MORE–PLR: Multi-Output Regression Employed for Partial Label Ranking

Santo M. A. R. Thies[1] ⋆, Juan C. Alfaro[3], and Viktor Bengs[1,2]

[1] Chair of Artificial Intelligence and Machine Learning, Institute of Informatics, Ludwig-Maximilians-Universität München, Munich, Germany, 80799
S.Thies@campus.lmu.de
[2] Munich Center for Machine Learning, Munich, Germany viktor.bengs@lmu.de
[3] Laboratorio de Sistemas Inteligentes y Minería de Datos, Departamento de Sistemas Informáticos, Instituto de Investigación en Informática de Albacete, Universidad de Castilla-La Mancha, Albacete, Spain, 02071
JuanCarlos.Alfaro@uclm.es

Abstract. The partial label ranking (PLR) problem is a supervised learning scenario where the learner predicts a ranking with ties of the labels for a given input instance. It generalizes the well-known label ranking (LR) problem, which only allows for strict rankings. So far, previous learning approaches for PLR have primarily adapted LR methods to accommodate ties in predictions. This paper proposes using multi-output regression (MOR) to address the PLR problem by treating ranking positions as multivariate targets, an approach that has received little attention in both LR and PLR. To effectively employ this approach, we introduce several post-hoc layers that convert MOR results into a ranking, potentially including ties. This framework produces a range of learning approaches, which we demonstrate in experimental evaluations to be competitive with the current state-of-the-art PLR methods.

Keywords: Preference learning · Bucket order · Multi-output regression · (Partial) label ranking

1 Introduction

In many machine learning applications, especially those involving human activities like assessing medical treatments, opinion polls, sports competitions, or recommender systems, the available data often consists of partial or entirely qualitative information rather than quantitative data. For example, consider the task of a human labeler during the fine-tuning step of a large language model (LLM): Given a prompt, the LLM generates various responses, which the human labeler ranks from worst to best without assigning numerical scores to the outputs (see Fig. 1). This purely qualitative signal is further processed to learn a suitable reward function for fine-tuning the LLM for better alignment [25].

⋆ Corresponding author: S.Thies@campus.lmu.de

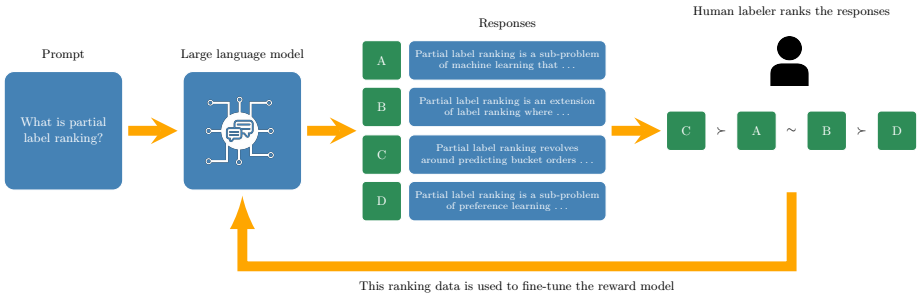


Fig. 1: Given a prompt, the LLM generates multiple answers, which are ranked from worst to best by a human labeler, and this ranking is subsequently used to fine-tune the LLM

The example above illustrates the learning task in the label ranking (LR) problem [31]. Here, a learner outputs a ranking of available labels (outputs) for a given instance (prompt), aiming to align this ranking with an unknown ground-truth ranking. This specific learning task falls within the broader field of preference learning [23], and it has been applied in various learning scenarios, such as multi-label classification [17], algorithm selection [19], and monocular depth estimation [26], among others.

Despite the widespread use of these learning methods, a practical limitation of the learning setting is that the learner must necessarily output a total order of the labels. Therefore, the current methods distinctly exclude the possibility of labels sharing the same rank, that is, being tied. If we revisit the previous example of the human labeler involved in LLM fine-tuning, it is quite common for them to assign the same rank to two (or more) LLM outputs (see Figure 1). This situation may occur, for instance, if the outputs are considered equally good, mediocre, or bad or if the labeler is not sufficiently informed and thus abstains from making explicit distinctions between ranks.

To address this limitation, the LR problem was recently generalized to the partial label ranking (PLR) problem, where the dataset may include rankings with ties [3], also referred to as bucket orders. Additionally, in the PLR problem, the learner can produce bucket orders (rankings with ties) instead of strict total orders. So far, the proposed methods have built upon existing LR approaches as a foundation and extended them to accommodate ties in the output. For instance, the decision tree approach outlined in [10] for the LR problem was extended by the initial work on PLR [3].

This paper introduces a meta-learning approach, previously explored only in a specialized form for LR problems [9,15]. The fundamental concept here is to consider the rank positions as multivariate targets, allowing every multi-output regression (MOR) learner to be considered a PLR learner. However, since the predicted multivariate vector does not necessarily correspond directly

to a rank position vector, a transformation step or PLR-post-hoc layer must be implemented afterward. To this end, we propose several PLR-post-hoc layers, each capable of converting an arbitrary multivariate vector into a rank position vector, thereby encoding the final bucket order. This results in an entire class of learning approaches termed MORE-PLR (multi-output regression employed for partial label ranking). Each learner within this class comprises two components: the underlying multi-output regression (base) learner and the employed PLR-post-hoc layer.

