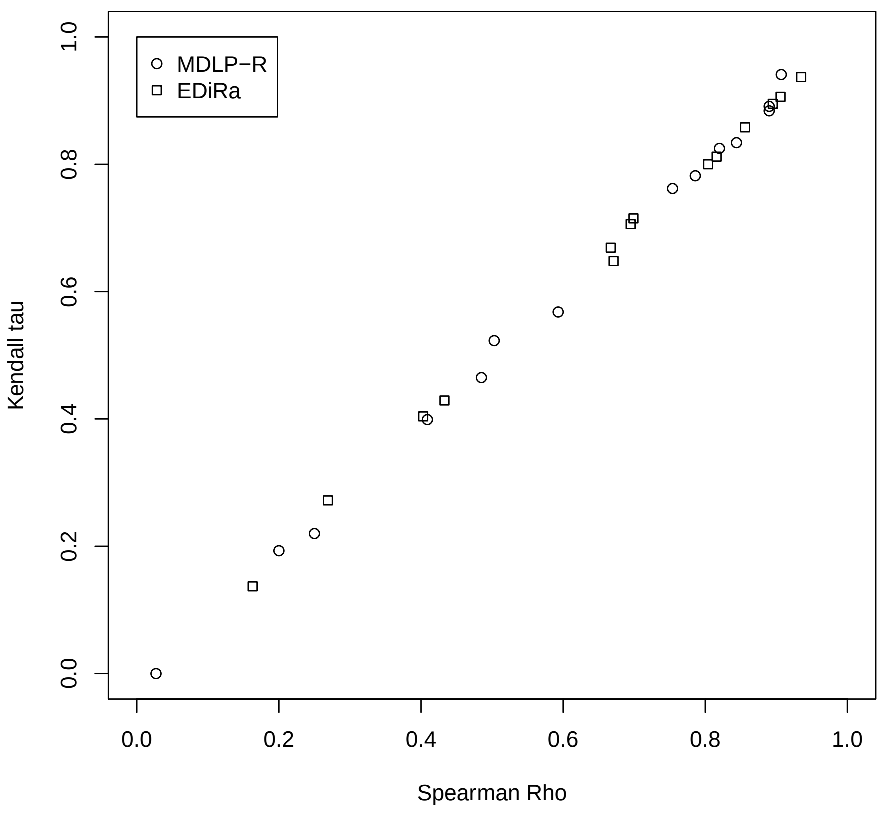


Figure 3.8: Comparison of the accuracy (in terms of Kendall τ) of APRIORI-LR in the datasets from Table 3.3. The data was discretized with MDLP-R (circles) and EDiRa (squares), using Kendall (vertical-axis) and Spearman (horizontal-axis) similarity measures as s'.

new measure and EDiRa bring new possibilities for processing ranking data and can motivate the creation of new methods for LR learning that cannot deal with continuous data. Furthermore, even though it was developed in the context of the LR task, it can be also applied to other fields such as regression since it is based on a distance measure such as Kendall τ.

We also investigated the robustness of the methods to the measure of ranking similarity used. We compared two different measures, observing that the results are very similar.

Empirical tests were carried out on benchmark problems from the KEBI repository. These datasets are adapted from UCI classification problems. Although they can be used for the development of methods, such as in this and many other LR papers, it is essential for the field that the methods are tested on real LR problems like meta-learning or predicting the rankings of financial analysts [6].

Acknowledgments

This research was partially supported by Fundo Europeu de Desenvolvimento Regional (FEDER), and also Projects "NORTE-07-0124-FEDER-000059" and "NORTE-07-0124-FEDER-000057", financed by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT).

We would like to thank our colleague Artur Aiguzhinov for making his method (Naive Bayes for Label Ranking) available for the experiments.

Chapter 4

Label Ranking Forests

Cláudio Rebelo de Sá, Carlos Soares, Arno Knobbe, Paulo Cortez

in Expert Systems Journal, 2016

Abstract

The problem of Label Ranking is receiving increasing attention from several research communities. The algorithms that have been developed/adapted to treat rankings of a fixed set of labels as the target object, include several different types of decision trees (DT). One DT-based algorithm, which has been very successful in other tasks but which has not been adapted for label ranking is the Random Forests (RF) algorithm. RFs are an ensemble learning method that combines different trees obtained using different randomization techniques. In this work, we propose an ensemble of decision trees for Label Ranking, based on Random Forests, which we refer to as Label Ranking Forests (LRF). Two different algorithms that learn DT for label ranking are used to obtain the trees. We then compare and discuss the results of LRF with standalone decision tree approaches. The results indicate that the method is highly competitive.

4.1 Introduction

Label Ranking (LR) is an increasingly popular topic in the machine learning literature [116, 36, 27, 28, 123]. LR studies a problem of learning a mapping from instances to rankings over a finite number of predefined labels. It can be considered a natural generalization of the conventional classification problem, where the goal is to predict a single label instead of a ranking of all the labels [26].

Some application of Label Ranking approaches are [74]: Meta-learning [16], where we try to predict a ranking of a set of algorithms according to the best expected accuracy on a given dataset; Microarray analysis [74] to find patterns in genes from Yeast on five different micro-array experiments (spo, heat, dtt, cold and diau); Image categorization [58] of landscape pictures from several categories (beach, sunset, field, fall foliage, mountain, urban).

There are two main approaches to the problem of LR: methods that transform the ranking problem into multiple binary problems and methods that were developed or adapted to treat the rankings as target objects, without any transformation. An example of the former is the ranking by pairwise comparisons [74]. Examples of algorithms that were adapted to deal with rankings as the target objects include decision trees [120, 26], naive Bayes [6] and k-Nearest Neighbor [17, 26]. Some of the latter adaptations are based on statistical distribution of rankings (e.g., [24]) while others are based on ranking distance measures (e.g., [120, 36]).

Tree-based models have been used in classification [111], regression [20] and also label ranking [120, 26, 35] tasks. These methods are popular for a number of reasons, including how they can clearly express information about the problem, because their structure is relatively easy to interpret even for people without a background in learning algorithms.

In classification, combining the predictive power of an ensemble of trees often comes with significant accuracy improvements [19]. One of the earliest examples of ensemble methods is *bagging* (a contraction of bootstrap-aggregating) [18]. In bagging, an ensemble of trees is generated and each one is learned on a random selection of examples from the training set. A popular ensemble method is Random Forests [19] which combines different randomization techniques.

Considering the success of Random Forests in terms of improved accuracy for classification and regression problems [13], some approaches have been proposed to deal with different targets, such as *bipartite rankings* [30]. Label

Ranking Forests should also be seen as a potential robust approach for LR. Adapting RF to Label Ranking can be a straightforward process once you have adapted decision trees.

In this work, we propose an approach of ensemble learners which we refer to as Label Ranking Forests (LRF). The proposed method is a natural adaptation of Random Forests for LR, combining the task-independent RF algorithm with the traditional algorithm for top-down induction of decision trees adapted for label ranking. The available adaptations of decision tree algorithms for LR include Label Ranking Trees (LRT) [26], Ranking Trees [115] and Entropy-based Ranking Trees [35]. Considering that the set of trees, in most cases, predict distinct rankings, one should also take into account ranking aggregation methods.

This paper extends previous work [35], in which we proposed a new version of decision trees for LR, called the Entropy-based Ranking Trees and empirically compared them to existing approaches. The main contribution in this paper is the new Label Ranking Forests algorithm, which is an adaptation of the RF ensemble method, using Entropy-based Ranking Trees as the base level algorithm. The results indicate that LRF are competitive with state of the art methods and improve the accuracy of standalone decision trees. An additional contribution is an extension of the original experimental study on Entropy-based Ranking Trees, by analyzing model complexity.

4.2 Label Ranking

In this section, we start by formalizing the problem of label ranking (Section 4.2.1) and then we discuss the adaptation of the decision trees algorithm for label ranking (Section 4.2.2) and one such adaptation, Entropy Ranking Trees (Section 4.2.3).

4.2.1 Formalization

The Label Ranking (LR) task is similar to classification. In classification, given an instance x from the instance space $\mathbb{X}$, the goal is to predict the label (or class) λ to which x belongs, from a predefined set $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$. In LR, the goal is to predict the ranking of the labels in $\mathcal{L}$ that is associated with x [74]. A ranking can be represented as a total order over $\mathcal{L}$ defined on the permutation space Ω. A total order can be seen as a permutation π of the set $\{1, \ldots, k\}$, such that $\pi(a)$ is the position of λ_a in π.

As in classification, we do not assume the existence of a deterministic $\mathbb{X} \to \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω [26]. This means that, for each $x \in \mathbb{X}$, there exists a probability distribution $\mathcal{P}(\cdot|x)$ such that, for every $\pi \in \Omega$, $\mathcal{P}(\pi|x)$ is the probability that π is the ranking associated with x. The goal in LR is to learn the mapping $\mathbb{X} \to \Omega$. The training data is a set of instances $D = \{\langle x_i, \pi_i \rangle\}, i = 1, \ldots, n$, where x_i is a vector containing the values $x_i^j, j = 1, \ldots, m$ of m independent variables describing instance i and π_i is the corresponding target ranking.

Given an instance x_i with label ranking π_i, and the ranking $\hat{\pi}_i$ predicted by an LR model, we can evaluate the accuracy of the prediction with loss functions on Ω. Some of these measures are based in the number of discordant label pairs:

$$\mathcal{D}(\pi, \hat{\pi}) = \#\{(a, b) | \pi(a) > \pi(b) \wedge \hat{\pi}(a) < \hat{\pi}(b)\}$$

If normalized to the interval $[-1, 1]$, this function is equivalent to Kendall's τ coefficient, which is a correlation measure where $\mathcal{D}(\pi, \pi) = 1$ and $\mathcal{D}(\pi, \pi^{-1}) = -1$, where π^{-1} denotes the inverse order of π (e.g. $\pi = (1, 2, 3, 4)$ and $\pi^{-1} = (4, 3, 2, 1)$).

The accuracy of a model can be estimated by averaging this coefficient over a set of examples. Other correlation measures, such as Spearman's rank correlation coefficient [118], have also been used [17]. Although we assume total orders, it may be the case that two labels are tied in the same rank (i.e. $\pi_i(a) = \pi_i(b), a \neq b$). In this case, a variation of Kendall's τ, the $tau - b$ [5] can be used.

4.2.2 Ranking Trees

Tree-based models have been used in classification [111], regression [20], and label ranking [120, 26, 35] tasks. These methods are popular for a number of reasons, including how they can clearly express information about the problem, because their structure is relatively easy to interpret even for people without a background in learning algorithms. It is also possible to obtain information about the importance of the various attributes for the prediction depending on how close to the root they are used.

The Top-Down Induction of Decision Trees (TDIDT) algorithm is commonly used for induction of decision trees [102]. It is a recursive partitioning algorithm that iteratively splits data into smaller subsets which are increasingly more homogeneous in terms of the target variable (Algorithm 3). A split is a test on one of the attributes that divides the dataset into two disjoint subsets. For instance, given a numerical attribute x^2, a split could be $x^2 \geq 5$. Given a splitting criterion that represents the gain in purity obtained with a split, the algorithm chooses the split that optimizes its value in each iteration. In its simplest form, the TDIDT algorithm only stops when the nodes are pure, i.e., when the value of the target attribute is the same for all examples in the node. This usually causes the algorithm to overfit, i.e., to generate models that capture the noise in the data, as well as the regularities that are of general usefulness. One approach to address this problem is to introduce a stopping criterion in the algorithm that tests whether the best split is significantly improving the quality of the model. If not, the algorithm stops and returns a leaf node. The algorithm is executed recursively for the subsets of the data obtained based on the best split until the stopping criterion is met. A leaf node is represented by a value of the target attribute generated by a rule that solves potential conflicts in the set of training examples that are in the node. That value is the prediction that will be made for new examples that fall into that node. In classification, the prediction rule is usually the most frequent class among the training examples.

Algorithm 3 TDIDT algorithm

```
Input: Dataset $D$
BestSplit = Test of the attributes that optimizes the SPLITTING CRITERION
if STOPPING CRITERION == TRUE then
    Determine leaf prediction based on the target values in $D$
    Return a leaf node with the corresponding LEAF PREDICTION
else
    LeftSubtree = TDIDT($D_{\neg BestSplit}$)
    RightSubtree = TDIDT($D_{BestSplit}$)
end if
```

The adaptation of this algorithm for label ranking involves an appropriate choice of the splitting criterion, stopping criterion and the prediction rule (Algorithm 3).

Splitting Criterion The splitting criterion is a measure that quantifies the quality of a given partition of the data. It is usually applied to all the

Table 4.1: Illustration of the splitting criterion

Attribute	Condition=true		Condition=false	
	values	rank corr.	values	rank corr.
x^1	a	0.3	$\{b,c\}$	-0.2
	b	0.2	$\{a,c\}$	0.1
	c	0.5	$\{a,b\}$	0.2
x^2	< 5	-0.1	≥ 5	0.1

possible splits of the data that can be made with tests on the values of individual attributes.

In Ranking Trees (RT) the goal is to obtain leaf nodes that contain examples with target rankings as similar between themselves as possible. To assess the similarity between the rankings of a set of training examples, the mean correlation between them is calculated using Kendall, Spearman or any other ranking correlation coefficient. The quality of the split is given by the weighted mean correlation of the values obtained for the subsets, where the weight is given by the number of examples in each subset.

For simplicity, if we ignore the weights, the splitting criterion of ranking trees is illustrated both for nominal and numerical attributes in Table 4.1. The nominal attribute x^1 has three values (a, b and c). Therefore, three binary splits are possible. For the numerical attribute x^2, a split can be made in between every pair of consecutive values. In this case, the best split is $x^1 = c$, with a mean correlation of 0.5, in comparison to a mean correlation of 0.2 for the remaining, i.e., the training examples for which $x^1 = \{a, b\}$.

Stopping Criterion The stopping criterion is used to determine if it is worthwhile to make a split or if there is a significant risk of overfitting [102]. A split should only be made if the similarity between examples in the subsets increases substantially. Let $\mathcal{S}_{parent}$ be the similarity between the examples in the parent node and $\mathcal{S}_{split}$ the weighted mean similarity in the subsets obtained with the best split. The stopping criterion is defined as follows [115]:

$$(1 + \mathcal{S}_{parent}) \geq \gamma(1 + \mathcal{S}_{split}) \tag{4.1}$$

Note that the relevance of the increase in similarity is controlled by the γ parameter. A $\gamma \geq 1$ does not ensure increased purity of child nodes. On the other hand, small γ values require splits with very large increase in purity, which means that the algorithm will stop the recursion early.

Table 4.2: Illustration of the prediction rule

	λ_1	λ_2	λ_3	λ_4
π_1	1	3	2	4
π_2	2	1	4	3
$\overline{\pi}$	1.5	2	3	3.5
$\hat{\pi}$	1	2	3	4

Prediction Rule The prediction rule is a method to generate a prediction from the (possibly conflicting) target values of the training examples in a leaf node. In LR, the aggregation of rankings is not so straightforward as in other tasks (e.g. classification or regression) and is known as the *ranking aggregation problem* [126]. It is a classical problem in social choice literature [31] but also in information retrieval tasks [51]. A *consensus ranking* minimizes the distance to all rankings [84]. A simple approach, which we adopted in this work, is to compute the average ranking [17] of the predictions. It is calculated by averaging the rank for each label λ_j, $\overline{\pi}(j) = \sum_i \pi_i(j)/n$. The predicted ranking $\hat{\pi}$ is the ranking $\overline{\pi}$ of the labels λ_j obtained based on the average ranks $\overline{\pi}(j)$. Table 4.2 illustrates the prediction rule used in this work.

4.2.3 Entropy Ranking Trees

Recently, we proposed an alternative approach to decision trees for ranking data, the Entropy-based Ranking Trees (ERT) [35]. ERT uses an adaptation of Information Gain (IG) [39] to assess the splitting points and the Minimum Description Length Principle Cut (MDLPC) [54] as the stopping criterion. To explain this method, we start by presenting the IG for rankings measure and then the adapted splitting and stopping criteria.

Decision trees for classification, such as ID3 [111], use Information Gain (IG) as a splitting criterion to determine the best split points. IG is a statistical property that measures the gain in entropy, between the prior and actual state [102]. In this case, we measure it in terms of the distribution of the target variable, before and after the split. In other words, considering a set S of size n_S, since entropy, H, is a measure of disorder, IG is basically how much uncertainty in S is eliminated after splitting on a numerical attribute x^a:

$$IG\left(x^a, T; S\right) = H\left(S\right) - \frac{|S_1|}{n_S} H\left(S_1\right) - \frac{|S_2|}{n_S} H\left(S_2\right)$$

where $|S_1|$ and $|S_2|$ are the number of instances on the left side (S_1) and the number of instances on the right side (S_2), respectively, of the cut point T in attribute x^a.

In cases where S is a set of rankings, we can use the entropy for rankings [39] which is defined as:

$$H_{ranking}\left(S\right)=\sum_{i=1}^{K}P\left(\pi_i,S\right)log\left(P\left(\pi_i,S\right)\right)log\left(\overline{kt}\left(S\right)\right) \tag{4.2}$$

where $P\left(\pi_i,S\right)$ is the proportion of rankings equal to π_i in S, K is the number of distinct rankings in S and $\overline{kt}\left(S\right)$ is the average normalized Kendall τ [85] distance in the subset S:

$$\overline{kt}\left(S\right)=\frac{\sum_{i=1}^{K}\sum_{j=1}^{n}\frac{\tau(\pi_i,\pi_j)+1}{2}}{K\times n_S}.$$

As in Section 4.2.2, the leaves of the tree should not be forced to be pure. Instead, a stopping criterion should be used to avoid overfitting and be robust to noise in rankings. Given an entropy measure, the adaptation of the splitting and stopping criteria comes in a natural way. As shown in [39], the *MDLPC Criterion* can be used as a splitting criterion with the adapted version of entropy $H_{ranking}$. This entropy measure also works with partial orders, however, in this work, we only use total orders.

4.3 Random Forests

Random Forests (RF) [19] are an ensemble method originally proposed for classification and regression problems. It essentially consists of the generation of multiple decision trees obtained using different randomization techniques. The set of predictions made by each of these trees is then aggregated to obtain the prediction of the ensemble.

