# Pattern mining for label ranking

Pinho Rebelo de Sá, C.F.

# 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) $\lambda$ 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 $\Omega$. In other words, a total order can be seen as a permutation $\pi$ of the set $\{1, \ldots, k\}$, such that $\pi(a)$ is the position of $\lambda_a$ in $\pi$.

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 $\Omega$ [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 $\pi$ is the ranking associated with $x$. The goal in LR is to learn the mapping $\mathbb{X} \to \Omega$. 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 $\pi_i$ is the corresponding target ranking.

Given an instance $x_i$ with label ranking $\pi_i$, and the ranking $\hat{\pi}_i$ predicted by an LR model, we evaluate the accuracy of the prediction with a loss function

on $\Omega$. 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 $\tau$ coefficient [85], which is a correlation measure where $\mathcal{D}(\pi, \pi) = 1$ and $\mathcal{D}(\pi, \pi^{-1}) = -1$ ($\pi^{-1}$ denotes the inverse order of $\pi$).

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 $\pi$.

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 $A \to \pi$ is defined as follows:

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

This essentially assigns a weight to each target $\pi_i$ in the training data, that represents its contribution to the probability that $\pi$ 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 $\tau$ or Spearman's $\rho$, can be used as $s'$. This general form assumes that below a given threshold, $\theta_{sup}$, it is not useful to discriminate between different similarity values, as they are too different from $\pi_a$ anyway.

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

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

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

As in [36], we use Kendall $\tau$ 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 $\pi$ is defined in [6] as the mean similarity between $\pi$ and all the others:

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

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

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

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

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

## 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 $S$:

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

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*

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

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

$$log_2\left(3^K - 2\right) - [K Ent\,(S) - K_1 Ent\,(S_1) - K_2 Ent\,(S_2)]$$

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 | $x^1$ | $\pi$ | $\lambda^{RAC}$ |
|---|---|---|---|
| **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 $\pi$. 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 $\lambda^{RAC}$. The natural split point identified earlier is completely undetectable in column $\lambda^{RAC}$. 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 $\pi_i$ 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 $\theta_{disc}$ is the threshold parameter of the similarity measure, equivalent to the threshold for similarity support, $\theta_{sup}$, in Equation 3.2. As in [40], we use Kendall $\tau$ 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, $\lambda_1$ is as different from $\lambda_2$ as $\lambda_2$ is from $\lambda_3$. 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 $\pi_1$ relative to sets $S_1$ and $S_2$, using Kendall $\tau$ 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 | $\pi_1$(1,2,3,4,5) | $\pi_2$(1,2,3,5,4) | $\pi_3$(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. $\pi_1$ matches 9 pairs with $\pi_2$, but it does not match any with $\pi_3$.

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:

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

where $K$ is the number of distinct rankings in $S$ and $Q(\pi_i, S)$ is the average similarity of the ranking $\pi_i$ 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, $abs\left(log\left(Q(\pi_i, S)\right)\right)$ can be seen as a dispersion measure around $\pi_i$. If a lot of rankings in $S$ are similar to $\pi_i$, $Q(\pi_i, S)$ will be close to 1. On the other hand, low values are obtained when there are no rankings in $S$ that are similar to $\pi_i$. 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 $abs\left(log\left(0.90\right)\right) = 0.105$ and $abs\left(log\left(0.5\right)\right) = 0.693$, as expected.

The value of $Ent_{LR}$ depends strongly on the threshold $\theta_{disc}$. 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 $\theta_{disc}$ 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 $\tau$ 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 $\tau$ 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 $\tau$ and Spearman $\rho$, 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 $\theta_{disc}$ 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*, $U_\pi$. $U_\pi$ 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 $U_\pi$ 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 | $U_\pi$ |
|---|---|---|---|---|---|
| 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 $\tau$ 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 $\theta_{disc}$ parameter

The entropy of a set of rankings varies depending on the value of the $\theta_{disc}$ 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 $\theta_{disc}$ thresholds, varying $\theta_{disc}$ from 0 to 1 by steps of 0.1.