Compared to the state-of-the-art method of pairwise comparison reduction, which inherently exhibits a quadratic runtime regarding the available labels, the MORE-PLR approach generally features a linear runtime dependency. Our experimental results demonstrate satisfactory accuracy on benchmark datasets, contingent on the combination of MOR learner and PLR-post-hoc layer used.

The paper is structured as follows. Starting with an overview of related work in Section 2, Section 3 formally describes the PLR and the MOR problems. Section 4 details the class of MORE-PLR learners, with a special focus on the potential PLR-post-hoc layers. Experiments on benchmark datasets using the state-of-the-art methods of PLR are presented in Section 5, followed by our main conclusions in Section 6.

2 Related Work

This section discusses the most significant works on LR and PLR learning scenarios.

Label Ranking. Three main types of approaches are commonly employed to address the LR problem [34]. The first type involves a reduction technique that transforms the LR problem into a series of classification tasks, which are then combined to make the final ranking predictions [9,20,22,32]. The second approach adapts classic machine learning algorithms, such as decision trees or neural networks, to the structure of the LR problem itself [10,27,28,33]. Finally, the third approach employs ensemble methods, where multiple models are trained and their predictions are aggregated for each instance [1,13,29]. The idea of using MOR, as proposed in this paper, has been explored in [15]. In [15], the single-target method was considered with tree-based learners, where the output was subsequently sorted to obtain the final predicted ranking. This approach represents a particular case of our proposed class of MORE-PLR learners. However, the effectiveness of this rather simplistic approach experimentally remains uncertain, as it disregards potential dependencies between ranking positions (see Section 3.2). Instead, the authors investigate theoretical questions regarding sample complexity bounds or Bayes errors. Another similar idea was explored in [9], focusing on restricted ordinal classification rather than MOR.

Partial Label Ranking. Methods addressing the PLR problem typically extend learning algorithms originally designed for the LR problem to handle ties in the

prediction. Decision trees [10], ensemble learning approaches [1], and the pairwise comparison reduction technique [20] have all been adapted for PLR as well [4,5]. Additionally, the instance-based PLR method applies the classical principle of nearest neighbors classification for PLR [3].

3 Preliminaries

This section introduces the fundamental theoretical concepts necessary for comprehending the paper's content. Section 3.1 formally introduces the partial label ranking (PLR) problem, elucidating all relevant concepts and terminology. Similarly, Section 3.2 introduces the multi-output regression (MOR) framework.

3.1 Partial Label Ranking

Let $\mathcal{I} = \{u_1, \ldots, u_k\}$ denote a set of items, where $k \in \mathbb{N}$. An ordered partition of $\mathcal{I}$ into non-empty disjoint subsets, denoted as $\mathcal{B} =< \mathcal{B}_1, \ldots, \mathcal{B}_c >$ with $c \in [k] = \{1, \ldots, k\}$, $\bigcup_{l=1}^{c} \mathcal{B}_l = \mathcal{I}$ and $\mathcal{B}_l \subseteq \mathcal{I}$, is referred to as a *bucket order*. It specifies a total order with ties $\succeq_{\mathcal{B}}$ on $\mathcal{I}$ as follows: if $u_i \in \mathcal{B}_{l_i}$ and $u_j \in \mathcal{B}_{l_j}$, then $u_i \succ_{\mathcal{B}} u_j$ if and only if $l_i < l_j$, while $u_i \sim_{\mathcal{B}} u_j$ if and only if $l_i = l_j$. In other words, items belonging to the same bucket are tied, while items in preceding buckets are ranked higher than (or preferred to) all items in subsequent buckets. Note that a total order (or full ranking) is recovered when $c = k$, meaning each bucket consists of exactly one item.

Another equivalent way of representing a bucket order $\succ_{\mathcal{B}}$ is through its associated bucket matrix $B = (B_{i,j})_{i,j=1}^{k} \in \{0, 0.5, 1\}^{k \times k}$, where each entry is computed as

$$B_{i,j} = \begin{cases} 1 & \text{if } u_i \succ_{\mathcal{B}} u_j, \\ 0.5 & \text{if } u_i \sim_{\mathcal{B}} u_j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In other words, pairs of items belonging to the same bucket are assigned an 0.5 entry. Otherwise, a 0 or 1 value is assigned based on whether an item is in a subsequent or preceding bucket, respectively.

Given some instance or feature space $\mathcal{X}$, and considering the items in $\mathcal{I}$ as *labels*, the *label ranking* (LR) problem [22] involves learning a mapping $\mathcal{X} \to \mathcal{S}_k^{\succ}$, where $\mathcal{S}_k^{\succ}$ represents the set of all strict total orders (full rankings) of the labels $\mathcal{I}$ of size k. This mapping, also known as a *preference model* or *label ranker*, is learned based on a dataset $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{S}_k^{\succ,\text{inc}}$. Here, $\mathcal{S}_k^{\succ,\text{inc}}$ is the set of incomplete strict rankings of $\mathcal{I}$, considering that some labels may not be ranked for specific instances. The goal of the model is to predict a strict total order $\succ_{\boldsymbol{x}}$ of $\mathcal{I}$ for a given input instance $\boldsymbol{x} \in \mathcal{X}$. This prediction establishes an ordering relationship between any two items in the form $u_i \succ_{\boldsymbol{x}} u_j$, interpreted as u_i being preferred over u_j given the context $\boldsymbol{x}$.