The RF algorithm is related to another popular ensemble method by the same author, Bagging [18], which stands for *bootstrap-aggregating*. This is an ensemble method that takes a predefined number s of samples (without replacement) from the training data to construct s models. Given a new example, s predictions are generated, which are then aggregated, usually with average or mode, to obtain a combined prediction.

RF can be regarded as an extension of bagging. Given a forest size s and a training dataset D, a set of bootstrap samples, $\{D'_1 \ldots, D'_s\}$ is generated by sampling with repetition from D. A decision tree is learned from each $\{D'_1 \ldots, D'_s\}$, which is grown in a slightly different way from the original. At each node, only a random subset of the m features can be used for splitting. In classification, the number of random features used in each split is usually $\sqrt{m}$ and in regression $\log_2 m$. This results in what is usually referred to as *random trees*. As in bagging, each of the s random trees makes predictions on the test data, which are then combined using a suitable aggregation method.

One of the reasons for the popularity of RF lays in the fact that they have few parameters to tune and can be applied to various tasks [117]. They require a simple implementation and even with small sample sizes it usually gives accurate results. Moreover, considering that it uses s independent learners, it can be parallelized.

One of the reasons that makes RF a popular approach is that it is possible to take advantage of the algorithm to assess variable importance [61].

4.3.1 Label Ranking Forests

Considering the success of Random Forests in terms of improved accuracy for classification and regression problems, some approaches have been proposed to deal with different targets, such as *bipartite rankings* [30]. Label Ranking Forests should also be seen as a potential robust approach for LR. Adapting RF to Label Ranking can be a straightforward process once you have adapted decision trees.

Thus, we propose a new ensemble LR algorithm, the Label Ranking Forests based on Random Forests. With this approach, we expect to increase the accuracy of Label Ranking tree methods.

In classification and regression, the aggregation of predictions is done in a simple way, mode and mean, respectively. However, as discussed in Section 4.2.2, the aggregation of rankings is not so straightforward. Like in Ranking Trees, we use the average ranking [17] to aggregate the predictions.

Given the similarity of the LR task to classification, the number of random subset features we use in each split is $\sqrt{m}$, the same value that is used in RF for classification.

When the algorithm is not able to find a good split on any of the $\sqrt{m}$ selected features for the root node, it looks for a split on all the m features instead. This prevents the random feature selection mechanism from generating empty trees.

4.4 Empirical Study

In this section we describe the empirical study to investigate the performance of LRF and the tree methods used at the base level. We start by describing the experimental setup (Section 4.4.1), then the results of the base-level algorithms (Section 4.4.2) and finally the results of the new algorithm (Section 4.4.3).

4.4.1 Experimental setup

The experiments are carried out on datasets from the KEBI Data Repository at the Philipps University of Marburg [26] that are typically used in LR research (Table 4.3). They are based on classification and regression datasets, obtained using two different transformation methods: A) the target ranking is a permutation of the classes of the original target attribute, derived from the probabilities generated by a Naive Bayes classifier; B) the target ranking is derived for each example from the order of the values of a set of numerical variables, which are then no longer used as independent variables. A few basic statistics of the datasets used in our experiments are presented in Table 4.3. Although these are somewhat artificial datasets, they are quite useful as benchmarks for LR algorithms.

A simple measure of the diversity of the target rankings is the *Unique Ranking's Proportion*, U_π. U_π is the proportion of distinct target rankings for a given dataset (Table 4.3). As a practical example, the *iris* dataset has 5 distinct rankings for 150 instances, which yields $U_\pi = \frac{5}{150} \approx 3\%$. This means that all the 150 rankings are duplicates of these 5.

The code for all the experiments presented in this paper has been written in R [113].[1]

The generalization performance of the LR methods was estimated using a methodology that has been used previously for this purpose [74]. The eval-

[1]The code is available at `https://github.com/rebelosa/labelrankingforests`.

Table 4.3: Summary of the KEBI datasets

Datasets	type	#examples	#labels	#attributes	U_π
autorship	A	841	4	70	2%
bodyfat	B	252	7	7	94%
calhousing	B	20,640	4	4	0.1%
cpu-small	B	8,192	5	6	1%
elevators	B	16,599	9	9	1%
fried	B	40,769	5	9	0.3%
glass	A	214	6	9	14%
housing	B	506	6	6	22%
iris	A	150	3	4	3%
pendigits	A	10,992	10	16	19%
segment	A	2310	7	18	6%
stock	B	950	5	5	5%
vehicle	A	846	4	18	2%
vowel	A	528	11	10	56%
wine	A	178	3	13	3%
wisconsin	B	194	16	16	100%

uation measure is Kendall's τ and the performance of the methods was estimated using ten-fold cross-validation.

4.4.2 Results with Label Ranking Trees

We evaluate the two variants of ranking trees described earlier: ranking trees (RT) and entropy ranking trees (ERT) (Sections 4.2.2 and 4.2.3). The RT algorithm has a parameter γ, that can affect the accuracy of the model. Based on previous results, we use $\gamma = 0.98$ for RT [35].

Table 4.4 presents the results obtained by the two decision tree approaches, RT and ERT, in comparison to the results for Label Ranking Trees (LRT), that are reproduced from the original paper [26]. We note that we have no information about the depth of the trees obtained with the latter and thus such information is omitted in Table 4.4. Even though LRT performs best in most of the cases presented, both RT and ERT are also competitive methods.

Figure 4.1 shows how much smaller ERT trees are, in general. By generating smaller trees, ERT provides more interpretable models when compared with RT. An exception is the *calhousing* dataset, where ERT generates larger

Table 4.4: Results obtained for Ranking Trees on KEBI datasets (the mean accuracy is represented in terms of Kendall's tau, τ; the best mean accuracy values are in **bold**)

	RT		ERT		LRT
	mean accuracy	depth	mean accuracy	depth	mean accuracy
authorship	.883	8.0	**.889**	4.0	.882
bodyfat	.111	11.9	**.182**	2.7	.117
calhousing	.182	1.0	.291	11.6	**.324**
cpu-small	**.458**	17.2	.437	6.1	.447
elevators	.746	18.9	.757	7.9	**.760**
fried	.797	20.2	.774	13.2	**.890**
glass	.871	8.2	.854	3.0	**.883**
housing	.794	12.9	.704	3.4	**.797**
iris	**.963**	4.3	.853	2.0	.947
pendigits	.871	14.0	.838	5.9	**.935**
segment	.929	12.0	.901	5.0	**.949**
stock	**.897**	10.8	.859	5.0	.895
vehicle	.817	11.0	.787	4.1	**.827**
vowel	**.833**	12.5	.598	3.6	.794
wine	.905	4.0	**.906**	2.0	.882
wisconsin	.334	10.0	.337	2.3	**.343**
average	.712	11.1	.685	5.1	**.730**

trees. However, in this case, the increase in size is justified by a reasonable increase of accuracy (Table 4.4).

To compare different ranking methods, we use a combination of Friedman's test and Dunn's Multiple Comparison Procedure [104], which has been used before for this purpose [17]. First we run the Friedman's test to check whether the results are different or not, with the following hypotheses:

H_0 : The distributions of Kendall's τ are equal

H_1 : The distributions of Kendall's τ are not equal

Using the Friedman test (implemented in the *stats* package [113]) we obtained a p-value $< 1\%$, which shows strong evidence against H_0. This means that there is a high probability that the three methods have different performance.

Thus, we tested which of the three methods are different from one another

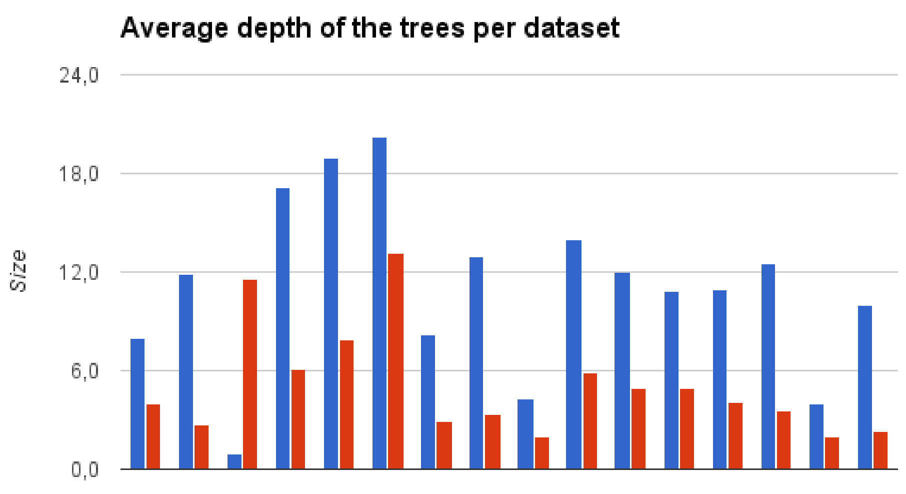


Figure 4.1: Comparison of the average depth of the trees obtained with RT (blue) and ERT (red) on KEBI datasets

with the Dunn's Multiple Comparison Procedure [104]. Using the R package *dunn.test* [45], we tested the following hypotheses for each pair of methods a and b:

H_0 The distributions of Kendall's τ for a and b are equal

H_1 The distributions of Kendall's τ for a and b are not equal

Table 4.5 indicates that there is no statistical evidence that the methods are different. The statistical tests confirm our observation that, although LRT generally obtains better results than RT and ERT, the latter approaches are competitive.

Table 4.5: Dunn's test results (p-values)

	RT	ERT	LRT
RT		0.22	0.37
ERT	0.22		0.13
LRT	0.37	0.13	

4.4.3 Results with Label Ranking Forests

We generated forests with 100 trees and aggregated the predicted rankings with the average ranking method [17]. Table 4.6 presents the results obtained by the Label Ranking Forests using RT and ERT, referred to as LRF-RT and LRF-ERT, respectively.

The average depth of the trees for LRF-RT is, for most cases, smaller than that of the tree obtained with the RT algorithm, while the accuracy is better. On average, for each 0.019 increase in accuracy there was a decrease of 1.8 in the average depth of the trees. One exception is the *elevators* dataset, with suffered a significant decrease in accuracy by using the LRF method.

The comparison between ERT and LRF-ERT leads to different observations. The average depth of the trees increases when using LRF. This can be explained by the fact that the measure of entropy for rankings used in ERT is very robust to noise in rankings [39]. Hence, it requires a larger amount of dissimilarity in a set of rankings to find a partition. As noted in Section 4.4.3 (Figure 4.1) the depth of the trees is much smaller with ERT than with RT.

In Figure 4.2, we can observe how much the accuracy increases/decreases with LRF when compared to the corresponding base-level trees alone. In the vast majority of datasets, there is some improvement in accuracy. The only exception is the *elevators* dataset, as mentioned above.

Using the same statistical tests as before (Section 4.4.2), we compare LRF-RT and LRF-ERT with the RT, ERT and LRT methods. With the Friedman's test we got a p-value $< 1\%$, which shows strong evidence against H_0. Then, now that we know that there are some differences between the 2 methods we will test which are different from one another with the Dunn's Multiple Comparison Procedure (Table 4.7). Since we got a p-value around 25%, between LRF-RT and the LRF-ERT, we cannot conclude that there is no statistical evidence that the methods are different.

On the pairwise comparisons of the methods Table 4.8, we measure how many times each method wins, in terms of accuracy. In this analysis, we conclude that Label Ranking Forests using RT give the best results, proving the effectiveness of the approach.

On the other hand, even though LRF-ERT shows some improvement in terms of accuracy relatively to ERT, it did not behave much better than RT or LRT (Table 4.8). Again, this might be caused by the fact that the measure

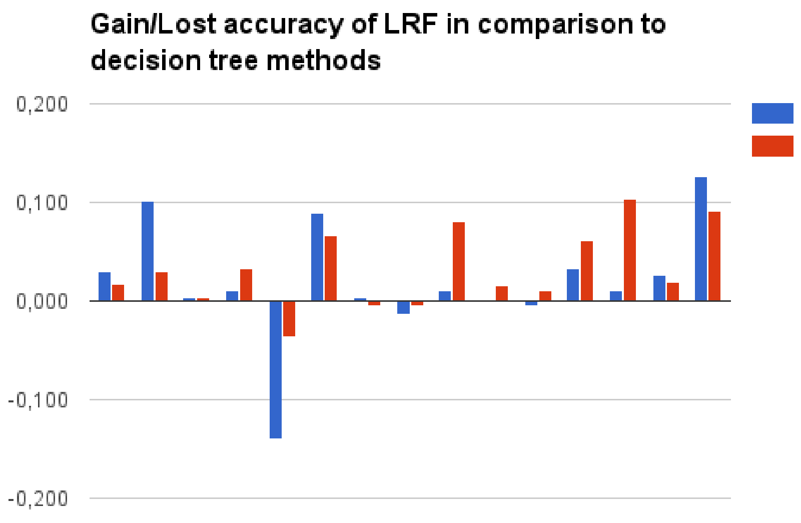


Figure 4.2: Accuracy gained/lost per dataset for using the ensemble method LRF, instead of standalone decision trees RT (blue) and ERT (red) on KEBI datasets

of entropy for rankings used in ERT is very robust to noise. For this reason, the depth of trees in LRF-ERT is, on average, 70% the depth of trees in LRF-RT. While this can be an advantage in terms of Label Ranking Trees, in Label Ranking Forests it is less relevant because it is hard to interpret 100 trees per dataset.

4.5 Conclusions

In this work, we propose an ensemble of decision tree methods for Label Ranking, called Label Ranking Forests (LRF). The method is tested with two different base-level methods Ranking Trees (RT) and Entropy-based Ranking Trees (ERT). We present an empirical evaluation using well known datasets in this field. We also extend the analysis from previous work for tree-based methods, RT and ERT, and compare with the state of the art *Label Ranking Trees* (LRT) approach.

The analysis on the decision trees shows that both RT and ERT are valid and competitive approaches. While RT usually gives better accuracy, on the other

Table 4.6: Results obtained for Label Ranking Forests on KEBI datasets, using two different label ranking trees, RT and ERT (the mean accuracy is represented in terms of Kendall's tau, τ; the best mean accuracy values are in **bold**)

	LRF-RT		LRF-ERT	
	mean accuracy	depth	mean accuracy	depth
authorship	**.912**	8.3	.906	7.7
bodyfat	**.212**	10.6	.211	5.3
calhousing	.185	1.0	**.294**	8.3
cpu-small	.469	13.9	**.471**	7.8
elevators	.605	10.0	**.721**	9.5
fried	**.887**	15.5	.841	14.5
glass	**.874**	6.0	.849	2.7
housing	**.780**	10.9	.699	3.7
iris	**.973**	4.9	.933	2.3
segment	**.930**	10.8	.917	5.2
stock	**.892**	9.9	.869	5.5
vehicle	**.850**	10.0	.849	9.4
vowel	**.844**	11.5	.701	4.9
wine	**.932**	4.3	.925	2.8
wisconsin	**.460**	8.8	.429	3.7
average	**.720**	9.1	.708	6.2

hand, ERT generates trees with much smaller depth (around 50% less, in comparison to RT). Our results were also compared with the published results for Label Ranking Trees (LRT) [26]. LRT has in general better accuracy than RT and ERT, however, statistical tests showed that none of the methods is significantly different. This means that both RT and ERT are competitive approaches, and, since they are distance-based methods, we can also say that this kind of approaches is worth pursuing.

The two ensemble approaches, LRF-RT and LRF-ERT, used the base ranking tree models RT and ERT, respectively. Similarly to the application of Random Forests to other tasks, there was a general increase in accuracy when compared to the corresponding base-level methods. The results confirm that both LRF-RT and LRF-ERT are highly competitive LR methods. LRF-RT, in particular, stands out as a clear winner in terms of accuracy.

As future work, we might improve the comparison with LRT method [26], by implementing it and testing it both as learning algorithm and as the base-level method for Label Ranking Forests. Also, LRF can potentially

Table 4.7: Dunn's test for all the methods (p-values)

	RT	ERT	LRT	LRF-RT	LRF-ERT
RT		0.23	0.34	0.31	0.44
ERT	0.23		0.13	0.11	0.28
LRT	0.34	0.13		0.46	0.29
LRF-RT	0.31	0.11	0.46		0.25
LRF-ERT	0.44	0.28	0.29	0.25	

Table 4.8: Pairwise comparisons of the methods in terms of win statistics.

	RT	ERT	LRT	LRF-RT	LRF-ERT	Total (Rank)
RT		9	6	3	7	25 (4)
ERT	6		3	2	4	15 (5)
LRT	9	12		7	9	37 (2)
LRF-RT	12	13	8		13	46 (1)
LRF-ERT	8	11	6	2		27 (3)

produce similar benefits as the Random Forest method, in terms of feature selection or input variable importance measurement, when applied to LR datasets. Finally, the experiments in this paper were carried out on a set of standard benchmark datasets, which represent artificial LR problems. We plan to apply these approaches on real world datasets e.g. related with user preferences [81].

Acknowledgments

This work is financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project ≪POCI-01-0145-FEDER-006961≫, and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

Chapter 5

Exceptional Preferences Mining

Cláudio Rebelo de Sá, Wouter Duivesteijn, Carlos Soares, Arno Knobbe

in the proceedings of Discovery Science, 2016

Abstract

Exceptional Preferences Mining (EPM) is a crossover between two subfields of datamining: local pattern mining and preference learning. EPM can be seen as a local pattern mining task that finds subsets of observations where the preference relations between subsets of the labels significantly deviate from the norm; a variant of Subgroup Discovery, with rankings as the (complex) target concept. We employ three quality measures that highlight subgroups featuring exceptional preferences, where the focus of what constitutes 'exceptional' varies with the quality measure: the first gauges exceptional overall ranking behavior, the second indicates whether a particular label stands out from the rest, and the third highlights subgroups featuring unusual pairwise label ranking behavior. As proof of concept, we explore five datasets. The results confirm that the new task EPM can deliver interesting knowledge. The results also illustrate how the visualization of the preferences in a Preference Matrix can aid in interpreting exceptional preference subgroups.