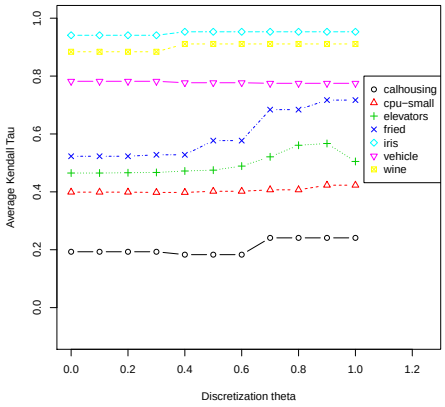


**Figure 3.1:** Accuracy of APRIORI-LR (expressed in terms of Kendall $\tau$) as $\theta_{disc}$ varies, in datasets where the distinct rankings represent less than 5% of the data. ($U_\pi < 5\%$)

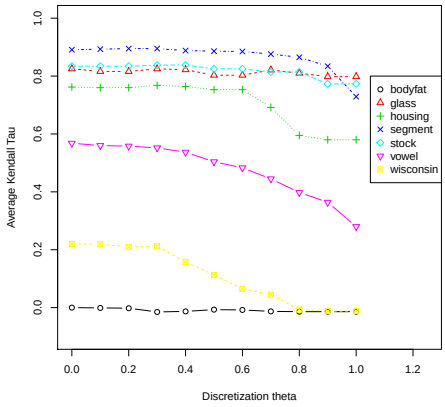


**Figure 3.2:** Accuracy of APRIORI-LR (expressed in terms of Kendall $\tau$) as $\theta_{disc}$ varies, in datasets where the distinct rankings represent more than 5% of the data. ($U_\pi \geq 5\%$)

The results indicate that $\theta_{disc}$ 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 $\theta_{disc}$ 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 $U_\pi$ is that they seem to behave differently. For the first group (Figure 3.1) as $\theta_{disc}$

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 $\theta_{disc}$ 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 $\theta_{disc}$ threshold allows the method to group larger ranges into each bin. In datasets with higher $U_\pi$, 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 $\theta_{disc}$ plays an important role in the effectiveness of the partitions made by MDLP-R. Also, by measuring $U_\pi$, we can get some clues about a reasonable value for $\theta_{disc}$.

### 3.5.2 Results on Artificial Datasets

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

| | | | Partitions | |
|---|---|---|---|---|
| TID | $x^1$ | $\pi$ | 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 $\pi_1$ is much closer to $\pi_2$ than to $\pi_3$, $T_2$ 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 $U_\pi$ should also increase. Therefore it becomes harder to learn an accurate model. In the following experiments, $U_\pi$ 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-