In situations where predicting a strict total order of the items is not always feasible or desired, the *partial label ranking* (PLR) problem allows for predicting orders with ties. Formally, this entails a mapping $\mathcal{X} \to \mathcal{S}_k^{\succ}$, where $\mathcal{S}_k^{\succ}$ represents the set of all total orders with ties (bucket orders) of the labels $\mathcal{I}$. Learning is conducted based on a dataset $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{S}_k^{\succ,\text{inc}}$, where $\mathcal{S}_k^{\succ,\text{inc}}$ denotes the set of incomplete rankings with ties of $\mathcal{I}$. Thus, unlike a typical LR dataset, a PLR dataset may include rankings with tied labels for specific instances.

The predominant approach for learning a predictor for the PLR problem involves using a classification model, denoted as $\mathcal{M}$, to predict the probabilities of each relationship between labels $u_i \succ u_j$, $u_i \sim u_j$, and $u_i \prec u_j$ for every pair of items $(u_i, u_j) \in \mathcal{I} \times \mathcal{I}$ with $i < j$. These probabilities are then used to construct a so-called *pair order matrix* $C \in [0,1]^{k \times k}$ as follows

$$C_{i,j} = \mathbb{P}(u_i \succ u_j \,|\, \mathcal{M}) + 0.5 \cdot \mathbb{P}(u_i \sim u_j \,|\, \mathcal{M}). \tag{2}$$

Here, $C_{i,j}$ represents the predicted probability of u_i being preferred over u_j, plus half the predicted probability of u_i being tied with u_j. Since this pair order matrix C does not necessarily correspond to a bucket matrix, it must be transformed into one to obtain the final prediction. This transformation is achieved by solving the *optimal bucket order problem* (OBOP), which aims to find the closest bucket matrix in terms of L_1 distance to the pair order matrix

$$\text{argmin}_B \sum\nolimits_{u_i, u_j \in \mathcal{I}} |B_{i,j} - C_{i,j}| \,. \tag{3}$$

Several algorithms are available to address the OBOP; see [2,30] for an overview.

Figure 2a illustrates the schematic PLR learning process. Each time a prediction for an instance $\boldsymbol{x}$ is sought, it involves querying the classification model $\mathcal{M}$ and then solving the OBOP to obtain the pair order matrix C.

3.2 Multi-output Regression

Classical single-output regression involves predictions of the form $h : \mathcal{X} \to \mathbb{R}$, meaning that for a given instance $\boldsymbol{x}$, the model predicts a real-valued target $\hat{y} = h(\boldsymbol{x})$. For example, consider predicting the price of a car based on its characteristics such as 75 horsepower, 1462 engine displacement, 4 cylinders, etc. *Multi-output regression* (MOR) is the multivariate extension of classical single-output regression [8], where predictions are of the form $H : \mathcal{X} \to \mathbb{R}^q$. In other words, for a given instance $\boldsymbol{x}$, a real-valued *multivariate* target $\hat{\boldsymbol{y}} = H(\boldsymbol{x})$ with $q \in \mathbb{N}$ target variables are predicted. For example, a car with specific characteristics requires a price and fuel consumption prediction.

Based on the latter example, we can infer that in MOR, there may be relationships within the feature space, between the feature space and the target space, and within the target space itself. In the previous example, there is undoubtedly some correlation between the price and a car's fuel consumption. Accordingly, most approaches for MOR extend single-output regression models to account for this potential additional correlation.

These methods can be classified into two categories. First, *tranformation methods* tackle the problem of constructing MOR models by combining multiple (standard) single-output models. This class of methods includes:

- The *single-target method* predicts each target variable independently, disregarding potential interdependencies.
- The *stacking method* performs regression for each target variable initially, then uses these predictions as additional features in a subsequent step.
- The *regression chain method* starts by defining an order (random or heuristic) for the target variables and then learns single-output regressors incorporating the predictions of the previous learner(s) in the chain into the feature space for the prediction of the next target variable.
- The *multi-output support vector regression* transforms the feature space to handle MOR as a single-output regression problem.

The second class of approaches is *algorithm adaptation methods*, which modify classical models to inherently support multi-output problems by capturing all relationships and dependencies among the outputs. Multi-target Gaussian processes [24] and multi-target regression trees [11] are two such algorithms proposed in the literature [8], among others.