5.1 Introduction

Consider a survey where detailed preferences of sushi types have been collected, along with demographic details of the respondents. For each example in the dataset, we have personal details (age, gender, income, etc.) as well as a set of sushi types, ordered by preference [81]. By mapping the demographic attributes and unusual preferences, marketeers would be able to target key demographics where specific sushi types have greater potential.

The study of preference data has been approached from a number of perspectives, grouped under the name *Preference Learning* (PL) (e.g., as Label Ranking [39, 25, 123]). Typically, the aim is to build a global predictive model, such that the preferences can be predicted for new cases. However, in several areas, such as marketing, there is also great value in identifying subpopulations whose preferences deviate from the norm. If some sushi type is markedly under preferred by a certain age group or in a certain region, then the vendor can develop specific strategies for those groups. Finding coherent groups of customers to focus on is an invaluable part of promotion strategies.

Arguably the most generic setting for discovering local, supervised deviations is that of Subgroup Discovery (SD) [88]. The aim of SD is to discover subgroups in the data for which the target shows an unusual distribution, as compared to the overall population [88]. SD is generic in the sense that the actual nature of the target variable can be quite diverse [1, 79, 121]. In this paper, we develop a Subgroup Discovery approach that focuses on a deviation target concept representing preferences over a fixed set of labels.

5.1.1 Main Contributions

This work provides focus specifically on the discovery of meaningful subgroups with exceptional preference patterns (see Section 5.4). We propose three quality measures for this purpose, reflecting different facets of interestingness one might have about the unusual preferences. All quality measures contrast the ranking of the labels in the subgroup with the ranking of the labels in the entire dataset; they differ in the granularity of the measured deviation. A subgroup is deemed interesting by the first quality measure if the overall ranking is exceptional, by the second quality measure if one particular label behaves exceptionally, and by the third quality measure if a single pair of labels displays exceptional behavior. Hence, Exceptional Preferences

Mining provides subgroups displaying exceptional ranking behavior; different quality measures allow for this exceptional behavior to either encompass the entire label space, or focus on more local peculiarities.

5.2 Label Ranking

Label Ranking (LR) studies the problem of learning a mapping from instances to rankings over a finite number of predefined labels [74]. It can be considered a variant of the conventional classification problem [26]. However, in contrast to a classification setting, where the objective is to assign examples to a specific class, in LR we are interested in assigning a complete preference order of the labels to every example.

More formally, in classification, given an instance x from the instance space $\mathbb{X}$, the goal is to predict the label (or class) λ to which x belongs, from a predefined set $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$. In Label Ranking, the goal is to order the labels in $\mathcal{L}$ by their association with x. A ranking is a total order over $\mathcal{L}$ defined on the permutation space Ω. A total order can be represented as a permutation π of the set $\{1, \ldots, k\}$, such that $\pi(a)$ is the position of λ_a in π.

A total order

$$\lambda_{\pi(1)} \underset{x}{\succ} \lambda_{\pi(2)} \underset{x}{\succ} \ldots \underset{x}{\succ} \lambda_{\pi(k)}$$

is associated with every instance $x \in \mathbb{X}$, representing a ranking $\pi \in \Omega$. In cases where the orders are partial, they are represented as rankings with ties [57].

The goal in label ranking is to learn the mapping $\mathbb{X} \rightarrow \Omega$. The training data is defined as D, which is a bag of n records of the form $x = (a_1, \ldots, a_m, \pi)$, where $\{a_1, \ldots, a_m\}$ is set of values from m independent variables $\mathcal{A}_1, \ldots, \mathcal{A}_m$ describing instance x and π is the corresponding target ranking.

Pairwise comparisons have been used to decompose LR or Multi-Label problems into binary problems [74]. In LR, the most relevant approach is Ranking by Pairwise Comparisons (RPC) [56], which decomposes the LR problem into a set of binary classification problems. Then, a learning method is trained with all examples for which either $\lambda_i \succ \lambda_j$ or $\lambda_j \succ \lambda_i$ is known [56]. The resulting predictions are then combined to predict a total or partial ranking [25].

Recently, some approaches have been suggested for mining preferences and ranks [70, 122]. These approaches tackle different problems from the one we propose in this paper. In [70], the authors suggest an approach to mine the rankings with association rules that search for *subranking* patterns, while our approach relates the ranking patterns with descriptors (otherwise referred to as independent variables). From a different perspective, [122] suggests a *ranked tiling* approach to search for rank patterns, whereas we are interested in the preference relations derived from the ranks.

5.3 Subgroup Discovery and Exceptional Model Mining

Subgroup Discovery (SD) [88] is a data mining framework that seeks subsets (satisfying certain user-specified constraints) of the dataset where something exceptional is going on. In SD, we assume a flat-table dataset D, which is a bag of n records of the form $x = (a_1, \dots, a_m, t_1, \dots, t_\ell)$. We call $\{a_1, \dots, a_m\}$ the *descriptors* and $\{t_1, \dots, t_\ell\}$ the *targets*, and we denote the collective domain of the descriptors by $\mathcal{A}$. We are interested in finding interesting subsets, called *subgroups*, that can be formulated in a *description language* $\mathcal{D}$. In order to formally define subgroups, we first need to define the following auxiliary concepts.
Definition 1 (Pattern and coverage)**.** *Given a description language* $\mathcal{D}$*, a* pattern $p \in \mathcal{D}$ *is a function* $p : \mathcal{A} \to \{0, 1\}$*. A pattern* p covers *a record* x *iff* $p(a_1, \dots, a_m) = 1$*.*

Patterns induce subgroups, and subgroups are associated with patterns, in the following manner.
Definition 2 (Subgroup)**.** *A* subgroup *corresponding to a pattern* p *is the bag of records* $S_p \in D$ *that* p *covers:*

$$S_p = \{x \in D \mid p\,(a_1, \dots, a_m) = 1\}$$

For simplicity, we will loosely identify pattern and subgroup with each other.

The exact choice of the description language is left to the domain expert or analyst. A typical choice is the use of conjunctions of conditions on attributes. Restricting the findings of SD from all subsets to only subgroups that can be defined in such a way, ensures results of the following form:

$$\text{Age} \geq 30 \;\wedge\; \text{Likes} = \text{Salmon Roe} \quad \text{is unusual}$$

Restricting the search from subsets to subgroups, combined with a sensible choice of description language, ensures that SD delivers subgroups that are defined in terms of attributes of the dataset. This means that the results are delivered in a form with which dataset domain experts are familiar. In other words, the focus of SD lies on delivering *interpretable* results.

Formally, the interestingness of a subgroup can be measured using all information available in its associated pattern. In practice, it depends on the task we are trying to solve. Therefore, we should define one or more *quality measures* to assess the interestingness we want to explore.
Definition 3 (Quality Measure)**.** *A* quality measure *is a function* $\varphi : \mathcal{D} \to \mathbb{R}$.

In the most common form of pattern mining, *frequent itemset mining* [3], interestingness is measured by the frequency of the pattern. *Subgroup Discovery* [88], on the other hand, measures interestingness in a *supervised* form. One designated target t_1 is identified in the dataset, and subgroup interestingness is gauged by an unusual distribution of that target. Hence, considering that a poll revealed that the majority of Japanese people like *Fatty tuna* sushi, an interesting subgroup could refer to a group of people for which the majority prefers *Tuna roll*:

$$\text{Age} \geq 30 \ \wedge \ \text{Lives in region} = \text{Hokkaido} \quad \Rightarrow \quad \text{Likes} = \text{Tuna roll}$$

If instead of a single target, multiple targets $t_1, \ldots, t_\ell$ are available, and if we are not interested in finding unusual target distribution, but unusual target interaction, we can employ *Exceptional Model Mining* (EMM) [48, 49] instead of SD. EMM is instantiated by selecting two things: a model class and a quality measure. Typically, a model class is defined to represent the unusual interaction between multiple targets we are interested in. A specific quality measure that employs concepts from that model class must be defined to express exactly when an interaction is unusual and, therefore, interesting.

The target concept at hand in this paper has only one target object t, which resembles SD. However, that target object is a label ranking $\pi \in \Omega$, as defined in Section 5.2. Hence it represents unusual interactions between multiple individual labels, which is more consistent with EMM.

5.3.1 Traversing the Search Space

Typically, *subgroups* are found by a level-wise search through attribute space [100]. We define constraints on single attributes and define the corresponding subgroups as those records satisfying each one of those constraints.

The actual phenomenon of the data that a given quality measure favors, depends on the target concept (binary, numeric, preferences, ...). For very small subgroups, one easily finds an unusual distribution of the target. To favor larger subgroups, one defines the quality measure such that it balances the exceptionality of the target distribution with the size of the subgroup.

SD approaches have been developed for binary, nominal [1] and numeric target variables [77, 79], as well as for targets encompassing multiple attributes [121]. However, none of the previous approaches is able to capture all the sets of preferences that can be derived from rankings within an SD framework.

5.4 Exceptional Preferences Mining

Exceptional Preferences Mining (EPM) is the search for subgroups with deviating preferences. Exactly what constitutes an interesting deviation in preferences is governed by the employed quality measure, and can be inspired by the application at hand.

When the number of labels is large, the search for preference patterns can be hard to analyze and visualize. A real world example is the Sushi dataset [81], which represents the preferences of 5 000 persons over 10 types of sushi. Even this relatively modest number of sushi types can be ranked in a large number of combinations: more than 98% of the 5 000 rankings present in this dataset are unique. This illustrates why it can be more difficult to directly learn a ranker that associates a reliable complete ranking for any subset in $\mathbb{X}$ when the number of labels is large.

In EPM, we want to search for strong preference behavior. However, in cases like the Sushi dataset, it is difficult to get strong total orders, due to the low number of ranking repetitions. In other words, searching for subgroups where all types of sushi are consistently ranked in this exact same order can be unfruitful. For this reason, we also propose lower-granularity measures that focus on one label versus the others (Labelwise). That is, we look for subgroups where at least one type of sushi is often preferred to all the others. As an example, if a subgroups ranks *tekka-maki* consistently in the top 3 while the majority in the dataset ranks it in the last 3, this measure will find it to be very interesting. We also propose a measure of even lower granularity, focusing on label versus label (Pairwise) preferences. This means that, if most people display a preference *tamago* $\succ$ *kappa-maki*, a subgroup where most people prefer *kappa-maki* $\succ$ *tamago* will be deemed interesting by this measure.

Our assumption is that, even though over 98% of the total rankings in the Sushi dataset are unique, there is plenty of information present in these rankings: the partial orders and pairwise comparisons can reveal interesting subgroups.

5.4.1 Preference Matrix

Let us define a function, ω, assigning a numeric value to the pairwise comparison of the labels λ and $\hat{\lambda}$:

$$\omega\left(\lambda,\hat{\lambda}\right) = \begin{cases} 1 & \text{if } \lambda \succ \hat{\lambda} \text{ (}\lambda\text{ preferred to }\hat{\lambda}\text{)} \\ -1 & \text{if } \lambda \prec \hat{\lambda} \text{ (}\hat{\lambda}\text{ preferred to }\lambda\text{)} \\ 0 & \text{if } \lambda \sim \hat{\lambda} \text{ (}\lambda\text{ indifferent to }\hat{\lambda}\text{)} \\ n/a & \text{if } \lambda \perp \hat{\lambda} \text{ (}\lambda\text{ incomparable to }\hat{\lambda}\text{)} \end{cases}$$

Note that, by definition, $\omega\left(\lambda,\hat{\lambda}\right) = -\omega\left(\hat{\lambda},\lambda\right)$. We can use ω to represent a ranking π as a *Preference Matrix* (PM), M_π:

$$M_\pi(i,j) = \omega_\pi(\lambda_i,\lambda_j)$$

M_π is, by definition, an antisymmetric matrix with $tr(M_\pi) = 0$. PMs can natively represent partial or incomplete orders but can also be aggregated to represent sets of rankings from an entire dataset D or subgroup S. To aggregate the entries, the *mean* or the *mode* can be used.

Aggregation of a PM for sets of rankings

The PM of a set of rankings from a dataset D with n rankings, M_D, aggregated with the mean is:

$$M_D(i,j) = \frac{1}{n}\sum_{\pi\in D} M_\pi(i,j)$$

Table 5.1: Example dataset $\hat{D}$. The first column is the only descriptor. The subsequent four columns represent the preferences among four labels, by providing their ranks. An alternative representation is presented in the rightmost section of the table.

$\mathcal{A}_1$	π				alternative π
	λ_1	λ_2	λ_3	λ_4	
0.1	4	3	1	2	$\lambda_3 \succ \lambda_4 \succ \lambda_2 \succ \lambda_1$
0.2	3	2	1	4	$\lambda_3 \succ \lambda_2 \succ \lambda_1 \succ \lambda_4$
0.3	1	4	2	3	$\lambda_1 \succ \lambda_3 \succ \lambda_4 \succ \lambda_2$
0.4	1	3	2	4	$\lambda_1 \succ \lambda_3 \succ \lambda_2 \succ \lambda_4$

The PM of the example dataset $\hat{D}$ (cf. Table 5.1) is the following:

$$M_{\hat{D}} = \begin{bmatrix} 0 & 0 & 0 & 0.5 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ -0.5 & 0 & -1 & 0 \end{bmatrix}$$

This representation enables easy detection of strong partial order relations in a set. If row i has all the values very close to 1, then λ_i is highly preferred in this group. If entry $M_{\hat{D}}(i,j) = 1$ or $M_{\hat{D}}(i,j) = -1$, then all rankings in $\hat{D}$ agree that $\lambda_i \succ \lambda_j$ or $\lambda_i \prec \lambda_j$, respectively.

All the elements of $\hat{D}$ reveal distinct total preferences, but λ_3 is always preferred to λ_2, which is easily verified by checking that $M_{\hat{D}}(3,2) = 1$. In the ranking representation of $\hat{D}$, this fact follows from four distinct combinations of ranks: rank $3 > 1$, rank $2 > 1$, rank $4 > 2$ and rank $3 > 2$ (this information is found in the two columns below λ_2 and λ_3). Conversely, λ_4 is never preferred to λ_3, which is represented by $M_{\hat{D}}(4,3) = -1$. In some cases, the overall trend is not as clear (e.g. λ_1 is preferred to λ_4 but not always) and in other cases, there is no trend at all (e.g. λ_1 and λ_2).

Representing a set of rankings as a PM has another advantage over the traditional permutation representation: it enables simple measurement of *labelwise* (by rows/columns of the PM) and *pairwise* (by single entries of the PM) distances (see Section 5.4.2).

From the PM of a subgroup S, one can derive a new ranking π_S. How to do so is a non-trivial question, which has received a lot of attention in several research fields with similar types of matrix [74]. The straightforward way is to sum the rows of the PM and then assign a score to each corresponding label. Higher values correspond to a relatively more preferred label.

The generation of a PM is basically a pairwise decomposition problem. The complexity is $\mathcal{O}\left(sk^2\right)$ per subgroup, where s is the size of the subgroup and k the number of labels in the ranking. Even though any number of labels is theoretically permitted in label ranking, in practice the number of labels is usually smaller than 20. Hence, the generation of PMs should not be an issue in terms of computational time.

We use a visual representation of PM that is a set of colored tiles (cf. Figure 5.1). Each tile represents an entry of the PM. The entries of a PM can vary from -1 to 1. The negative entries of the matrix are represented with red tiles, the positive with green tiles, and 0 is represented in white. The colored tiles fade out as they get closer to 0.

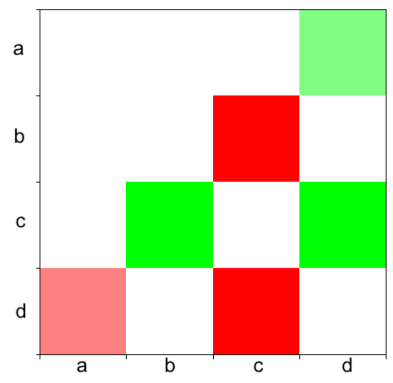


Figure 5.1: PM representation of the set of rankings in $\hat{D}$ (cf. Table 5.1). Dark green tiles represent 1 and dark red tiles represent -1.

5.4.2 Characterizing Exceptional Subgroups

The table has now been set to formally define the quality measures for EPM, which will evaluate how exceptional the preferences are in the subgroups. A subgroup can be considered interesting both by the amount of deviation (distance) and by its size (number of records *covered* by the subgroup, cf. Section 5.3) [52]. Since, reasonable quality measures should take both these factors into account, we divide the quality measures into two parts: the *distance* component and the *size* component.

$$QM_S = size_S \cdot distance_S$$

In order to allow direct comparisons between different quality measures, both components are normalized to the interval $[0, 1]$. A common measure for the size in Subgroup Discovery is $\sqrt{s}$ [87]. To normalize, we use the square root of the fraction of the dataset covered by S: $size_S = \sqrt{s/n}$.

Before introducing the distance components, let us first define a distance matrix L_S, as the distance matrix between the PMs M_S and M_D:

$$L_S = \frac{1}{2}(M_D - M_S)$$

where $S \subseteq D$ (division by 2 limits the distance to the interval $[-1, 1]$). We can measure different properties of L_S and represent them with a numeric value. This way we get an indicator of the quality of the distance of preferences for a subgroup. Consider the subgroup $\hat{S}_1 : A_1 \geq 0.3$, which covers the last two cases from our example dataset $\hat{D}$. Its PM is:

$$M_{\hat{S}_1} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ -1 & 0 & -1 & 0 \\ -1 & 1 & 0 & 1 \\ -1 & 0 & -1 & 0 \end{bmatrix}$$

The first row clearly reveals that λ_1 is always preferred to all other labels in this subgroup. If we compute the difference matrix $L_{\hat{S}_1}$ we get:

$$L_{\hat{S}_1} = \begin{bmatrix} 0 & -0.5 & -0.5 & -0.25 \\ 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0.25 & 0 & 0 & 0 \end{bmatrix}$$

The difference matrix $L_{\hat{S}_1}$ shows that the behavior of λ_1 is exceptional in $\hat{S}_1$.

Only subgroups for which we can infer at least one pairwise preference are considered interesting in Exceptional Preferences Mining. That is, subgroups with a PM containing only zeros are not considered interesting.