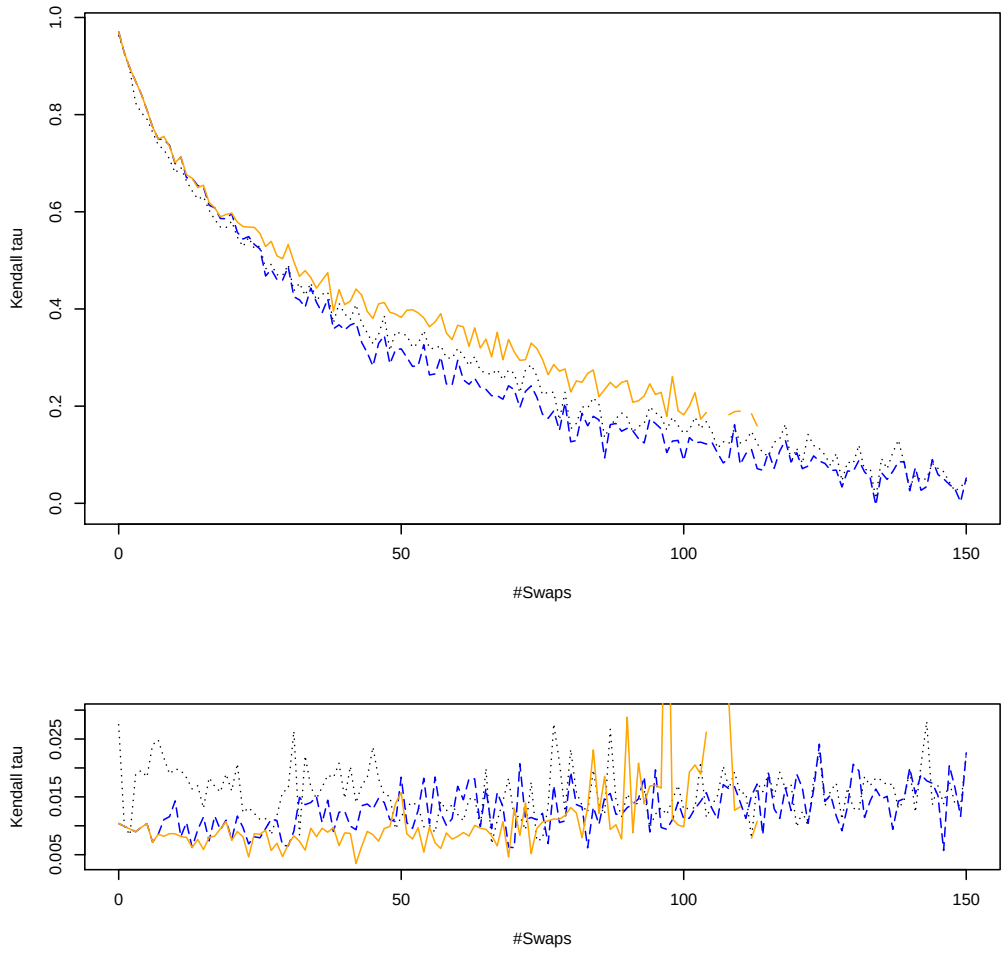


**Figure 3.3:** Accuracy (Top) and its Standard Deviation (Bottom) of the APRIORI-LR (expressed in terms of Kendall $\tau$) as a function of the number of swaps and its standard deviation, for MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line).

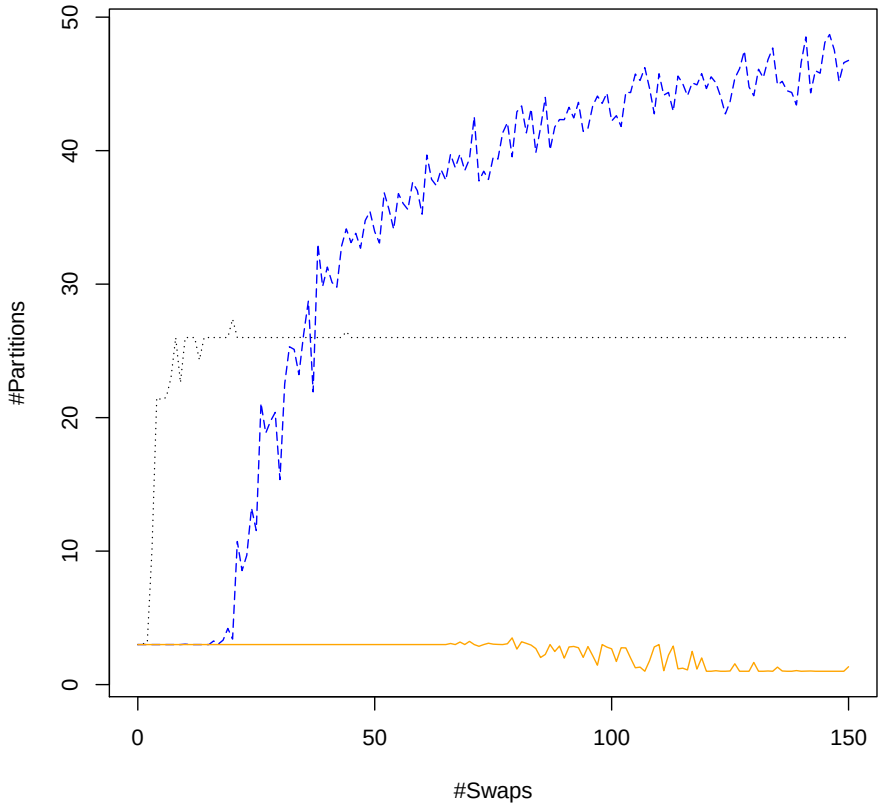


**Figure 3.4:** Comparison of the average number of partitions generated by MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line).