4 MORE-PLR

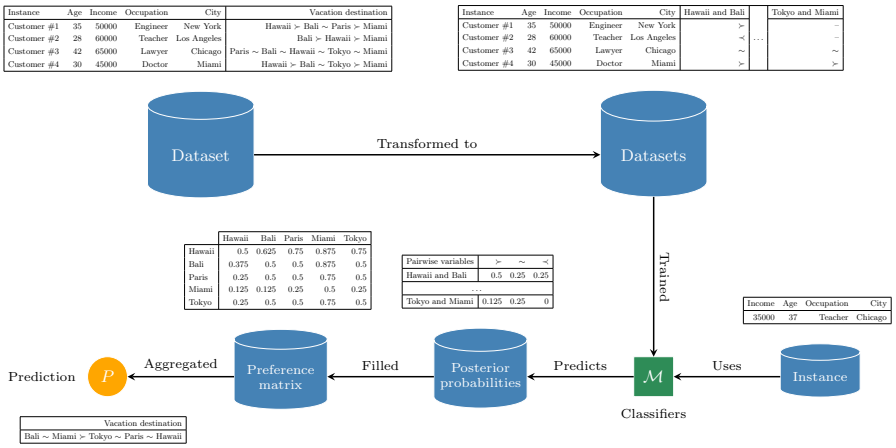
This section introduces our meta-learning approach for the PLR problem, which involves treating a bucket order equivalently as a vector of rank positions used as target values in MOR. As the output of a MOR model may not directly correspond to a rank position vector encoding the desired bucket order, we employ a PLR-post-hoc layer to transform it accordingly. At a high level, the learning process framework (see Figure 2b) mirrors the standard PLR learning process (see Figure 2a). Specifically, the multi-output regressor takes the role of the pairwise classifier, while the PLR-post-hoc layer replaces the OBOP step.

4.1 Bucket Orders as Rank Position Vectors

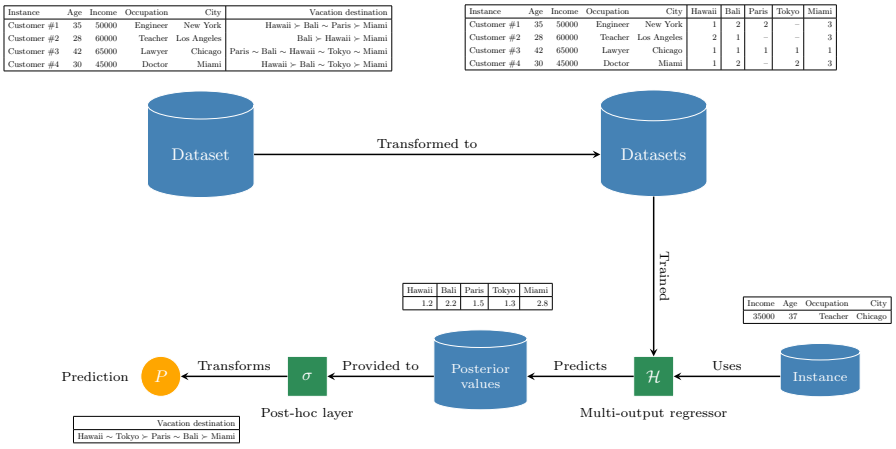
Currently, the basic building block for learning in the PLR problem is that a bucket order $\succ_{\mathcal{B}}$ is represented equivalently by bucket matrix through (1). However, another equivalent representation of a bucket order $\succ_{\mathcal{B}}$ is through a *rank position vector*. This vector $\boldsymbol{p} = (p_1, \ldots, p_k)$, where each $p_i \in 1, \ldots, k$, describes the ranking relationship among a set of items according to some weak order. In particular, the rank position vector $\boldsymbol{p}$ for a bucket order $\succ_{\mathcal{B}}$ is obtained according to

$$p_i = 1 + \sum\nolimits_{l=1}^{c} \mathbb{1}_{\{B_l \succ_{\mathcal{B}} u_i\}}, i \in [k], \tag{4}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function and $B_l \succ_{\mathcal{B}_l} u_i$ means that u_i is in a subsequent bucket of B_l. For example, let us assume that we have $k = 4$ items and



(a) PLR



(b) MORE

Fig. 2: In this example, the goal is to predict a total order with ties of vacation destinations for a customer based on their demographic information

the buckets are $\mathcal{B} =< \{u_1\}, \{u_2, u_3\}, \{u_4\} >$. Then, the rank position vector is $\boldsymbol{p} = (1, 2, 2, 3)$.

Transforming the rankings in a PLR dataset, which consists solely of complete rankings $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{S}_k^{\succ}$, allows it to be used as a MOR dataset $\mathcal{D} \subseteq \mathcal{X} \times \mathbb{R}^d$. Thus, we can immediately use this dataset for a multi-output regressor. Now, let us consider a dataset in the form $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{S}_k^{\succ,\text{inc}}$, where incomplete rankings may be present. Applying the same procedure as before, we transform the PLR dataset into a MOR dataset $\mathcal{D} \subseteq \mathcal{X} \times (\mathbb{R} \cup \{\text{NA}\})^d$, where missing labels are assigned a NA value (see Figure 2b). The most straightforward approach is to remove all data points with a NA value from the dataset and train the MOR model on the remaining data. Depending on the MOR method, fewer data points can be removed. When using the single-target method, all data points without NA values for the respective target variable are used, potentially having more training data as in the simple approach. As we will see later, this makes the single-target method perform better for incomplete data cases.

4.2 PLR-post-hoc Layers

Predictions in PLR problems are bucket orders of the underlying items, which can be encoded equivalently using a bucket order matrix or a rank position vector. However, applying MOR on the transformed data, as described above, typically does not yield a rank position vector directly during prediction. Instead, the predictions are usually k-dimensional vectors, which we will refer to as prediction vectors going forward. To apply these predictions in the PLR problem, they must be transformed into suitable rank position vectors.