As we are interested in subgroups with exceptional preferences, we use the distance matrix L_S to measure exceptionality. The distance measures we employ here typically consider a particular subset of the cells of the distance matrix L_S.

Norm

Maximizing the distance of preferences is also maximizing the magnitude of L_S. The most fundamental mathematical way to measure the magnitude of a vector or matrix is the *norm.* Hence we can use the *Frobenius norm* of L_S as a distance measure.

$$\text{Norm}(S) = \sqrt{s/n} \cdot ||L_S||_F = \sqrt{s/n} \cdot \sqrt{\sum_{i=1}^{k} \sum_{j=1}^{k} L\,(i,j)^2}$$

If one is searching for preference deviations in general, one should use the *Norm* quality measure, as it considers all the PM entries at the same time. After the subgroups are found, ideally, we can derive a complete ranking from their PMs. The overall deviation can be due to one label deviating strongly or from multiple labels deviating less strongly.

Labelwise

An interesting task in the PL field is the *labelwise* analysis [25]. Instead of focusing on a whole ranking, it focuses on the preference behavior from the perspective of individual labels. A data analyst might be interested in finding if a particular label λ behaves substantially different according to most members in a subgroup S, compared to its behavior on the overall dataset. Hence, the fact that only one label behaves differently, disregarding the interaction between the other labels, can also be interesting. We can measure the distance of each label, in subgroup S, by computing the *norm* of the rows from L_S. Since in this case we are interested in exceptionality of only one label, we consider the maximum value found:

$$\text{Labelwise}(S) = \sqrt{s/n} \cdot \max_{i=1,\dots,k} \frac{1}{(k-1)} \sum_{j=1}^{k} L\,(i,j)$$

Pairwise

Another well-studied task in PL is Pairwise Preferences [74] which decomposes the preferences into pairs label-vs-label. In situations where there are not even exceptional labelwise preferences, one can still search for localized preference strongholds. If we are interested in subgroups with, at least one pair with distinctive preference behavior, we can employ the following quality measure:

$$\text{Pairwise}(S) = \sqrt{s/n} \cdot \max_{i,j=1,\dots,k} L\,(i,j)$$

This quality measure is the least restrictive of this set: a subgroup is interesting if one pair of labels interacts unusually, disregarding all other label interactions.

5.5 Experiments

We incorporate Exceptional Preferences Mining in the Cortana[1] software package [101]. This package delivers a generic framework for SD, implements several SD instances, and offers many generic features allowing for different SD approaches. The description language consists of logical conjunctions of conditions on single attributes.

Our experiments use a standard *beam search* approach. Since the Subgroup Discovery algorithm itself is not the topic of this paper, we will skip over the algorithmic details, but they can be found elsewhere: the relevant pseudo-code is given in [48, Algorithm 1]. The most influential parameters are set as follows: we use a relatively generous search width w (also known as beam width or beam size) of 100, allowing for a relatively broad (albeit heuristic) search, and a maximum search depth d of 2, which keeps the resulting subgroups interpretable. We explore some striking subgroups found with the quality measures on a variety of datasets, providing evidence of the versatility of our work.

All the findings we present in this paper have gone through the DFD validation procedure [50] with 100 copies, and all have been found significant at a significance level of $\alpha = 1\%$.

5.5.1 Datasets

Statistics regarding the datasets used in this work are shown in Table 5.2. The majority are Label Ranking datasets from the KEBI Data Repository at Philipps University of Marburg [26]. These datasets were adapted from multi-class and regression problems both from the UCI repository [96] and the Statlog collection [26]. In the process, the features were normalized, and their names were replaced by $A1, A2, \ldots, Am$. Therefore, on these datasets, the reported subgroups cannot be interpreted on the original dataset domain,

[1] `http://datamining.liacs.nl/cortana.html`

whereas for general datasets, this interpretability is a key feature of Exceptional Preference Mining. We choose to experiment with these datasets anyway, since they are well-known in the preference learning community.

For illustrating domain-specific interpretation of the results, we experiment with two further datasets. We adapt the COIL 1999 Competition Data from UCI [96]. This dataset concerns the frequencies of algae populations in different environments. We refer to this dataset as *Algae*. The original COIL dataset consists of 340 examples, each representing measurements of a sample of water from different European rivers in different periods. The measurements include concentrations of chemical substances such as nitrogen (in the form of nitrates, nitrites and ammonia), oxygen and chlorine. Also the pH, season, river size and flow velocity are registered. For each sample, the frequencies of 7 types of algae are also measured. In this work, we consider the algae concentrations as preference relations by ordering them from larger to smaller concentrations. Those with 0 frequency are placed in last position and equal frequencies are represented with ties. Missing values are set to 0.

Our final dataset is the Sushi preference dataset [81], which is composed of demographic data about 5 000 people and sushi preferences. Each person sorts a set of 10 different sushi types by preference. The 10 types of sushi, are a) shrimp, b) sea eel, c) tuna, d) squid, e) sea urchin, f) salmon roe, g) egg h) fatty tuna, i) tuna roll and j) cucumber roll. Since the attribute names were not transformed in this dataset, we can make a richer analysis of it.

Table 5.2: Dataset details. The column U_π represents the percentage of unique rankings.

Datasets	#examples	#labels	#attributes	U_π
Cpu-small	8 192	5	6	1%
Elevators	16 599	9	9	1%
Wisconsin	194	16	16	100%
Algae (COIL)	316	7	10	72%
Sushi	5 000	10	10	98%

For all the experiments, all results and statistical tests are completed in less than 5 minutes on an Intel Core 2 Duo CPU @ 2.93GHz with 4GB RAM.

5.5.2 Results

We start this section by presenting a discovery which provides an exemplary demonstration of one advantage of the PM representation.

Elevators dataset

Figure 5.2 shows the subgroup with highest score found with the *Norm* quality measure in the Elevators dataset. Considering the base matrix, which has information from all the rankings in the dataset, we conclude that e, f, g, h have fixed relative positions: $e \succ g \succ f \succ h$. This information is not easy to obtain with the usual representations of rankings, but is clearly revealed in the PM representation. In fact, 13 403 from a total of 16 599 rankings have $e \succ g \succ f \succ h$. This illustrates how the visual ranking representation in a PM can be very useful for supporting predictive methods and for data exploration. The subgroup, $A6 \geq 0.436$, covering 7 048 instances, had a norm of 0.0028. It shows a distinct behavior between the sets of labels a, b, c, d and the set e, f, g, h. In the whole data, labels a, b, c, d are a bit more desirable than e, f, g, h. However, in the subgroup, the latter are clearly preferred to a, b, c, d.

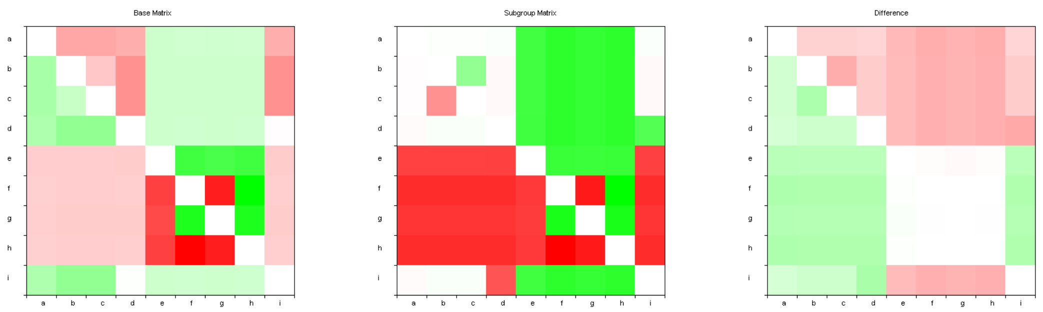


Figure 5.2: PM representation of the dataset Elevators (base matrix), the subgroup $A6 \geq 0.436$ (subgroup matrix) and the difference (difference matrix).

Wisconsin

Using the *Norm* quality measure on the Wisconsin dataset, we obtain 30 subgroups, the 1$^{\text{st}}$-ranked of which (it happens to occur at depth 1 in the search) is represented in Figure 5.3. The base matrix reveals that the dataset has balanced preferences, by the low intensity of the colored tiles. The red rows of the PM of subgroup $A_5 \leq -0.527$ (Subgroup Matrix in Figure 5.3) indicate a strong behavior of the labels f, h and i. The PM reveals that labels f, h, i are consistently ranked lower than the other labels in this specific subgroup. Since PMs are antisymmetric, the 3 green columns represent the same phenomena but from the perspective of the other labels. If we focus

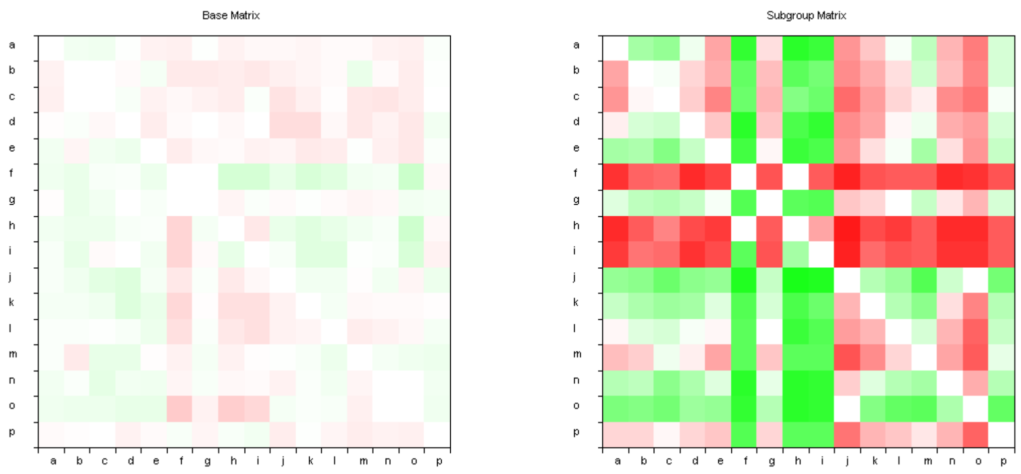


Figure 5.3: PM representation of the dataset Wisconsin (Base Matrix) and the subgroup $A_5 \leq -0.527$ (Subgroup Matrix).

on these 3 labels, we can see that tile (f, h) is white, which means f and h are equivalent. On the other hand, tiles (i, f) and (i, h) are green, which means that $i \succ f$ and $i \succ h$. If one had to guess a reliable partial order from this subgroup using only the PM, a logical choice would be to say that $a, b, c, d, e, g, j, k, l, m, n, o, p \succ i \succ f, h$.

Algae

With the Algae dataset, we obtain results about the concentrations of algae with the *Norm* measure. One such example is that during *Spring*, the types of algae a, b and c are much more common in rivers than the others. This can be easily concluded by studying the PM representation of the subgroup (Figure 5.4). This subgroup has a norm of 0.010647. On the other hand, we also see an interesting behavior during the *Autumn* season, with a norm of 0.01058.

With the *Labelwise* measure, we find more than 400 subgroups, the best of which is presented in Figure 5.5. The PM clearly reveals the effect of the Labelwise quality measure: in the subgroup, the label a is strongly preferred

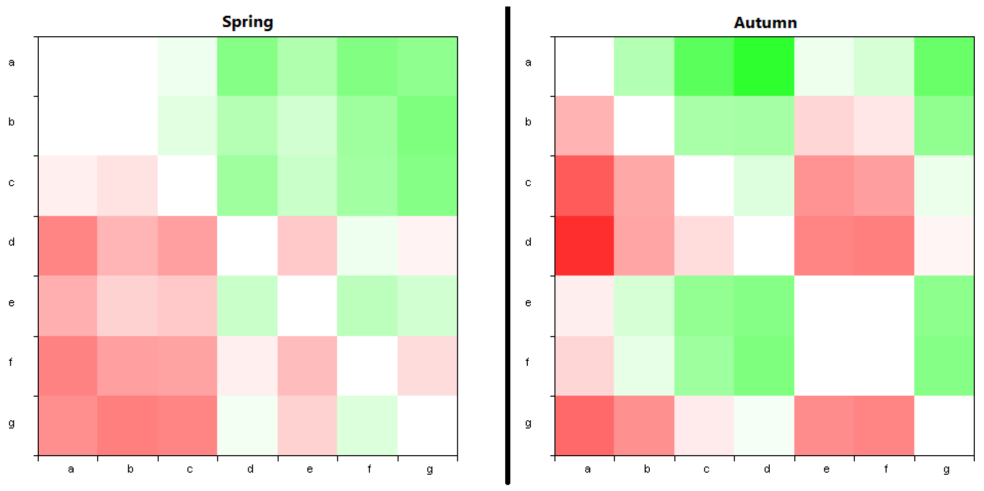


Figure 5.4: PM representation of the subgroups $Season = Spring$ (left subgroup matrix) and $Season = Autumn$ (right subgroup matrix) from the Algae dataset.

over all others, while the image is much more nuanced over the whole dataset. If we ignore the label a, the PMs for both the overall dataset and the subgroup are rather bland, and their difference is not very pronounced. But for this one particular label a, the behavior on the subgroup is extremely clear-cut, and the Labelwise quality measure picks up on that effect.

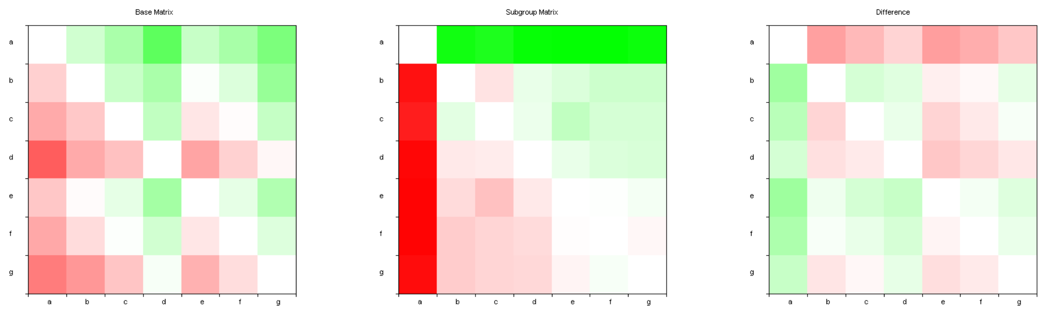


Figure 5.5: PM representation of the dataset Algae (base matrix) and the subgroup $V10 \leq 59 \wedge V6 \leq 11.867$ (subgroup matrix), with difference matrix on the right.

Sushi

With the *Labelwise* measure, we find 149 subgroups on the Sushi dataset. We present the best subgroup using this measure in Figure 5.6. The subgroup

(Males over 30 years) shows a preference for Sea Urchin, since the majority of men rank this sushi type in the top 4. By contrast, in the whole population, more than half rate it between 5th to 10th, and every fifth person rate it in last place.

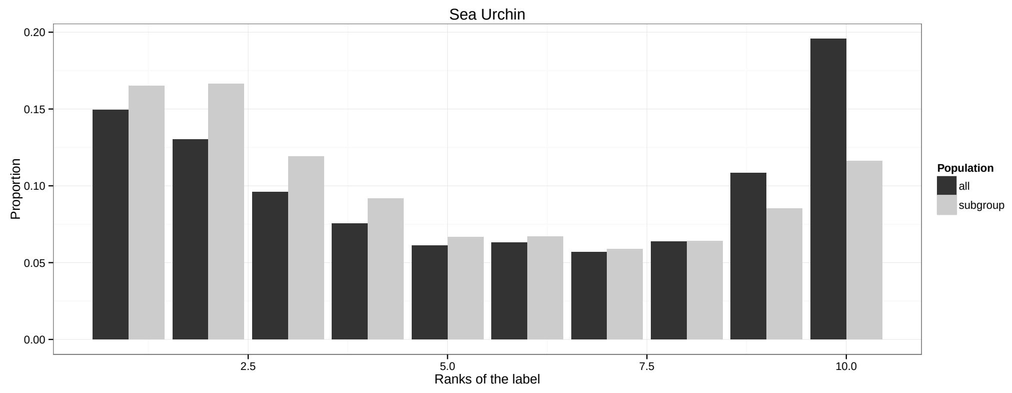


Figure 5.6: Percentage of ranks for Sea Urchin (Sushi dataset) for all individuals in comparison to the subgroup (males older than 30 years).

Cpu-small

On the Cpu-small dataset, the subgroup $A6 \geq 0.127$ ranks the best for the *Pairwise* quality measure. Around 80% of the 2 221 instances of this subgroup agree that $a \succ d$, in contrast to the 30% in the whole dataset of 8 192 instances.

5.6 Conclusions

We introduce Exceptional Preferences Mining (EPM), a supervised local pattern mining task where the target concept is a ranking of a fixed set of labels. The result of this task is a set of subgroups, which are coherent subsets of the dataset that can be described in terms of a conjunction of few conditions on an attribute, where the label preferences are exceptional in some sense.

The relevant statistics on a set of preference relations is collected in the cells of a Preference Matrix (PM). A PM is compiled for the entire dataset, and for each subgroup under consideration. A subgroup whose PM deviates significantly from the PM for the whole dataset is then considered to be interesting. We define three quality measures for EPM that instantiate this concept of 'interesting' to different levels of granularity. The *Norm* quality measure deems a subgroup interesting if the full set of preference relations is substantially displaced. The *Labelwise* quality measure highlights subgroups where any one label interacts exceptionally with the other labels, agnostic of how those other labels interact with each other. The *Pairwise* quality measure finds a subgroup interesting if any one pair of labels display exceptional preference relations. Hence, by choosing the appropriate quality measure, EPM delivers subgroups featuring preference relations that are exceptional at your preferred scope.

The experiments with the *Norm* quality measure on the Elevators dataset illustrate the value of the PM visualization. The PM, as displayed in Figure 5.2, clearly indicates that there are strong relations between a subset of the available labels. We learn that quite frequently, labels e, f, g, h have fixed relative positions: $e \succ g \succ f \succ h$. This information is not easy to obtain with the usual representations of rankings, but is clearly revealed through the PM visualization. The experiments with the *Labelwise* quality measure on the Sushi dataset illustrate the relative merit of this quality measure: it focuses on subgroups where one particular label is exceptionally under- or overappreciated. The subgroup presented has a penchant for Sea Urchin (cf. Figure 5.6). The *Pairwise* measure shows its potential on the Cpu-small dataset by identifying a subgroup with strong exceptional preferences with respect to the pair of labels a and d.