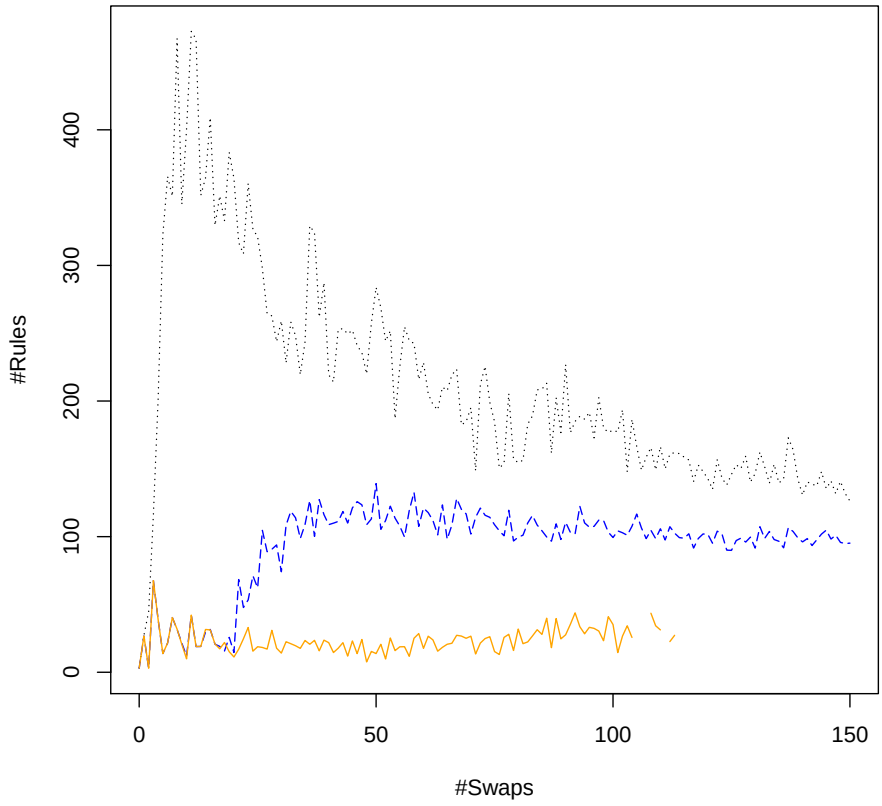


**Figure 3.5:** Comparison of the number of rules generated by APRIORI-LR after discretization with MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line).