To accomplish this, we introduce several transformations presented below, which can be viewed as post-hoc layers within the overall learning process. These layers, termed PLR-post-hoc layers, are mappings of the form $\sigma : \mathbb{R}^k \to \{1, \ldots, k\}^k$, and they operate on the output $\hat{\boldsymbol{y}} = H(\boldsymbol{x})$ of a MOR learner (see Figure 2b). At the heart of these layers is the construction of an auxiliary bucket order $\succ_{\mathcal{B}(\sigma)}$ for the entries of the output vector $\hat{\boldsymbol{y}}$. This auxiliary order can subsequently be represented using a rank position vector denoted as $\boldsymbol{p}_{\mathcal{B}(\sigma)}(\hat{\boldsymbol{y}})$. The layer mapping itself is defined as $\sigma(\hat{\boldsymbol{y}}) = \boldsymbol{p}_{\mathcal{B}(\sigma)}(\hat{\boldsymbol{y}})$.

Round-Rank (RR) Layer. A straightforward approach is to derive the rank vector by rounding the k-dimensional prediction vector. The bucket order $\succ_{\mathcal{B}(\sigma_{\text{RR}})}$ is then determined from the entries of the output vector $\hat{\boldsymbol{y}}$ as follows

$$\begin{aligned} \hat{y}_i \succ_{\mathcal{B}(\sigma_{\text{RR}})} \hat{y}_j \quad &\Leftrightarrow \quad \text{round}(\hat{y}_i) > \text{round}(\hat{y}_j), \\ \hat{y}_i \sim_{\mathcal{B}(\sigma_{\text{RR}})} \hat{y}_j \quad &\Leftrightarrow \quad \text{round}(\hat{y}_i) = \text{round}(\hat{y}_j), \end{aligned}$$

where $\text{round}(x) = \text{argmin}_{n \in \mathbb{N}} |n - x|$ is the rounding operator[4] From this, we derive a rank position vector $\boldsymbol{p}_{\mathcal{B}(\sigma_{\text{RR}})}(\hat{\boldsymbol{y}})$ using (4).

[4] We take the larger number in the case of two minimizers.

The advantage of this layer is its compatibility with any underlying MOR learner, accommodating ties so that items can share the same rank and consequently be placed in the same bucket. This is particularly useful when items are predicted to have values close to an actual rank position. However, a drawback is its sensitivity to the rounding operator, which can lead to decisions that may seem arbitrary. For instance, items with prediction values like 2.48 and 2.52 might end up in different buckets despite their proximity.

Prediction Interval (PI) Layer. This layer is based on the concept of overlapping intervals. If we have prediction intervals of the form $[\hat{y}_i - c_i, \hat{y}_i + c_i]$ available, we can construct a bucket order $\succ_{\mathcal{B}(\sigma_{\mathrm{PI}})}$ as follows

$$\begin{aligned}
\hat{y}_i \succ_{\mathcal{B}(\sigma_{\mathrm{PI}})} \hat{y}_j \quad &\Leftrightarrow \quad \nexists \{i_1, \ldots, i_l\} \subset \{1, \ldots, k\} \setminus \{i, j\} : \ \forall r = 0, \ldots, l, \\
&\quad [\hat{y}_{i_r} - c_{i_r}, \hat{y}_{i_r} + c_{i_r}] \cap [\hat{y}_{i_{r+1}} - c_{i_{r+1}}, \hat{y}_{i_{r+1}} + c_{i_{r+1}}] \neq \emptyset, \\
\hat{y}_i \sim_{\mathcal{B}(\sigma_{\mathrm{PI}})} \hat{y}_j \quad &\Leftrightarrow \quad \exists \{i_1, \ldots, i_l\} \subset \{1, \ldots, k\} \setminus \{i, j\} : \ \forall r = 0, \ldots, l, \\
&\quad [\hat{y}_{i_r} - c_{i_r}, \hat{y}_{i_r} + c_{i_r}] \cap [\hat{y}_{i_{r+1}} - c_{i_{r+1}}, \hat{y}_{i_{r+1}} + c_{i_{r+1}}] \neq \emptyset,
\end{aligned}$$

where we set $i_0 = i$ and $i_{l+1} = j$. In other words, items are placed in the same bucket if their prediction intervals overlap or if there is a cascading effect: if a position i overlaps with position j and this overlaps extends to a third position m, then i and m get assigned the same bucket due to their overlap with j.

It is worth noting that this can lead to undesired position vectors representing essentially only one bucket. However, the resulting rank position vector $\boldsymbol{p}_{\mathcal{B}(\sigma_{\mathrm{PI}})}(\hat{\boldsymbol{y}})$ offers the advantage of being grounded in the models' uncertainty when multiple items are placed in the same bucket. In fact, the prediction interval represents all possible predictions given the model's uncertainty. If these intervals overlap, it suggests that the items have the same rank from the model's perspective of uncertainty. Unfortunately, not all methods inherently provide prediction intervals, but standard techniques such as Gaussian processes or random forests do. Alternatively, through conformal prediction, one can construct prediction intervals for any prediction method [6].