As we argued in Section 5.3, one of the main benefits of a local pattern mining method such as EPM is that it delivers interpretable results. That means that the resulting subgroups are ideally suited to instigate real-world policies and actions. However, due to the employed preprocessing in the KEBI datasets (cf. Section 5.5.1), interpretation of results on those datasets falters. Only the experiments on the Algae and Sushi datasets allow a more extensive exploration of interpretable results. In future work, we would be interested in evaluating EPM on more label ranking datasets that come with interpretable attributes.

Acknowledgments

This work was supported by the European Union through the ERC Consolidator Grant FORSIED (project reference 615517). This work was also financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961 and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

Chapter 6

Permutation Tests for Label Ranking

Cláudio Rebelo de Sá, Carlos Soares, Arno Knobbe

in local proceedings of the 27th Benelux Conference on Artificial Intelligence, 2015

Abstract

In recent years, many Label Ranking (LR) methods have been proposed, along with an increasing number of datasets. The validation of these algorithms has been done empirically, as is usual in Machine Learning, by testing them on a set of benchmark datasets. Due to the scarcity of real-world LR data, most of the experiments are based on LR datasets that were adapted from classification and regression datasets from the UCI repository and Statlog project. In this work, we want to test how strong is the relation between the target rankings and independent variables. In other supervised learning tasks, target swap randomization methods have been used to test it. We propose two target swap randomization approaches for LR and apply them on KEBI datasets. Our results show that there are meaningful relations between the independent variables and the target rankings and that the relative importance of each label in a ranking varies in some cases.

6.1 Introduction

The study of label ranking is receiving increased attention [27, 36, 28, 116, 64]. Label Ranking (LR) studies the problem of learning a mapping from instances to rankings over a finite number of predefined labels. It can be considered a variant of the conventional classification problem [26]. However, in contrast to a classification setting, where the objective is to assign examples to a specific class, in LR we are interested in predicting the (possibly incomplete) true preference order of the labels for every example. This means that the true ranking of the labels is available for the training examples.

Due to the lack of benchmark LR datasets, 16 semi-synthetic datasets were proposed in [26]. They are based on multi-class and regression datasets from the UCI repository and Statlog project. For multi-class problems, also referred as *type A*, the naive Bayes classifier was trained to give a probability score to each class, and the true ranking is based on that score. For the regression problems, *type B*, some numeric attributes were normalized and the true ranking was based on the relative order of their values.

Since then, this set of 16 datasets has been used by the majority and the most influential contributions in the Label Ranking field [28, 27, 116, 64]. However, it is unclear if the type B datasets contain any real relations between the target rankings and independent variables. While type A can be interpreted as the preferences of an agent, which in this case is the naive Bayes classifier, on type B, the relations is application-specific and it is unclear whether it exists or not. To test whether such a relation exist, we expect to find strong statistical validation of it.

In many data mining applications, Swap Randomizations techniques are used together with statistical tests to validate the significance of findings [62]. After swapping the position of the values along the attributes, the resulting models are compared with the ones obtained from the original data. Therefore, statistical significance tests can be used to validate the latter.

In this work, we investigate the usefulness of the type B datasets from the KEBI data repository by comparison to type A. For that purpose, we propose two swap randomization methods specific for the LR task. Our results show that both types of semi-synthetic data have relevant preference information.

The paper is organized as follows: Section 6.2 introduces the LR problem. Section 6.3 introduces the swap randomization and Section 6.4 describes the method proposed here. Section 6.5 presents the experimental setup and discusses the results. Finally, Section 6.6 concludes this paper.

6.2 Label Ranking

The LR task is similar to classification. In classification, given an instance x from the instance space $\mathbb{X}$, the goal is to predict the label (or class) λ to which x belongs, from a predefined set $\mathcal{L} = \{\lambda_1, \dots, \lambda_k\}$. In LR, the goal is to predict the ranking of the labels in $\mathcal{L}$ that are associated with x [74]. A ranking can be represented as a total order over $\mathcal{L}$ defined on the permutation space Ω. In other words, a total order can be seen as a permutation π of the set $\{1, \dots, k\}$, such that $\pi(a)$ is the position of λ_a in π.

As in classification, we do not assume the existence of a deterministic $\mathbb{X} \to \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω [26]. This means that, for each $x \in \mathbb{X}$, there exists a probability distribution $\mathcal{P}(\cdot|x)$ such that, for every $\pi \in \Omega$, $\mathcal{P}(\pi|x)$ is the probability that π is the ranking associated with x. The goal in LR is to learn the mapping $\mathbb{X} \to \Omega$. The training data is a set of instances $D = \{\langle x_i, \pi_i \rangle\}, i = 1, \dots, n$, where x_i is a vector containing the values $x_i^j, j = 1, \dots, m$ of m independent variables describing instance i and π_i is the corresponding target ranking.

Given an instance x_i with label ranking π_i, and the ranking $\hat{\pi}_i$ predicted by an LR model, we evaluate the accuracy of the prediction with a loss function on Ω. One such function is the number of discordant label pairs,

$$\mathcal{D}(\pi, \hat{\pi}) = \#\{(a,b)|\pi(a) > \pi(b) \wedge \hat{\pi}(a) < \hat{\pi}(b)\}$$

If normalized to the interval $[-1, 1]$, this function is equivalent to Kendall's τ coefficient [85], which is a correlation measure where $\mathcal{D}(\pi, \pi) = 1$ and $\mathcal{D}(\pi, \pi^{-1}) = -1$ (π^{-1} denotes the inverse order of π).

The accuracy of a model can be estimated by averaging this function over a set of examples. This measure has been used for evaluation in recent LR studies [26, 40] and, thus, we will use it here as well. However, other correlation measures, like Spearman's rank correlation coefficient [118], can also be used.

6.2.1 IB-PL

Instance-Based Placket-Luce (IB-PL) is an highly competitive method in label ranking proposed in [24]. It is a local prediction method based on the nearest neighbor estimation principle. Given a new instance $\hat{x}$ it uses the $\{\pi_1, \ldots, \pi_K\}$ rankings associated with the K nearest neighbors to predict the ranking $\hat{\pi}$ associated with $\hat{x}$. The estimation of $\hat{\pi}$ is made using a Maximum Likelihood Estimation of the Plackett-Luce (PL) model which assumes that the rankings have been produced independently of each other.

6.2.2 APRIORI-LR

APRIORI-LR is an algorithm that generates *Label Ranking Association Rules* (LRAR) [36] which are a straightforward adaptation of Class Association Rules (CAR): $A \rightarrow \pi$ Where $A \subseteq desc(\mathbb{X})$ and $\pi \in \Omega$. Where $desc(\mathbb{X})$ is the set of descriptors of instances in $\mathbb{X}$, typically pairs $\langle attribute, value \rangle$. Similar to how predictions are made with CARs in CBA (Classification Based on Associations) [97], when an example matches the antecedent of the rule, $A \rightarrow \pi$, the predicted ranking is π.

6.2.3 Datasets

Even though Label Ranking potentially has a large number of practical applications [74], before the KEBI datasets, there were not many datasets available [26]:

- Meta-learning [17] on which we try to predict a total ranking of a set of algorithms accordingly to the best expected accuracy for each dataset.
- Microarray [74] which provides information of genes from Yeast on five different micro-array experiments (spo, heat, dtt, cold and diau).
- Image categorization [58] of landscape pictures from several categories (beach, sunset, field, fall foliage, mountain, urban).

To solve this problem, the KEBI Label Ranking data repository was created [26]. Data from the UCI repository and Statlog collection was transformed into Label Ranking data using the following two procedures:

type A The multi-label data is used in the training of a naive Bayes classifier. The predicted class probabilities are ranked by decreasing order for each example, which will result in a label ranking. (To avoid incomplete rankings, the labels with lower indexes are ranked first in case of ties)

type B With the regression data, the process consisted of transforming some attributes into labels. A selected set of attributes are normalized and then ranked by descent order for each example. The remaining attributes will then be used to predict the rankings. As some of the attributes are correlated, this transformation is believed to keep a relation between predictors and rankings.

While the type A rankings can be interpreted as the preferences of a classifier, namely the naive Bayes, the interpretation on type B is not so clear. As mentioned in [26], type B datasets lead to more difficult learning problems. In this work, we analyze both data types.

6.3 Swap Randomization

Swap randomization consists of the creation of randomized datasets $\{D'_i\}_{i=1,\ldots,s}$ from a given dataset D to compare and validate the findings of data mining algorithms. We can maintain the margins of the attributes of D in all $\{D'_i\}_{i=1,\ldots,s}$ by swapping the position of the values per attribute (see Figure 6.1).

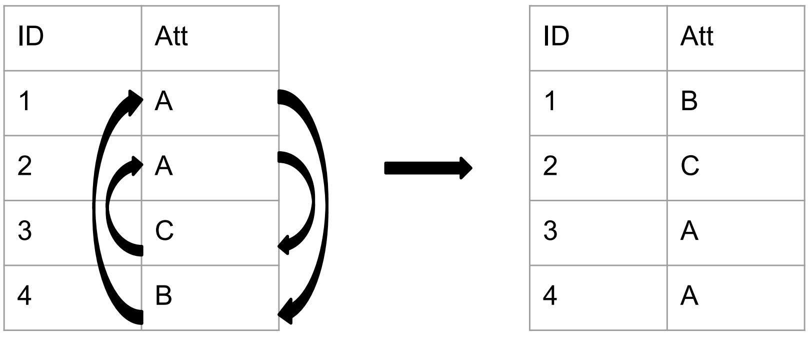


Figure 6.1: Illustration of a swap randomization per attribute.

Given an interest measure a_i, for example the accuracy of a learning method, an estimation of $\{a_i\}_{i=1,\ldots,s}$ can be obtained for $\{D'_i\}_{i=1,\ldots,s}$, respectively. Considering a_D as the estimation of a in the dataset D by a given method, if a_D deviates significantly from the distribution of $\{a_i\}_{i=1,\ldots,s}$ we can consider a_D to be significant, otherwise we do not consider it to be relevant [62].

The same concept has also been widely used in the classification task to validate classifiers [63, 103], and is commonly referred to as the *permutation test*. The *p-value* can be seen as the fraction of $\{D'_i\}_{i=1,\ldots,s}$ where the classifiers obtained better results than in D. In other words, this procedure measures to what extent the accuracy of classifiers could have been due to chance [105]. The null hypothesis assumes that there is no relation between the independent variables and the targets.

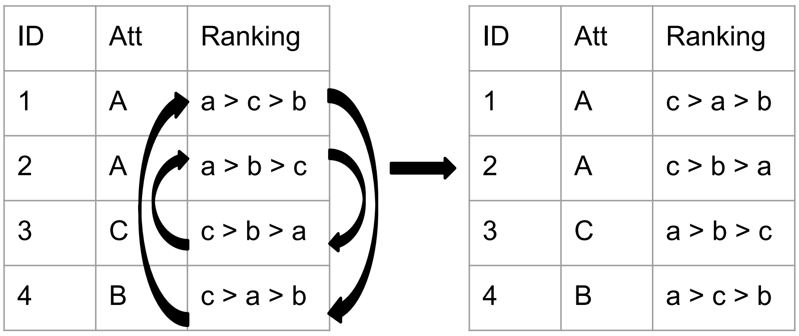
If we reverse the interpretation, we can also use learning methods to assess the information contained in the datasets. By using more than one learner we can avoid the bias of the methods.

6.4 Validating ranking data with permutation tests

Swap Randomization is used to verify the significance of Data Mining discoveries from any given method [62]. If we use the same concept and randomly permute the position of the target attribute relatively to the independent variables, we should be able to verify if the relation attribute-target is also meaningful. While the target class has only one dimension, the target ranking has k dimensions. This property allows us to make partial permutations i.e. we permute the ranks of one label while leaving the remaining ranks unchanged. The different approaches are detailed below.

6.4.1 Random permutation of rankings

Randomly permuting the rankings is a natural adaptation of the methods used in classification like in [63]. By randomly permuting the target rankings, we want to test the *strength* of the relation $\mathbb{X} \to \Omega$ in the data, as exemplified in Table 6.2. After the permutation, since we break this relation, we can measure how the LR learners behave and compare with the results on the original data. If the differences are not significant, we can conclude that there is no real relation $\mathbb{X} \to \Omega$. Otherwise, we can statistically show that the attributes-ranking relations are meaningful.



ID	Att	Ranking
1	A	a > c > b
2	A	a > b > c
3	C	c > b > a
4	B	c > a > b

ID	Att	Ranking
1	A	c > a > b
2	A	c > b > a
3	C	a > b > c
4	B	a > c > b

Figure 6.2: Illustration of a permutation of rankings.

6.4.2 Random permutation of labels

In LR the target can be seen as a multidimensional variable, from which both labelwise and pairwise levels of information can be extracted. We refer to a *labelwise permutation* when the ranks of a specific label are permuted.

By permuting one label at a time, we can assess the importance of each label by dataset. We can then compare the distributions of the permuted labels with the non-permuted results.

In [19], each attribute was permuted at a time to measure the impact of variables in prediction, in terms of misclassification rate. The results in [19] indicated that some variables did not contribute to increase the predictive power of the method used, while others were very important. Similarly, in our approach to labels, we test if similar conclusions can be drawn, but in terms of relevance of the labels of rankings.

We would like to note that this process will never lead to a completely different ranking from the original, since only the relation of one label versus the others is affected per ranking. This is exactly what we intend here, in order to test the relevance of a label at a time. The process is exemplified in Figure 6.3, where the label a is permuted within the rankings.

6.5 Experiments

We use two LR algorithms, APRIORI-LR [36] and IB-PL [24]. The performance of the methods is estimated using a ten-fold cross-validation in terms of Kendall's τ. The data for APRIORI-LR was discretized with *equal width*

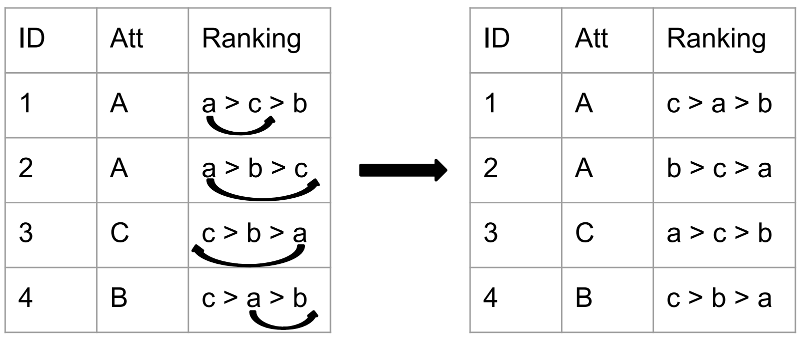


Figure 6.3: Illustration of a labelwise permutation of the label a.

discretization with 4 bins.

Table 6.1: Summary of the datasets.

Datasets	type	#examples	#labels	#attributes
bodyfat	B	252	7	7
calhousing	B	20,640	4	4
cpu-small	B	8,192	5	6
elevators	B	16,599	9	9
glass	A	214	6	9
housing	B	506	6	6
iris	A	150	3	4
segment	A	2310	7	18
stock	B	950	5	5
vehicle	A	846	4	18
vowel	A	528	11	10
wine	A	178	3	13
wisconsin	B	194	16	16

To compare the results, we used the *t.test* function from the *stats* package [113] with a confidence level of 95%. In Section 6.5.1 we use the standard *t-test* approach and in Section 6.5.2 a paired *t-test*. The p-values are mentioned below.

To check whether the mean accuracy in the original data is better or not, we use the following hypotheses:

H_0 The mean accuracy is equivalent in both original and permuted datasets.

H_1 The mean accuracy on the original datasets is better than the average accuracy on the permuted datasets.

If the p-value $< 5\%$, we reject H_0.

6.5.1 Ranking permutations

For each dataset, we performed 100 random permutations of the targets and measured the accuracy for APRIORI-LR and IB-PL. Then we compared with 10 repetitions on the original data. The distributions for IB-PL are shown in Figure 6.4.

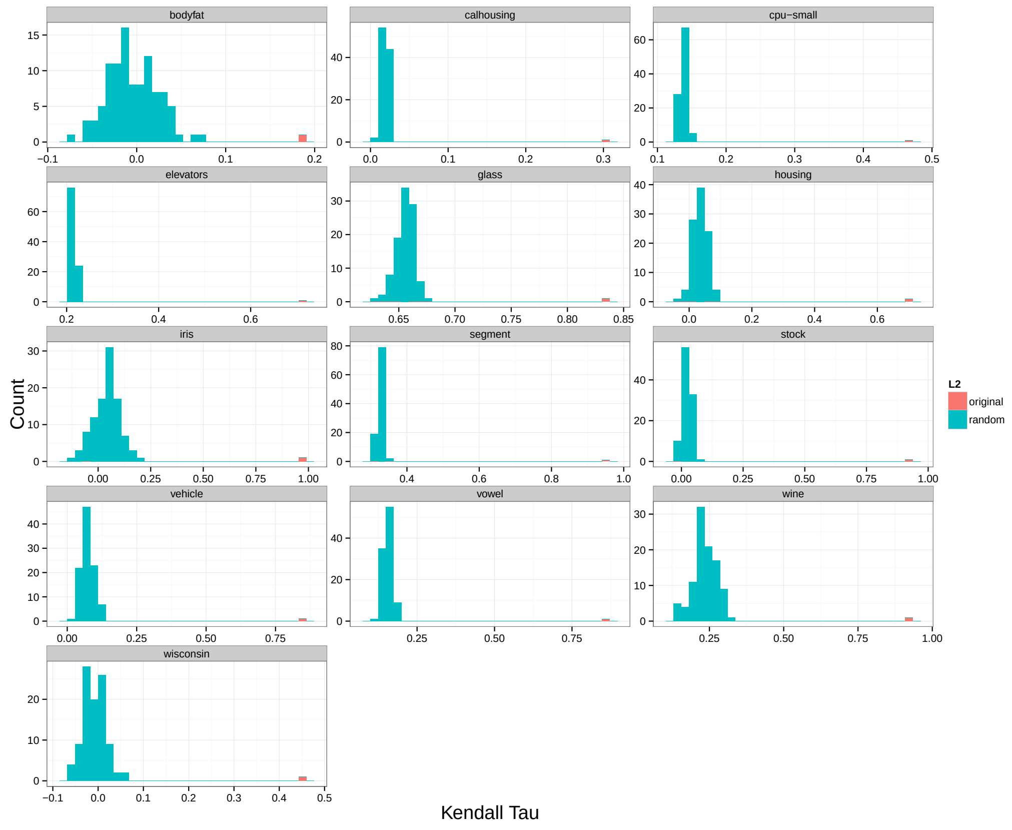


Figure 6.4: Distribution of the accuracy of IB-PL on randomized rankings (blue) and original data (red).