**Algorithm 2** Parameter tuning method.

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

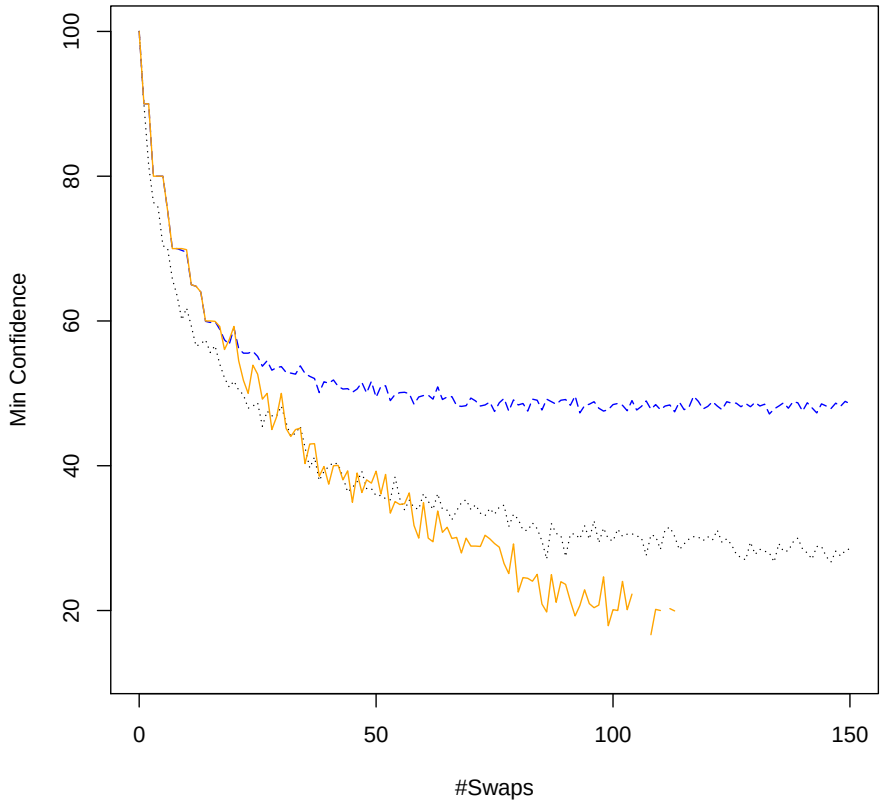


**Figure 3.6:** Comparison of the average *minconf* used by APRIORI-LR with the three discretization methods. MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line).