5 Experiments

This section investigates the performance of some learning approaches in the MORE-PLR class and compares them to the current state-of-the-art models for the PLR problem. We first describe the technical background of the experimental evaluation and then present the results on the standard benchmarking datasets.

5.1 Reproducibility and Methodology

All experiments were executed on a machine with up to 128 CPU cores and 96GB of RAM. The source code for evaluation and the complete results can be found at `https://github.com/Advueu963/MORE-PLR`.

The models have not received significant fine-tuning of the hyperparameters, and the results can be seen as out-of-the-box performance. All algorithms were evaluated using five repetitions of a ten-fold cross-validation method to compensate for potential random deviations. We measure the accuracy using the τ_X *rank correlation coefficient* [14]. Formally, given two bucket orders $\mathcal{B}^r$ and $\mathcal{B}^p$ defined over the items $\mathcal{I}$, the τ_X rank correlation coefficient is given by

$$\tau_X(\mathcal{B}^r, \mathcal{B}^p) = \frac{\sum_{u=1}^{k} \sum_{v=1}^{k} \beta_{uv}^r \beta_{uv}^p}{k\ (k-1)}, \tag{5}$$

where

$$\beta_{uv}^l = \begin{cases} 1 & \text{if } y^u \succ_{\mathcal{B}^l} y^v \text{ or } y^u \sim_{\mathcal{B}^l} y^v, \\ -1 & \text{if } y^v \succ_{\mathcal{B}^l} y^u, \\ 0 & \text{if } u = v. \end{cases}$$

The τ_X rank correlation coefficient values lie in the interval $[-1, 1]$. Hereby, 1 indicates a perfect correlation between the bucket orders, -1 is a perfect correlation of a bucket order to its reversed bucket order, and a value close to zero with no correlation. Thus, the model aims to achieve values of τ_X to 1. When speaking of CPU time in the following, we mean the time needed to train a learner and the time needed for inference averaged over the 5×10-cv.

The results were analyzed using the procedure described in [12,18] with the `exreport` software tool [7]. First, a *Friedman test* [16] was applied with the null hypothesis that all the algorithms have equal performance. If this hypothesis was rejected, a *post-hoc test* using *Holm's procedure* [21] was performed to compare all the algorithms against the one ranked first by the Friedman test. Both tests were conducted at a significance level of 5%.

We also considered the case of 30% and 60% missing labels in the rankings. The instances with missing class labels are dropped from the training datasets following the typical procedure [3,10].

5.2 Datasets and Algorithms

We used the standard datasets on which methods for PLR have been experimentally investigated (see [3,4,5]). These are modifications of datasets used for LR [10], which we also include to investigate potential differences in performance for these two settings. The datasets are publicly available at: `https://www.openml.org/search?type=data&sort=runs&status=active&uploader_id=%3D_25829`

The below-listed algorithms were used:

- *Pairwise comparison reduction technique* [4]. It is the current state-of-the-art method for the PLR problem, called ranking by pairwise comparison (`RPC`).
- *Single-target methods*. This learner uses the single-target method (`ST`) as the MOR technique with the RR (`ST-RR`) and PI (`ST-PI`) as layers.
- *Chain methods*. We also investigate the chaining method (`Chain`) as the underlying MOR method, which utilizes a possible dependency between the targets, which we call *order*. However, to determine the best order, one must

check all possible $k!$ orders, which is not feasible. Therefore, we propose using the correlation of the target variables as a good indicator. To build an order, we first look at the target variable with the highest additive correlation compared to all other target variables and choose it as the starting target. This approach is applied recursively to the remaining targets until the order is created. We tested it with the RR (`Chain-RR`) and PI (`Chain-PI`) layers.
- *Algorithm adaption method.* Since random forest regressors can directly model MOR [8], we also included it as a representative of the algorithm adaption methods (`Native`). Both the RR (`Native-RR`) and PI (`Native-PI`) layers are used to adapt it to the PLR problem.

We used random forests (classifier or regressor, as corresponding) as a base estimator for all the algorithms. Random forests, as an ensemble method, allows for obtaining prediction intervals as follows: if $\#trees$ is the number of trees and σ_i is the standard deviation of the individual outputs, we set

$$c_i = \frac{q\sigma_i}{\sqrt{\#trees}},$$

which corresponds to the sample standard error. The σ-factor $1 \leq q \leq 3$ specifies the probability that the confidence interval will overlap with the true target corresponding to the 3σ rule. Note that this could be adapted by multiplying it by a quantile of the normal distribution, but we do not do it for the sake of simplicity.

The impact of different values is depicted in Fig. 3, where the averaged τ_X rank correlation coefficient (only for complete rankings for clarity) over all datasets is provided. According to these results, for the LR problem, higher values of q correspond to lower accuracy, while for the PLR problem, the opposite trend is observed. Therefore, to balance across both problems, we used $q = 1$.

5.3 Results

This section analyses the algorithms regarding accuracy and efficiency.

Accuracy Table 1 provides a summary of the hypothesis tests conducted, organized by problem tested (columns) and missing percentage (rows). Both Friedman's test p-value and Holm's post-hoc test results are displayed for each group. It should be noted that even if the data only contains strict rankings, i.e., if the problem is an LR problem, the PLR learners can still predict bucket orders.