In Figure 6.4 it is clear that in most datasets the distribution of the accuracy on the permuted datasets is less than the accuracy with the original dataset. When the difference is big, it indicates that the algorithm is not being able to find relevant patterns in the randomized datasets. The statistical tests indicated that for all the cases, there is a significantly better mean accuracy on the original datasets, with p-values $\ll 1\%$. Very identical results were obtained using the APRIORI-LR algorithm.

6.5.2 Labelwise permutations

In this part of the experiments, we permuted one label at a time with 10 repetitions. By comparing it to the 10 repetitions on the original data, we can statistically test whether the latter are better.

Even though we used two LR methods, if we can statistically show that at least one method yielded better results with the original data than with a label permuted, then we do not need to consider the other on that particular label.

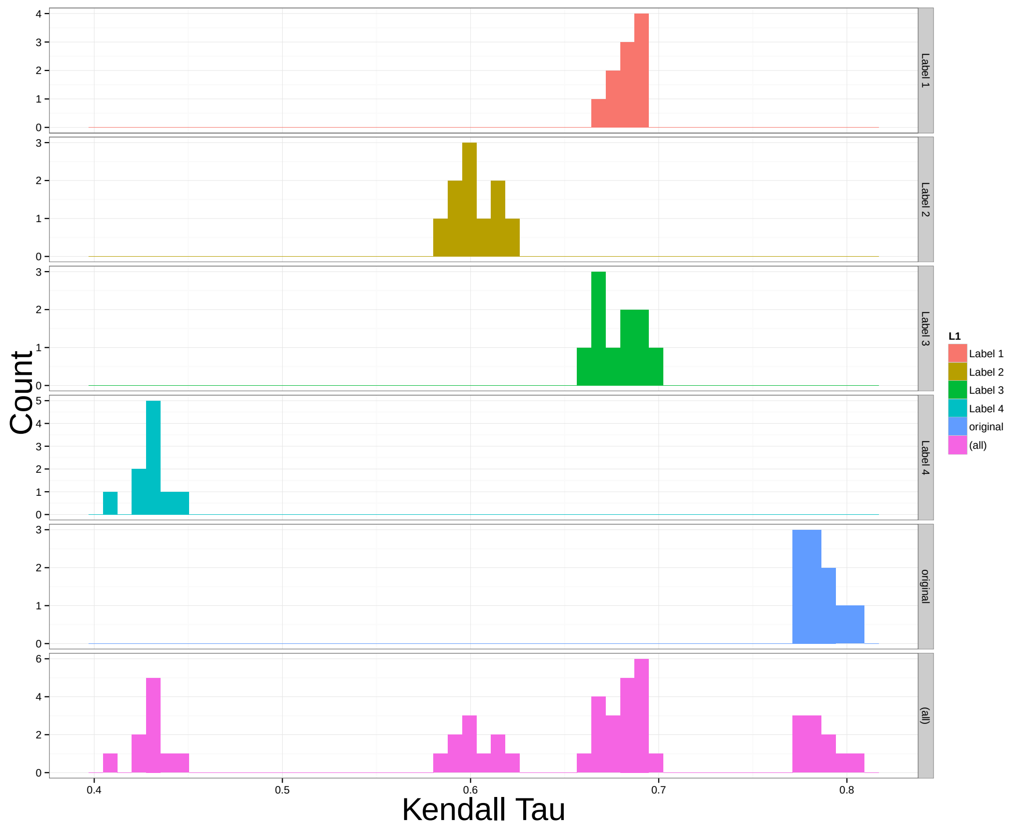


Figure 6.5: Distribution of the accuracy of APRIORI-LR on vehicle dataset per permuted label and original data.

In Figure 6.5 it is clear that the distribution of the accuracy with any label permuted is less than the accuracy on the original dataset. Therefore it seems that there are no doubts about the importance of each label for the accuracy. Also, from Figure 6.5 it is clear how label 4, when randomized, affects the accuracy in a more extreme way than the remaining labels. Statistical tests confirm that with p-values $\ll 1\%$ using both APRIORI-LR and IB-PL.

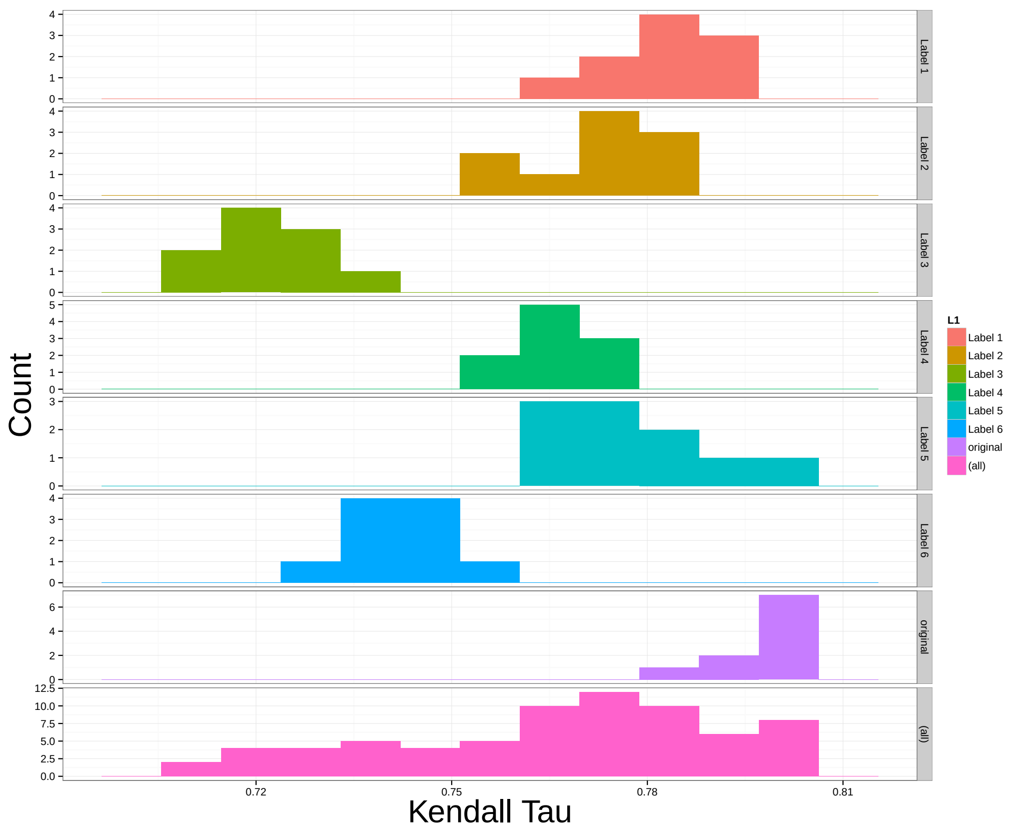


Figure 6.6: Distribution of the accuracy of APRIORI-LR on glass dataset per permuted labels and original data.

In the results obtained with the glass dataset, on Figure 6.6, the difference is not clear and the distribution of some randomized labels overlaps the distribution with the original data. However, statistical tests indicated that the distribution on the original data is significantly better.

On the other hand, it is also very interesting how labels 3 and 6 have a much higher impact on the accuracy for this model than the others. Similar to [19], we can suggest a level of relevance by label, using this approach.

Figure 6.7 gives the accuracy distribution of IB-PL per label permuted and with the original dataset. In this case, statistical tests obtained a p-value

$< 5\%$ for all the permuted labels except for label 6. Also APRIORI-LR failed to obtain a p-value $< 5\%$ for the same label.

This seems to indicate that label 6 does not have a very relevant relation with the other labels. This is somewhat expected from type B datasets rather than type A. Since in the former, some attributes were transformed into labels of a ranking, if these come from attributes that are not strongly related with the remaining, the label rank should also be measured as irrelevant.

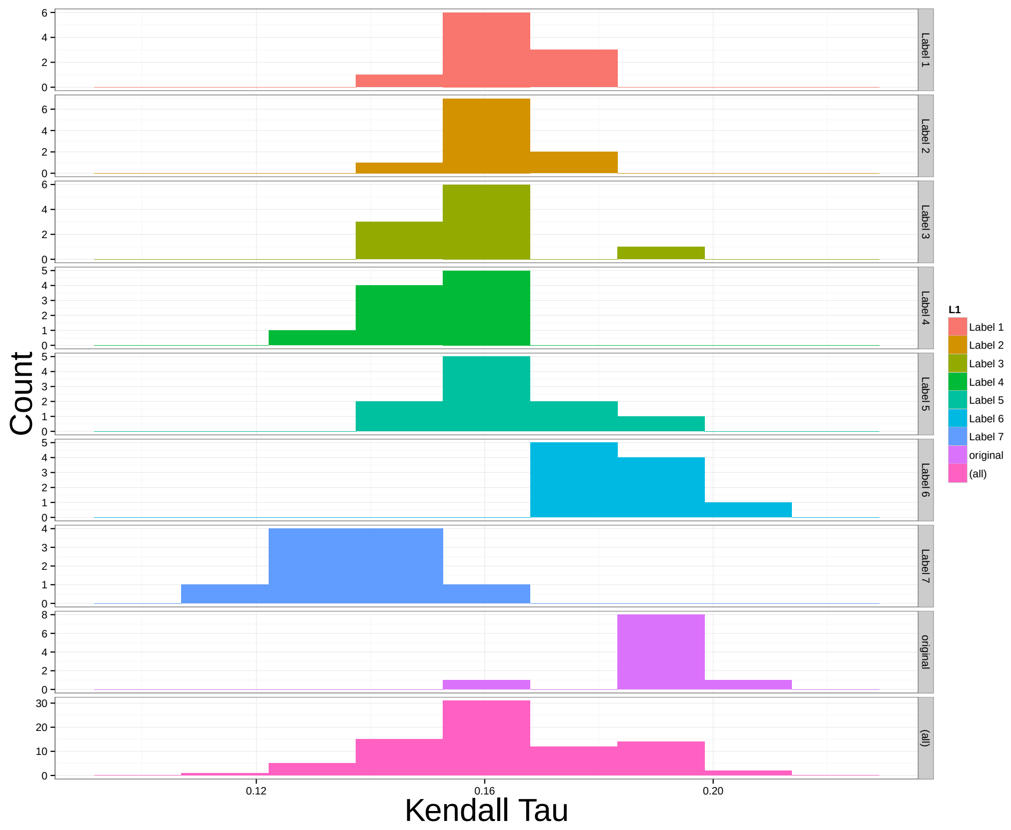


Figure 6.7: Distribution of the accuracy of IB-PL on the bodyfat dataset per permuted label and original data.

Due to space limitations we do not present results for all datasets, but instead we show the most relevant which are also representative of the others.

6.6 Conclusions

In this work, we show that, even though KEBI datasets have a semi-synthetic nature, they carry relevant preference information that can be learned by contemporary label rankers. In particular, there were no obvious differences between the type A and type B datasets. Statistical tests showed that the prediction models over this datasets are not due to chance.

This work also proposes a simple way to measure the relevance of each label on the prediction accuracy, based on the work of [19]. We also found out that some labels seem to affect the accuracy more than other, such as in the *glass* and *vehicle* dataset.

This methods can also be used on real world datasets too, in order to give a richer analysis. For example, by measuring the relative importance of each label or determining which algorithm is more resistant to *noise in rankings* [39]. In the future, we intend to propose a specific method to assess the relevance of ranking data with a proper statistical framework.

Chapter 7

Conclusions

In this thesis, we addressed label ranking problems with popular data mining techniques. In most cases, typical data mining approaches had to be adapted to better explore the complexity of the object of study, the rankings. Ranking are objects with multiple dimensions. Hence, one challenge is to define the border between similar and distinct rankings (Chapter 2). On the other hand, this multi-dimensionality allowed us to explore different facets of the rankings (Chapter 5 and Chapter 6).

We proposed methods that are either direct or reduction techniques. Considering the results obtained, we believe that direct and reduction approaches complement each other by providing different perspectives of the label ranking problem (Chapter 5).

Whenever applicable, we compared our findings with the state-of-the-art label ranking approaches. The good results obtained demonstrate that the proposed approaches are meaningful and competitive. In particular, the adaptation of one popular approach for classification and regression tasks, Random Forests, led to a highly competitive label ranking method (Chapter 4).

Label Ranking Association Rules were proposed as a predictive approach for label ranking [36]. In Chapter 2, we consolidated the work on Label Ranking Association Rules and presented an extensive empirical analysis of its behavior. The performance was analyzed from different perspectives, such as *accuracy*, *number of rules* and *average confidence*. The results show that, for label ranking datasets in general, similarity-based interest measures contribute positively to the accuracy of the model. Results also seem to indicate that the higher the entropy of the rankings on a dataset, the more the accuracy can be affected by the similarity threshold. This can be used as an indicator for setting the threshold according to the characteristics of the data.

In Chapter 3, we proposed two *supervised* discretization approaches for label ranking. The two methods, based on a well-known supervised discretization approach for classification, are referred to as *Minimum Description Length Partition for Ranking* (MDLP-R) and Entropy-based Discretization for Ranking (EDiRa). Both use different heuristic measures of entropy, based on the *Shannon entropy* [54], to discretize numeric variables.

An analysis of MDLP-R was performed in terms of the similarity threshold parameter. It was clear that, in simple scenarios, MDLP-R deals with noisy ranking data appropriately and that the threshold plays a major role in its behavior. When there are only a few distinct rankings in the data, the method can be less sensitive to the ranking similarities. We also observed that, in more complex situations, MDLP-R tends to overfit the data.

For comparison, a supervised discretization method for classification was also used, recurring to a Ranking As Class transformation [39]. Hence, the original MDLP discretization method for classification was also used in label ranking problems. However, as expected the latter failed to distinguish very similar, but not equal, rankings [39]. This RAC transformation also comes with the problem that, the number of classes can be extremely large, up to a maximum of $k!$, where k is the number of labels in $\mathcal{L}$.

Concerning the second method proposed, EDiRa, the experiments indicate that this is a more stable and efficient method when compared to MDLP-R. An analysis of EDiRa shows that it clearly outperforms MDLP-R and does not have the problem of overfitting, in the presence of noisy ranking data, as its predecessor. The proposed supervised discretization approaches can motivate the creation of new methods that, otherwise, could not deal with continuous data.

The measure of entropy used in EDiRa is more simple and showed better sensibility to ranking than the previous one. We also believe that, despite its heuristic nature, makes sense and may be more generally useful in label ranking. Furthermore, it can be also applied to other fields (e.g. regression) since it is based on a distance measure such as Kendall τ.

In Chapter 4, this measure of entropy was implemented in the splitting process of a decision tree, giving rise to a novel ranking tree approach, Entropy Ranking Trees. We also implemented and analyzed an improved version of Ranking Trees [115].

An analysis of the behavior concluded that both are valid and competitive approaches. In general, Entropy Ranking Trees generated trees with much smaller depth than Ranking Trees. On the other hand, Ranking Trees had better accuracy. Statistical tests showed that none of the methods is significantly different from the state-of-the-art approach, Label Ranking Trees [26].

As a natural extension of this work, and considering the success of Random Forests for classification and regression tasks [13], we proposed Label Ranking Forests in Chapter 4. Two versions were proposed. One approach used Ranking Trees as base model and the other used Entropy Ranking Trees. We observed a clear improvement of the accuracy in comparison to the corresponding base methods. The results confirmed that Label Ranking Forests are highly competitive label ranking methods.

In Chapter 5 we introduced Exceptional Preferences Mining for mining label ranking data. It consists of a supervised local pattern mining task where the target concept is a ranking of a fixed set of labels. The result of this task is a set of subgroups, described as a conjunction of conditions, where the label preferences are exceptional in some sense. Three quality measures were developed to measure different kinds of exceptionality in preferences, *Pairwise*, *Labelwise* and *Norm*. A discussion of the relative merits, drawbacks, and foci of the quality measures was provided, including guidelines regarding when to use which measure.

One of the main benefits of a local pattern mining method such as Exceptional Preferences Mining is that it delivers interpretable results. That means that the resulting subgroups are ideally suited to instigate real-world policies and action. In particular, the experiments on the Algae and Sushi datasets provided a valuable exploration of the data with interpretable results. In terms of visualization of rankings, the Preference Matrix visualization was able to reveal information that was not easy to obtain with the usual representations of rankings.

In Chapter 2, we proposed Pairwise Association Rules (PAR) as a decomposition method for mining label ranking datasets. Pairwise Association Rules successfully found interesting subranking patterns in both the *Algae* and *Sushi* datasets. The results clearly show the potential of this relaxed approach that finds subsets of data for which, some parts of rankings are frequently observed. This approach is more relaxed than Label Ranking Association Rules, in the sense that it does not force to find complete rankings. In future research, Pairwise Association Rules could also be used for predictive tasks.

In Chapter 6, we investigated the usefulness of the type B datasets from the KEBI repository, and proposed two *swap randomization* methods specifically for label ranking datasets. As in [62], we used statistical tests to validate the significance of the findings.

We conclude that, even though KEBI datasets have a semi-synthetic nature, they carry relevant preference information that can be learned by contemporary label rankers. In particular, there were no obvious differences between the type A and type B datasets.

As a side note, one minor contribution was the adoption of the Algae dataset (Chapter 5). This dataset, originally for multi-regression problems (referred as COIL 1999 Competition Data [96]), was approached in this thesis from a label ranking perspective. Here, the set of frequencies of the algae were interpreted as rankings. This led to a different approach of the problem, where we want to understand in which conditions some algae prevail and others not.