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

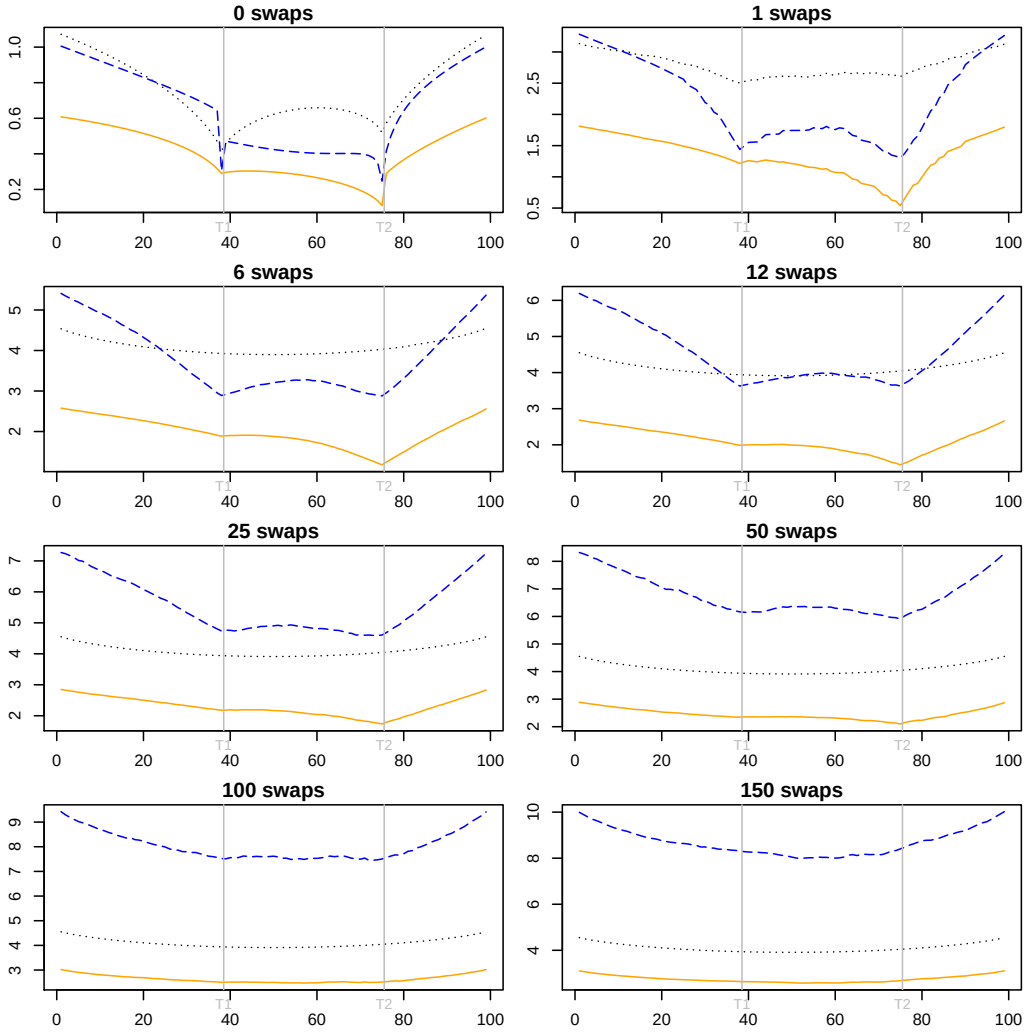


**Figure 3.7:** Comparison of the entropy values for MDLP (black dotted line), MDLP-R (blue dashed line) and EDiRa (orange line) for each candidate cut point for different number of swaps. Where the vertical axis is the entropy and the horizontal axis the candidate cut points.

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, $\tau$.

| | MDLP | | | | MDLP-R | | | | EDiRa | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\tau$ | mconf | #rules | #part | $\tau$ | mconf | #rules | #part | $\tau$ | 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 $\tau$ | .651 | - | - | - | .591 | - | - | - | .656 | - | - | - |
| standard dev $\tau$ | .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 $U_\pi$ 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, $\tau$).

| | MDLP | | MDLP-R | | EDiRa | |
|---|---|---|---|---|---|---|
| | $\tau$ | #part | $\tau$ | #part | $\tau$ | #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 $\tau$ | .607 | - | .593 | - | .605 | - |
| standard dev $\tau$ | .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 $U_\pi$, 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 $\rho$ 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 $\theta_{disc}$ 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 $\theta_{disc}$ 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

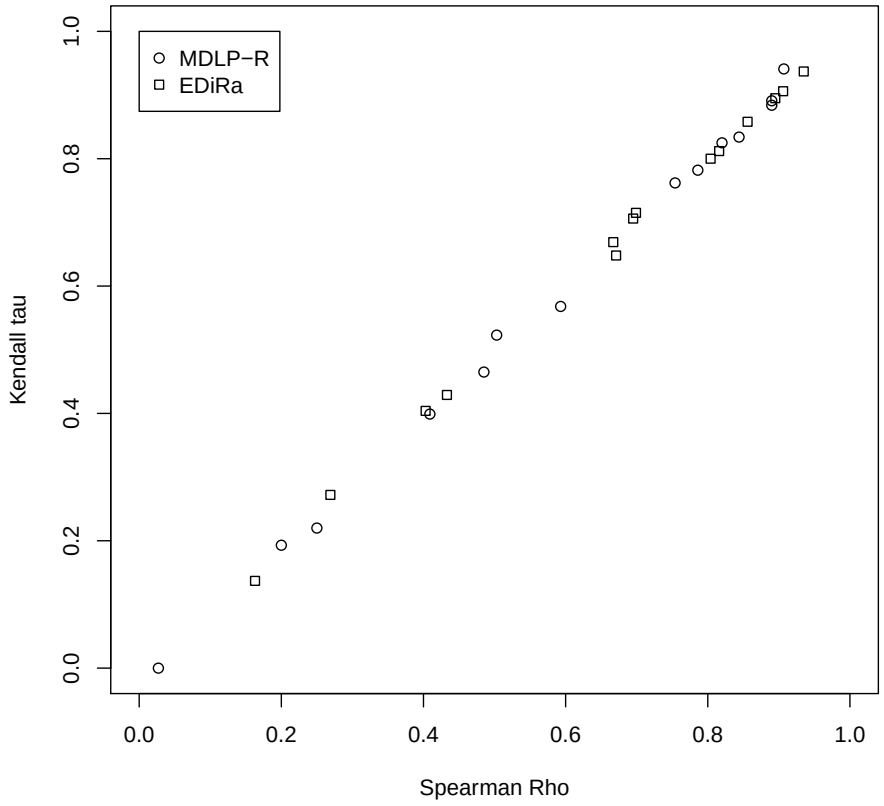


**Figure 3.8:** Comparison of the accuracy (in terms of Kendall $\tau$) of APRIORI-LR in the datasets from Table 3.3. The data was discretized with MDLP-R (circles) and EDiRa (squares), using Kendall (vertical-axis) and Spearman (horizontal-axis) similarity measures as $s'$.

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 $\tau$.

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.