In light of these results, the MORE-PLR learners perform particularly well in LR problems with complete rankings, as they are all ranked ahead of the `RPC` method. Moreover, with incomplete rankings, the `ST-PI` method is statistically different from `RPC` and the rest of the MORE-PLR learners, except for the `ST-RR` algorithm, which shows no statistical difference compared to it. Therefore, for LR problems, it can be concluded that the `ST` methods provide the best performance.

Looking at the results for the PLR problems, `RPC` is ranked first with 0% and 30% of missing labels but performs worse with 60%, where the `ST-RR` is

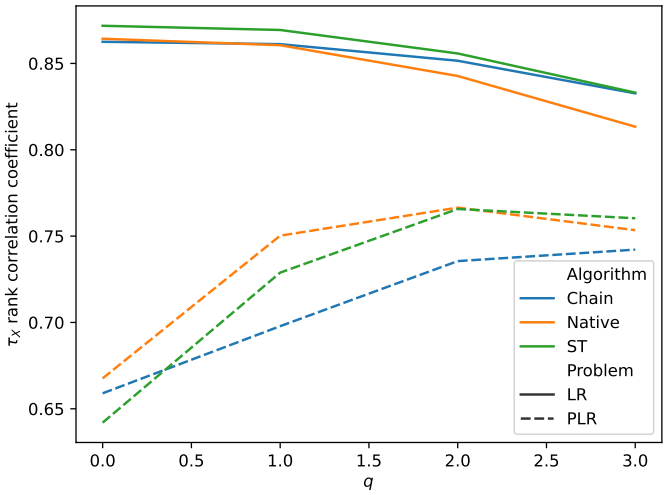


Fig. 3: Impact of q in the accuracy of the algorithms

ranked first. Furthermore, with complete rankings, there is no statistical difference between `RPC`, `Chain-RR`, and `Native-RR`. However, in incomplete rankings, `RPC`, `ST-RR`, and `ST-PI` perform equally well. Therefore, for PLR problems, these learners are more stability against missing labels compared to `Native` and `Chain`.

Considering both problems simultaneously, the `ST-RR` method consistently ranks first according to the Friedman test across all cases. Only the `Chain-PI` method shows statistical difference in the complete case. In contrast, the `ST-PI` and `RPC` methods do not exhibit any statistical difference in incomplete cases. Overall, the `ST` methods demonstrate greater stability across different problems and varying degrees of missing labels than the rest of the algorithms.

CPU Time Table 2 provides the time results (in seconds) of the algorithms with complete rankings. These results conclude that the fastest algorithms are `Native-RR` and `Native-PI`, as they only fit one model. Additionally, the results show that `ST-RR` and `ST-PI` are significantly faster overall concerning `RPC`, particularly with datasets such as letter. This is expected, as k models need to be fitted for `ST` in comparison to $\binom{k}{2}$ in `RPC`. The `RPC` approach sometimes demonstrates faster performance on datasets with a small number of labels (see iris or authorship), as `ST` and `RPC` fit a similar number of models in these cases. The difference in complexity can be attributed to the post-hoc layers, which act as minor bottlenecks, resulting in complexities of $\mathcal{O}(k)$ for RR and $\mathcal{O}(k^2)$ for PI. Importantly, the transformation step of RPC requires solving the OBOP problem, which has a complexity of $\mathcal{O}(klog(k))$.