We proposed *Preference Rules*, as a generic term of association rules for mining ranking data. Label Ranking Association Rules and Pairwise Association Rules can be regarded as specialization of general association rules that handle ranking data. We strongly believe that such a distinction is important to emphasize the complexity of the rankings, in comparison to other type of targets [57].

As future work, we believe that PAR have potential to be used as predictive models. However, a straightforward implementation might not give satisfactory results since pairwise conflicts can appear (e.g. $A \rightarrow \lambda_1 \succ \lambda_2 \wedge \lambda_2 \succ \lambda_1$). For this, proper aggregation techniques must be used. Also, giving the unusual structure of PAR, with multiple items in the consequent, appropriate interest measures can be developed to handle this type of rules [8].

In our opinion, Exceptional Preferences Mining, can be useful in other fields, other than the ones explored in this work (Chapter 5). For example, in the discovery of profiles with same voting trends. Also, as we broaden the scope of Exceptional Preferences Mining, more quality measures can be developed to better suit the problems at hand.

List of Acronyms

DM	Data Mining
AR	Association Rules
LR	Label Ranking
LRAR	Label Ranking Association Rules
PAR	Pairwise Association Rules
MDLP	Minimum Description Length Principle
MDLP-R	Minimum Description Length Principle for Ranking data
EDiRa	Entropy-based Discretization for Ranking data
RAC	Ranking As Class

Bibliography

[1] T. Abudawood and P. A. Flach. Evaluation measures for multi-class subgroup discovery. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part I*, pages 35–50, 2009.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993.*, pages 207–216, 1993.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499, 1994.

[5] A. Agresti. *Analysis of ordinal categorical data.* Wiley series in probability and mathematical statistics. J. Wiley, Hoboken, 2010.

[6] A. Aiguzhinov, C. Soares, and A. P. Serra. A similarity-based adaptation of naive bayes for label ranking: Application to the metalearning problem of algorithm recommendation. In *Discovery Science - 13th International Conference, DS 2010, Canberra, Australia, October 6-8, 2010. Proceedings*, pages 16–26, 2010.

[7] R. Arens. Learning SVM ranking functions from user feedback using document metadata and active learning in the biomedical domain. In *Preference Learning.*, pages 363–383. Springer, 2010.

[8] M. Z. Ashrafi, D. Taniar, and K. A. Smith. Redundant association rules reduction techniques. In *AI 2005: Advances in Artificial In-*

telligence, 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, December 5-9, 2005, Proceedings, pages 254–263, 2005.

[9] P. J. Azevedo and A. M. Jorge. Comparing rule measures for predictive association rules. In *Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007, Proceedings*, pages 510–517, 2007.

[10] P. J. Azevedo and A. M. Jorge. Ensembles of jittered association rule classifiers. *Data Min. Knowl. Discov.*, 21(1):91–129, 2010.

[11] P. L. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9:1823–1840, 2008.

[12] S. D. Bay. Multivariate discretization for set mining. *Knowl. Inf. Syst.*, 3(4):491–512, 2001.

[13] G. Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13:1063–1095, 2012.

[14] R. I. Brafman. Preferences, planning and control. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, pages 2–5, 2008.

[15] F. Brandenburg, A. Gleißner, and A. Hofmeier. Comparing and aggregating partial orders with kendall tau distances. *Discrete Math., Alg. and Appl.*, 5(2), 2013.

[16] P. Brazdil and C. Soares. Exploiting Past Experience in Ranking Classifiers. In H. Bacelar-Nicolau, F. C. Nicolau, and J. Janssen, editors, *Applied Stochastic Models and Data Analysis*, pages 299–304. Instituto Nacional de Estatística, 1999.

[17] P. Brazdil, C. Soares, and J. P. da Costa. Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, 2003.

[18] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[19] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

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

[21] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA.*, pages 255–264, 1997.

[22] J. Cerquides and R. L. de Mántaras. Proposal and empirical comparison of a parallelizable distance-based discretization method. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997*, pages 139–142, 1997.

[23] W. Cheng. *Label Ranking with Probabilistic Models.* PhD dissertation, Philipps-Universität Marburg Fachbereich Mathematik und Informatik, 2012.

[24] W. Cheng, K. Dembczynski, and E. Hüllermeier. Label ranking methods based on the plackett-luce model. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 215–222, 2010.

[25] W. Cheng, S. Henzgen, and E. Hüllermeier. Labelwise versus pairwise decomposition in label ranking. In *LWA 2013. Lernen, Wissen & Adaptivität, Workshop Proceedings Bamberg, 7.-9. October 2013*, pages 129–136, 2013.

[26] W. Cheng, J. C. Huhn, and E. Hüllermeier. Decision tree and instance-based learning for label ranking. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 161–168, 2009.

[27] W. Cheng and E. Hüllermeier. Label ranking with abstention: Predicting partial orders by thresholding probability distributions (extended abstract). *Computing Research Repository, CoRR*, abs/1112.0508, 2011.

[28] W. Cheng, E. Hüllermeier, W. Waegeman, and V. Welker. Label ranking with partial abstention based on thresholded probabilistic models. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2510–2518, 2012.

[29] D. K. Y. Chiu, B. Cheung, and A. K. C. Wong. Information synthesis based on hierarchical maximum entropy discretization. *J. Exp. Theor. Artif. Intell.*, 2(2):117–129, 1990.

[30] S. Clémençon, M. Depecker, and N. Vayatis. Ranking forests. *Journal of Machine Learning Research*, 14(1):39–73, 2013.

[31] J. C. de Borda. Mémoire sur les élections au scrutin. 1781.

[32] C. R. de Sá, C. Soares, A. Knobbe, and P. Cortez. Label ranking forests. *Expert Systems*, pages n/a–n/a, 2016.

[33] C. R. de Sá. Mining association rules for label ranking. Master's thesis, Faculty of Sciences, University of Porto, 2010.

[34] C. R. de Sá, W. Duivesteijn, C. Soares, and A. J. Knobbe. Exceptional preferences mining. In *Discovery Science - 19th International Conference, DS 2016, Bari, Italy, October 19-21, 2016, Proceedings*, pages 3–18, 2016.

[35] C. R. de Sá, C. Rebelo, C. Soares, and A. J. Knobbe. Distance-based decision tree algorithms for label ranking. In *Progress in Artificial Intelligence - 17th Portuguese Conference on Artificial Intelligence, EPIA 2015, Coimbra, Portugal, September 8-11, 2015. Proceedings*, pages 525–534, 2015.

[36] C. R. de Sá, C. Soares, A. M. Jorge, P. J. Azevedo, and J. P. da Costa. Mining association rules for label ranking. In *Advances in Knowledge Discovery and Data Mining - 15th Pacific-Asia Conference, PAKDD 2011, Shenzhen, China, May 24-27, 2011, Proceedings, Part II*, pages 432–443, 2011.

[37] C. R. de Sá, C. Soares, A. M. Jorge, P. J. Azevedo, and A. Knobbe. Preference rules. *submitted to Information Fusion Journal*, 2017.

[38] C. R. de Sá, C. Soares, and A. Knobbe. Permutation tests for label ranking. In *Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC 2015)*, 2015.

[39] C. R. de Sá, C. Soares, and A. J. Knobbe. Entropy-based discretization methods for ranking data. *Inf. Sci.*, 329:921–936, 2016.

[40] C. R. de Sá, C. Soares, A. J. Knobbe, P. J. Azevedo, and A. M. Jorge. Multi-interval discretization of continuous attributes for label ranking. In *Discovery Science - 16th International Conference, DS 2013, Singapore, October 6-9, 2013. Proceedings*, pages 155–169, 2013.

[41] O. Dekel, C. D. Manning, and Y. Singer. Log-linear models for label ranking. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 497–504, 2003.

[42] K. Dembczynski, W. Kotlowski, R. Slowinski, and M. Szelag. Learning of rule ensembles for multiple attribute ranking problems. In *Preference Learning.*, pages 217–247. Springer, 2010.

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

[44] S. Destercke. A pairwise label ranking method with imprecise scores and partial predictions. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part II*, pages 112–127, 2013.

[45] A. Dinno. *dunn.test: Dunn's Test of Multiple Comparisons Using Rank Sums*, 2015. R package version 1.2.3.

[46] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 194–202, 1995.

[47] J. Doyle. Prospects for preferences. *Computational Intelligence*, 20(2):111–136, 2004.

[48] W. Duivesteijn. *Exceptional Model Mining.* PhD thesis, Leiden University, 2013.

[49] W. Duivesteijn, A. Feelders, and A. J. Knobbe. Exceptional model mining - supervised descriptive local pattern mining with complex target concepts. *Data Min. Knowl. Discov.*, 30(1):47–98, 2016.

[50] W. Duivesteijn and A. J. Knobbe. Exploiting false discoveries - statistical validation of patterns and quality measures in subgroup discovery. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 151–160, 2011.

[51] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 613–622, 2001.

[52] V. Dzyuba and M. van Leeuwen. Interactive discovery of interesting subgroup sets. In *Advances in Intelligent Data Analysis XII - 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings*, pages 150–161, 2013.

[53] T. Elomaa and J. Rousu. Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. *Data Min. Knowl. Discov.*, 8(2):97–126, 2004.

[54] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1022–1029, 1993.

[55] J. C. Fodor and M. R. Roubens. *Fuzzy preference modelling and multicriteria decision support*, volume 14 of *Theory and Decision Library D:*. Springer Netherlands, Dordrecht, 1994.

[56] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *Machine Learning: ECML 2003, 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, pages 145–156, 2003.

[57] J. Fürnkranz and E. Hüllermeier, editors. *Preference Learning*. Springer, 2010.

[58] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, 2008.

[59] J. Fürnkranz, E. Hüllermeier, and S. Vanderlooy. Binary decomposition methods for multipartite ranking. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part I*, pages 359–374, 2009.

[60] S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Trans. Knowl. Data Eng.*, 25(4):734–750, 2013.

[61] R. Genuer, J. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010.

[62] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *TKDD*, 1(3), 2007.

[63] P. Golland, F. Liang, S. Mukherjee, and D. Panchenko. Permutation tests for classification. In *Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings*, pages 501–515, 2005.

[64] M. Gurrieri, X. Siebert, P. Fortemps, S. Greco, and R. Slowinski. Label ranking: A new rule-based label ranking method. In *Advances on Computational Intelligence - 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2012, Catania, Italy, July 9-13, 2012. Proceedings, Part I*, pages 613–623, 2012.

[65] M. Halkidi and M. Vazirgiannis. Quality assessment approaches in data mining. In *The Data Mining and Knowledge Discovery Handbook.*, pages 661–696. Springer, 2005.

[66] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2000.

[67] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.

[68] S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: A new approach to multiclass classification. In *Algorithmic Learning Theory, 13th International Conference, ALT 2002, Lübeck, Germany, November 24-26, 2002, Proceedings*, pages 365–379, 2002.

[69] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, chapter Unsupervised Learning, pages 485–585. Springer New York, New York, NY, 2009.

[70] S. Henzgen and E. Hüllermeier. Mining rank data. In *Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, pages 123–134, 2014.

[71] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining - A general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, 2000.

[72] K. M. Ho and P. D. Scott. Zeta: A global method for discretization of continuous variables. In *Proceedings of the Third International Con-*

ference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997, pages 191–194, 1997.

[73] W. Huang, Y. Pan, and J. Wu. Supervised discretization for optimal prediction. *Procedia Computer Science*, 30:75 – 80, 2014.

[74] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artif. Intell.*, 172(16-17):1897–1916, 2008.

[75] M. Iqbal, I. Mukhlash, and H. M. Astuti. The comparison of cba algorithm and cbs algorithm for meteorological data classification. *ISICO 2013*, 2013.

[76] F. Jiang and Y. Sui. A novel approach for discretization of continuous attributes in rough set theory. *Knowl.-Based Syst.*, 73:324–334, 2015.

[77] N. Jin, P. A. Flach, T. Wilcox, R. Sellman, J. Thumim, and A. J. Knobbe. Subgroup discovery in smart electricity meter data. *IEEE Trans. Industrial Informatics*, 10(2):1327–1336, 2014.

[78] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142, 2002.

[79] A. M. Jorge, F. Pereira, and P. J. Azevedo. Visual interactive subgroup discovery with numerical properties of interest. In *Discovery Science, 9th International Conference, DS 2006, Barcelona, Spain, October 7-10, 2006, Proceedings*, pages 301–305, 2006.

[80] R. J. B. Jr., R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Min. Knowl. Discov.*, 4(2/3):217–240, 2000.

[81] T. Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 583–588, 2003.

[82] T. Kamishima, H. Kazawa, and S. Akaho. A survey and empirical comparison of object ranking methods. In *Preference Learning.*, pages 181–201. Springer, 2010.

[83] A. Karatzoglou and M. Weimer. Collaborative preference learning. In *Preference Learning.*, pages 409–427. Springer, 2010.

[84] J. Kemeny and J. Snell. *Mathematical Models in the Social Sciences.* MIT Press, 1972.

[85] M. Kendall and J. Gibbons. *Rank correlation methods.* Griffin London, 1970.

[86] R. Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992.*, pages 123–128, 1992.

[87] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. American Association for Artificial Intelligence, 1996.

[88] W. Klösgen and J. M. Zytkow, editors. *Handbook of Data Mining and Knowledge Discovery.* Oxford University Press, New York, NY, USA, 2002.

[89] W. Kotlowski, K. Dembczynski, and E. Hüllermeier. Bipartite ranking through minimization of univariate loss. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1113–1120, 2011.

[90] S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

[91] G. Lebanon and J. D. Lafferty. Conditional models on the ranking poset. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 415–422, 2002.

[92] M. D. Lee, M. Steyvers, and B. Miller. A cognitive model for aggregating people's rankings. *Public Library of Science, PLOS ONE*, 9(5):e96431, 2014.

[93] W. H. K. Leo A. Goodman. Measures of association for cross classifications. *Journal of the American Statistical Association*, 49(268):732–764, 1954.

[94] B. Letham, T. H. Mccormick, C. Rudin, and D. Madigan. Building interpretable classifiers with rules using bayesian analysis, 2012.

[95] W. Li, J. Han, and J. Pei. CMAR: accurate and efficient classification based on multiple class-association rules. In *Proceedings of the*

2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA, pages 369–376, 2001.

[96] M. Lichman. UCI machine learning repository, 2013.

[97] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York City, New York, USA, August 27-31, 1998*, pages 80–86, 1998.

[98] B. Liu, W. Hsu, and Y. Ma. Mining association rules with multiple minimum supports. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999*, pages 337–341, 1999.

[99] H. Liu and R. Setiono. Feature selection via discretization. *IEEE Trans. Knowl. Data Eng.*, 9(4):642–645, 1997.

[100] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258, 1997.

[101] M. Meeng and A. Knobbe. Flexible enrichment with cortana–software demo. In *Proceedings of BeneLearn*, pages 117–119, 2011.

[102] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[103] A. M. Molinaro, R. Simon, and R. M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 2005.

[104] H. Neave and P. Worthington. *Distribution-free Tests*. Routledge, 1992.

[105] M. Ojala and G. C. Garriga. Permutation tests for studying classifier performance. *Journal of Machine Learning Research*, 11:1833–1863, 2010.

[106] E. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Trans. Knowl. Data Eng.*, 15(1):57–69, 2003.

[107] J. S. Park, M. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995.*, pages 175–186, 1995.

[108] J. S. Park, M. Chen, and P. S. Yu. Efficient parallel and data mining for association rules. In *CIKM '95, Proceedings of the 1995 International*

Conference on Information and Knowledge Management, November 28 - December 2, 1995, Baltimore, Maryland, USA, pages 31–36, 1995.

[109] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent item sets with convertible constraints. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 433–442, 2001.

[110] J. Pinto da Costa and C. Soares. A weighted rank measure of correlation. *Australian and New Zealand Journal of Statistics*, 47(4):515–529, 2005.

[111] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[112] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[113] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2015.

[114] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 239–248, 2005.

[115] C. Rebelo, C. Soares, and J. Costa. Empirical Evaluation of Ranking Trees on Some Metalearning Problems. In J. Chomicki, V. Conitzer, U. Junker, and P. Perny, editors, *Proceedings 4th AAAI Multidisciplinary Workshop on Advances in Preference Handling*, 2008.

[116] G. Ribeiro, W. Duivesteijn, C. Soares, and A. J. Knobbe. Multilayer perceptron for label ranking. In *Artificial Neural Networks and Machine Learning - ICANN 2012 - 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 11-14, 2012, Proceedings, Part II*, pages 25–32, 2012.

[117] E. Scornet, G. Biau, and J.-P. Vert. Consistency of random forests. *ArXiv e-prints*, May 2014.

[118] C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:72–101, 1904.

[119] S. Thomas and S. Sarawagi. Mining generalized association rules and sequential patterns using SQL queries. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*

(KDD-98), New York City, New York, USA, August 27-31, 1998, pages 344–348, 1998.

[120] L. Todorovski, H. Blockeel, and S. Dzeroski. Ranking with predictive clustering trees. In *Machine Learning: ECML 2002, 13th European Conference on Machine Learning, Helsinki, Finland, August 19-23, 2002, Proceedings*, pages 444–455, 2002.

[121] L. Umek and B. Zupan. Subgroup discovery in data sets with multi-dimensional responses. *Intell. Data Anal.*, 15(4):533–549, 2011.

[122] T. L. Van, M. van Leeuwen, S. Nijssen, A. C. Fierro, K. Marchal, and L. D. Raedt. Ranked tiling. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, pages 98–113, 2014.

[123] S. Vembu and T. Gärtner. Label ranking algorithms: A survey. In *Preference Learning.*, pages 45–64. Springer, 2010.

[124] F. L. Wauthier, M. I. Jordan, and N. Jojic. Efficient ranking from pairwise comparisons. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 109–117, 2013.

[125] G. I. Webb. Discovering significant rules. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 434–443, 2006.

[126] S. Yasutake, K. Hatano, E. Takimoto, and M. Takeda. Online rank aggregation. In *Proceedings of the 4th Asian Conference on Machine Learning, ACML 2012, Singapore, Singapore, November 4-6, 2012*, pages 539–553, 2012.

[127] Y. Zhou, Y. Liu, X. Z. Gao, and G. Qiu. A label ranking method based on gaussian mixture model. *Knowl.-Based Syst.*, 72:108–113, 2014.

Nederlandse Samenvatting

Voorkeuren hebben altijd een rol gespeeld in ons dagelijks leven. Het kopen van de juiste auto, het kiezen van een geschikt huis, en zelfs beslissen wat te eten, zijn enkele triviale voorbeelden van keuzes die expliciet of impliciet iets vertellen over onze voorkeuren. De recente trend om alsmaar groeiende hoeveelheden data te verzamelen geldt ook voor data over voorkeuren.

Het extraheren en modelleren van voorkeuren levert ons waardevolle informatie over de keuzes van groepen en individuen. In gebieden als e-commerce, die typisch gaan over de keuzes van duizenden mensen, kan het vastleggen van voorkeuren een moeilijke taak zijn. Om deze redenen zijn methoden uit de kunstmatige intelligentie (specifieker, het machinaal leren) van toenemend belang voor het ontdekken en automatisch leren van modellen over voorkeuren.

Het deelgebied van machinaal leren dat zich richt op de studie en het modelleren van voorkeuren staat bekend als *Preference Learning* (PL). We kijken specifiek naar een deeltaak binnen PL, namelijk Label Ranking (LR). Kort gezegd, een LR dataset bestaat uit een verzameling observaties, beschreven door attributen (de onafhankelijk variabelen), en een ordening over een (eindige) verzameling labels (de afhankelijke variabele). In LR zijn we geïnteresseerd in het voorspellen van de ordening van de labels, gebaseerd op de waarden van de onafhankelijk variabelen.

In dit promotieproject zijn meerdere aanpakken voor het LR-probleem voorgesteld en geanalyseerd. We onderzochten Label Ranking Association Rules (LRAR), wat het LR-equivalent is van de Class Association Rules. Een LRAR is een associatieregel waarvan de items gebaseerd zijn op waarden voor de onafhankelijke variabelen, en de conclusie van de regel een ordening over de labels voorstelt. Verder stelden we de zogenaamde Pairwise Association Rules (PAR) voor, die gedefinieerd zijn als associatieregels waarvan de conclusie een verzameling van paarsgewijze voorkeuren weergeeft. PAR kunnen, net als LRAR, zowel beschrijvend als voorspellend gebruikt worden.

Onze analyse is echter gericht op de beschrijvende eigenschappen, terwijl de LRAR vooral gemodelleerd zijn als voorspellende modellen.

Het is algemeen bekend dat preprocessing (het voorbewerken van data) een belangrijke component is van machinaal leren. Dit is net zo zeer het geval voor LR als voor elke andere machinaal leren taak. Om een voorbeeld te noemen: LRAR, net als associatieregels, kunnen niet direct met numerieke data omgaan, zodat de data vooraf gediscretiseerd moet worden. Er bestond echter nog niet zo'n methode voor het geval van LR. Om die reden stelden we twee LR-specifieke discretisatiemethoden voor. Beide methoden zijn gebaseerd op nieuwe maten voor de entropie van ordeningen (ranking entropy).

Het grootste deel van dit project is gericht op methoden voor het ontdekken van patronen. Echter, gezien de de populariteit van beslisboommethoden en hoe deze methoden inzichtelijk informatie over de taak kunnen weergeven, introduceerden we Entropy Ranking Trees. Hoewel eerdere methoden bestonden voor het aanpassen van beslisboomalgoritmen aan LR, was het natuurlijk om vanuit een maat voor de entropie van ordeningen te onderzoeken hoe deze in deze algoritmen geïntegreerd konden worden. Een andere erg populaire modelleeraanpak is die van de ensemble learning. Specifiek het Random Forest (willekeurige woud) algoritme is zeer succesvol gebleken, maar was nog nooit aangepast naar LR. Het betreft hier een ensemble-methode die verschillende bomen combineert die op willekeurige manier verkregen zijn. We stelden dus een ensemble voor Label Ranking voor die gebaseerd is op Random Forest, genaamd Label Ranking Forest.

Onze zoektocht door het veld van de preference learning werd voortgezet met het ontdekken van lokale patronen. De taak heet Exceptional Pattern Mining (het ontdekken van uitzonderlijke patronen) en kan gezien worden als het ontdekken van lokale patronen over deelverzamelingen van de observaties waarvan de voorkeursverhoudingen tussen een deel van de labels afwijken van de norm. In andere woorden, het is een variant op de zogenaamde Subgroup Discovery taak, waarbij de doelvariabele een ordening betreft. We gebruiken drie kwaliteitsmaten die subgroepen benadrukken die uitzonderlijke voorkeuren vertonen, waarbij de specifieke nadruk wat betreft 'uitzonderlijk' verschilt per maat. De resultaten tonen ook aan hoe de visualisatie van voorkeuren in een voorkeursmatrix (Preference Matrix) kan helpen bij het interpreteren van subgroepen van uitzonderlijke voorkeuren.

Als laatste suggereerden we een aanpak voor het testen van de relatie tussen ordeningen en de onafhankelijke variabelen in een LR dataset. Zoals in andere leertaken met toezicht (supervised learning) is randomiseren van de doelvariabele door middel van omwisselen (swap randomisation) gebruikt om deze test te doen. We stelden twee manieren voor om dit te doen voor LR, en hebben ze toegepast op LR datasets.

De experimentele resultaten tonen het potentieel van de genoemde aanpakken.

English Summary

Preferences have always been present in many tasks in our daily lives. Buying the right car, choosing a suitable house or even deciding on the food to eat, are trivial examples of decisions that reveal information, explicitly or implicitly, about our preferences. The recent trend of collecting increasing amounts of data is also true for preference data.

Extracting and modeling preferences can provide us with invaluable information about the choices of groups or individuals. In areas like e-commerce, which typically deal with decisions from thousands of users, the acquisition of preferences can be a difficult task. For these reasons, artificial intelligence (in particular, machine learning) methods have been increasingly important to the discovery and automatic learning of models about preferences.

The subfield of machine learning which focuses on the study and modeling of preferences is *Preference Learning* (PL). We focus on one subtask of PL, Label Ranking (LR). In simple terms, a LR dataset consists of a set of observations described by attributes (independent variables) and a ranking of a (finite) set of labels (target or dependent variable). In LR, we are interested in predicting the ranking of the labels for a new observation based on the values of the independent variables.

In this Ph.D. project, several approaches were analyzed and proposed to deal with the LR problem. We investigated Label Ranking Association Rules (LRAR), which are the equivalent of Class Association Rules for the LR task. A LRAR is an association rule, where the items are based on the values of the independent values and the right-hand side is a ranking of the labels. Furthermore, we proposed Pairwise Association Rules (PAR), which are defined as association rules with a set of pairwise preferences in the consequent. Like LRAR, PAR can be used both as descriptive and predictive models. However, our analysis of PAR has focused on its descriptive properties, while LRAR have been essentially studied as predictive models.

Preprocessing methods are well known to be an essential part of machine learning processes. This is true for LR as for any other machine learning task. For example, LRARs, like association rules, cannot handle numeric data directly, which needs to be discretized beforehand. However, no LR-specific methods existed. Hence, we proposed two discretization approaches that are specific for LR problems. Both approaches are based on new measures of *ranking entropy*.

Most of this project has focused on pattern mining methods. However, given the popularity of decision tree methods and how these can clearly express information about the problem, we proposed Entropy Ranking Trees. Although previous approaches existed to adapting decision tree (DT) algorithms for LR, having proposed a measure of ranking entropy, we found it natural to investigate its integration on DT algorithms. Another very popular modeling approach is ensemble learning. In particular, the Random Forests (RF) algorithm has been very successful but was not adapted for LR. RF are an ensemble learning method that combines different trees obtained using different randomization techniques. Hence, we proposed an ensemble of decision trees for Label Ranking, based on Random Forests, which we refer to as Label Ranking Forests (LRF).

We continued our journey on the field of preference learning by combining it with local pattern mining. The task is named Exceptional Preferences Mining (EPM) and can be seen as a local pattern mining task that finds subsets of observations where the preference relations between subsets of the labels significantly deviate from the norm. In other words, it is a variant of Subgroup Discovery, with rankings as the target. We employed three quality measures that highlight subgroups featuring exceptional preferences, where the focus of what constitutes 'exceptional' varies with the measure. The results also illustrate how the visualization of the preferences in a Preference Matrix can aid in interpreting exceptional preference subgroups.

Finally, we proposed an approach to test the relation between the rankings and independent variables in LR datasets. As in other supervised learning tasks, target swap randomization methods have been used to test it. So, we proposed two target swap randomization approaches for LR and apply them on LR datasets.

Experimental results show the potential of the approaches mentioned.

Resumo

É comum lidarmos com preferências no nosso dia-a-dia. Quando compramos um carro, procuramos uma casa ou mesmo quando decidimos o que comer, estamos a tomar decisões que revelam informação sobre as nossas preferências. Nos dias que correm, cada vez mais dados são recolhidos, onde se incluem também dados sobre preferências.

A extracção e a criação de modelos de preferências, podem fornecer informações valiosas sobre determinados grupos ou indivíduos. Em áreas de negócio como o comércio electrónico, que lidam com informações de milhares de utilizadores, a modelação de preferências pode constituir um desafio. Por isso, métodos de Inteligência Artificial (em particular, *machine learning*), têm sido cada vez mais usados para a descoberta e aprendizagem automática de modelos sobre preferências.

A área de *machine learning* que lida a modelação e estudo de preferências é chamada de *Preference Learning* (PL). O tema deste doutoramento, foca em uma sub-área de PL denominada de Label Ranking (LR). Em LR, os dados consistem em observações constituídas por *atributos* (variáveis independentes) e *rankings* de um conjunto finito de objetos (*target* ou variáveis dependentes). O objectivo é prever esses rankings para novas observações, baseando-se nos valores fornecidos das variáveis independentes. Neste trabalho, foram propostas várias abordagens ao problema de LR.

Exploramos as Label Ranking Association Rules (LRAR), que são equivalentes às Class Association Rules no contexto de LR. Por definição, uma LRAR é uma *regra de associação* onde o *antecedente* é um conjunto de itens baseados nos valores das variáveis independentes, e o *consequente* é um ranking. Com uma estrutura semelhante, também propusemos as Pairwise Association Rules (PAR), definidas como regras de associação onde o consequente é um conjunto de *pairwise comparisons.* Tal como as LRAR, as PAR podem ser usadas como abordagens descritivas e como modelos de previsão. No entanto, a nossa análise foca-se nas propriedades descritivas das PAR, enquanto que as LRAR foram usadas como modelos preditivos.

Métodos de pré-processamento são uma parte essencial nos processos de *machine learning*. As LRAR, tal como regras de associação comuns, não conseguem lidar directamente com variáveis numéricas, que, por sua vez, têm que ser discretizadas à priori. Dado que não existiam métodos de discretização especificamente para dados de LR, foram propostas duas abordagens baseadas em medidas de *entropia de rankings*.

Apesar de a maior parte deste trabalho focar em métodos de *pattern mining*, tendo em conta a popularidade de métodos como *árvores de decisão* e pela forma clara como expressam informação, propusemos as *Entropy Ranking Trees*. Mesmo já existindo árvores de decisão para LR, uma vez que tinha sido proposta a medida de entropia de rankings, achamos natural estudar a sua integração neste modelos. Outra abordagem também muito popular em *machine learning* é *ensemble learning*. Nomeadamente, um algoritmo denominado *Random Forests* (RF), tem sido bem sucedido, mas nunca tinha sido adaptado para LR. O método de RF, combina vários modelos de árvores de decisão que são geradas usando algumas técnicas de randomização. Por isso, propusemos *ensembles* de árvores de decisão, baseados em RF, que chamamos de Label Ranking Forests.

Continuamos a nossa jornada na área de PL, combinando-a com técnicas de *local pattern mining*. O método, a que chamamos de Exceptional Preferences Mining (EPM), pode ser visto como uma técnica de *local pattern mining* que encontra sub-conjuntos de observações onde as preferências se desviam do normal. Por outras palavras, é uma variante de *Subgroup Discovery*, em que os rankings são o *target*. Par isso, foram propostas três medidas (*quality measures*) que procuram sub-conjuntos que apresentem preferências consideradas excepcionais. Os resultados obtidos realçam também uma forma proposta de representar preferências, a *Preference Matrix*.

Por último, apresentamos formas de testar a relação entre variáveis independentes e rankings, em dados de LR. Uma técnica denominada *target swap randomization*, também aplicada em problemas de classificação, foi implementada para este tipo de testes. Além disso, também foram propostas duas variantes, baseadas em *target swap randomization*, para se adequarem melhor ao problema.

Os resultados experimentais apresentados demonstram o potencial dos métodos aqui propostos.

List of publications

1. C. R. de Sá, C. Soares, A. M. Jorge, P. J. Azevedo, and J. P. da Costa. Mining association rules for label ranking. In *Advances in Knowledge Discovery and Data Mining - 15th Pacific-Asia Conference, PAKDD 2011, Shenzhen, China, May 24-27, 2011, Proceedings, Part II*, pages 432–443, 2011.

2. C. R. de Sá, C. Soares, A. J. Knobbe, P. J. Azevedo, and A. M. Jorge. Multi-interval discretization of continuous attributes for label ranking. In *Discovery Science - 16th International Conference, DS 2013, Singapore, October 6-9, 2013. Proceedings*, pages 155–169, 2013.

3. C. R. de Sá, C. Rebelo, C. Soares, and A. J. Knobbe. Distance-based decision tree algorithms for label ranking. In *Progress in Artificial Intelligence - 17th Portuguese Conference on Artificial Intelligence, EPIA 2015, Coimbra, Portugal, September 8-11, 2015. Proceedings*, pages 525–534, 2015.

4. C. R. de Sá, C. Soares, and A. Knobbe. Permutation tests for label ranking. In *Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC 2015)*, 2015.

5. C. R. de Sá, C. Soares, and A. J. Knobbe. Entropy-based discretization methods for ranking data. *Inf. Sci.*, 329:921–936, 2016.

6. C. R. de Sá, C. Soares, A. Knobbe, and P. Cortez. Label ranking forests. *Expert Systems*, pages n/a–n/a, 2016.

7. C. R. de Sá, W. Duivesteijn, C. Soares, and A. J. Knobbe. Exceptional preferences mining. In *Discovery Science - 19th International Conference, DS 2016, Bari, Italy, October 19-21, 2016, Proceedings*, pages 3–18, 2016.

Other publications

1. C. R. de Sá, J. Costa, C. Soares, P. Azevedo, and A. M. Jorge. Mining association rules for ordinal data classification using an unimodal model. In DSIE'2012 *Doctoral Symposium on Informatics Engineering*, page 165.

2. V. Cerqueira, F. Pinto, C. R. de Sá, and C. Soares. Combining boosted trees with metafeature engineering for predictive maintenance. In *Advances in Intelligent Data Analysis XV - 15th International Symposium, IDA 2016, Stockholm, Sweden, October 13-15, 2016, Proceedings*, pages 393–397, 2016.

Acknowledgments

Being surrounded by friends and having the friendship of colleagues is very important to me. For this reason, I would like to thank the people that made my life more pleasant during my Ph.D.:

My coworkers at LIACS. A great thank to Benjamin, Harm, Irene, Jan, Kleanthie, Liu Di, Marvin, Michael, Rafael, Ricardo, Rob, Shengfa, Ugo and Wouter, for all the great moments we shared as colleagues and friends.

My colleagues at Xidian University, in special to Zhang Lu and Leticia for being such good companions during my period in China.

My coworkers at INESC Porto. In particular to Fábio Pinto, Márcia Oliveira and Pedro Abreu for all the nice conversations and brainstorms.

I would also like to thank to all my family that someway or another encouraged me to get here. A special thanks to my parents Augusto and Filomena, my sister Jacinta and my brother Pedro, for all the support, presence and motivation in many key moments of my studies. I thank my grandfather Joaquim for taking care of me while I was young. I also thank my cousin Hugo, for all the nice moments playing mind games.

A big hug to all my friends in Portugal. In particular to Ana Soares, Filipe, Hugo, Pedro and Vasco. A special thank to João Dias, for being a true friend when I always needed.

I also thank all those friends that made me feel like home in Leiden. In special to Ana Abreu, Elena, Eugenia, Francesca, Francesco, Heather, Ines Carqueijeiro, Ines Tescione, James, José, Käthe, Lia, Luana, Luis, Manuela, Mathieu, Matteo, Maren, Marta, Miguel, Müge, Nico, Pablo, Paula, Pedro, Pelin, Pepa, Simone, Sofia, Stefano and Vicente.

Finally I would like to thank my "band" *La casa*, Dimitris, Mar and Vicente, for all the great musical moments we had together!

Curriculum Vitae

Cláudio Sá was born in Santa Maria da Feira, Portugal, on the 18th of May in 1984. From 1999 until 2002 he was a student at Escola Secundária de Santa Maria da Feira in Feira. He then studied Physics and Applied Mathematics with a major in Astronomy at Porto University, obtaining his B.Sc. degree in 2008. Part of his degree was done at the National University of Ireland in Galway, Ireland, as an Erasmus program. In 2008 he began studying a Master's programme at Porto University, graduating with a M.Sc. degree in Mathematical Engineering in December 2010. Between December 2010 and September 2011, he was working as a research assistant at the Laboratory of Artificial Intelligence and Decision Support (LIAAD) in Porto, Portugal.

In September 2011, Cláudio started his Ph.D. in Computer Science in Porto University. One year later, Cláudio started his Ph.D. in Computer Science at LIACS, the Leiden Institute of Advanced Computer Science, Leiden University. During his Ph.D., from April 2013 until July 2013, he was a visiting Ph.D. student at Xidian University in Xi'an, China. From June 2015 to September 2015 he was also working part-time as a data scientist at the Sports and Nutrition Faculty of the Hogeschool van Amsterdam.

This Ph.D. trajectory was performed under the guidance of Joost N. Kok, and with direct supervision of Carlos Soares and Arno J. Knobbe.