Table 1: Friedman's and Holm's tests for accuracy with varying miss probability

LR						**PLR**						**Both**					
Missing percentage: 0%						**Missing percentage: 0%**						**Miss percentage: 0%**					
Friedman p-value: $\mathbf{1.204 \times 10^{-6}}$						Friedman p-value: $\mathbf{2.268 \times 10^{-10}}$						Friedman p-value: $\mathbf{2.187 \times 10^{-2}}$					
Holm results						Holm results						Holm results					
Method	Rank	P-value	Win	Tie	Loss	Method	Rank	P-value	Win	Tie	Loss	Method	Rank	P-value	Win	Tie	Loss
ST-RR	2.23	-	-	-	-	RPC	1.83	-	-	-	-	ST-RR	3.34	-	-	-	-
Chain-PI	3.23	$\mathbf{4.759 \times 10^{-1}}$	8	1	4	Chain-RR	2.81	$\mathbf{1.770 \times 10^{-1}}$	14	1	3	Chain-RR	3.47	$\mathbf{9.2484 \times 10^{-1}}$	14	0	17
ST-PI	3.23	$\mathbf{4.759 \times 10^{-1}}$	7	0	6	Native-RR	3.17	$\mathbf{1.282 \times 10^{-1}}$	15	0	3	Native-RR	3.74	$\mathbf{9.2484 \times 10^{-1}}$	18	0	13
Native-PI	3.46	$\mathbf{4.391 \times 10^{-1}}$	11	1	1	ST-RR	4.14	4.097×10^{-3}	14	2	2	RPC	3.97	$\mathbf{7.5489 \times 10^{-1}}$	15	2	14
Chain-RR	4.38	4.410×10^{-2}	10	0	3	Native-PI	4.56	6.262×10^{-4}	16	0	2	Native-PI	4.10	$\mathbf{6.6843 \times 10^{-1}}$	21	1	9
Native-RR	4.54	3.230×10^{-2}	12	0	1	ST-PI	4.92	9.265×10^{-5}	15	1	2	ST-PI	4.21	$\mathbf{5.6220 \times 10^{-1}}$	20	0	11
RPC	6.92	1.837×10^{-7}	13	0	0	Chain-PI	6.58	2.526×10^{-10}	17	0	1	Chain-PI	5.18	4.8311×10^{-3}	23	2	6
Miss percentage: 30%						**Miss percentage: 30%**						**Miss percentage: 30%**					
Friedman p-value: $\mathbf{2.222 \times 10^{-10}}$						Friedman p-value: $\mathbf{3.135 \times 10^{-18}}$						Friedman p-value: $\mathbf{2.760 \times 10^{-26}}$					
Holm results						Holm results						Holm results					
Method	Rank	P-value	Win	Tie	Loss	Method	Rank	P-value	Win	Tie	Loss	Method	Rank	P-value	Win	Tie	Loss
ST-PI	1.08	-	-	-	-	RPC	1.50	-	-	-	-	ST-RR	1.87	-	-	-	-
ST-RR	1.92	$\mathbf{3.180 \times 10^{-1}}$	12	0	1	ST-RR	1.83	$\mathbf{6.434 \times 10^{-1}}$	12	0	6	ST-PI	2.03	$\mathbf{7.688 \times 10^{-1}}$	16	0	15
RPC	3.62	5.473×10^{-3}	13	0	0	ST-PI	2.72	$\mathbf{1.793 \times 10^{-1}}$	15	0	3	RPC	2.39	$\mathbf{6.938 \times 10^{-1}}$	19	0	12
Chain-RR	5.23	3.003×10^{-6}	13	0	0	Native-PI	4.17	6.385×10^{-4}	18	0	0	Native-PI	4.74	5.023×10^{-7}	31	0	0
Native-RR	5.27	3.003×10^{-6}	13	0	0	Native-RR	5.33	4.072×10^{-7}	18	0	0	Native-RR	5.31	1.529×10^{-9}	31	0	0
Chain-PI	5.35	2.346×10^{-6}	13	0	0	Chain-PI	5.83	8.839×10^{-9}	18	0	0	Chain-PI	5.63	3.719×10^{-11}	31	0	0
Native-PI	5.54	8.389×10^{-7}	13	0	0	Chain-RR	6.61	7.597×10^{-12}	18	0	0	Chain-RR	6.03	2.013×10^{-13}	31	0	0
Miss percentage: 60%						**Missing percentage: 60%**						**Miss percentage: 60%**					
Friedman p-value: $\mathbf{3.259 \times 10^{-12}}$						Friedman p-value: $\mathbf{3.667 \times 10^{-17}}$						Friedman p-value: $\mathbf{1.277 \times 10^{-29}}$					
Holm results						Holm results						Holm results					
Method	Rank	P-value	Win	Tie	Loss	Method	Rank	P-value	Win	Tie	Loss	Method	Rank	P-value	Win	Tie	Loss
ST-PI	1.08	-	-	-	-	ST-RR	1.56	-	-	-	-	ST-RR	1.71	-	-	-	-
ST-RR	1.92	$\mathbf{3.180 \times 10^{-1}}$	12	0	1	RPC	2.06	$\mathbf{4.875 \times 10^{-1}}$	11	0	7	ST-PI	1.87	$\mathbf{7.688 \times 10^{-1}}$	16	0	15
RPC	3.08	3.651×10^{-2}	13	0	0	ST-PI	2.44	$\mathbf{4.341 \times 10^{-1}}$	15	0	3	RPC	2.48	$\mathbf{3.165 \times 10^{-1}}$	24	0	7
Native-PI	4.65	7.281×10^{-5}	13	0	0	Native-RR	4.61	6.607×10^{-5}	18	0	0	Native-RR	4.63	3.105×10^{-7}	31	0	0
Native-RR	5.58	4.634×10^{-7}	13	0	0	Native-PI	4.89	1.469×10^{-5}	18	0	0	Native-PI	5.18	1.047×10^{-9}	31	0	0
Chain-RR	5.62	4.248×10^{-7}	13	0	0	Chain-RR	6.14	9.674×10^{-10}	18	0	0	Chain-RR	5.92	8.464×10^{-14}	31	0	0
Chain-PI	6.08	2.168×10^{-8}	13	0	0	Chain-PI	6.31	2.526×10^{-10}	18	0	0	Chain-PI	6.21	1.429×10^{-15}	31	0	0

Given that the random forests algorithm has a complexity of $\mathcal{O}(bmnlog(n))$, where b represents the number of decision trees, m the number of features, and n the number of data points, the overall complexities are as follows:

- `RPC`: $\mathcal{O}(k^2bmnlog(n) + klog(k))$
- `ST-RR` and `Chain-RR`: $\mathcal{O}(kbmnlog(n) + k)$
- `ST-PI` and `Chain-PI`: $\mathcal{O}(kbmnlog(n) + k^2)$
- `Native-RR`: $\mathcal{O}(bmnlog(n) + k)$
- `Native-PI`: $\mathcal{O}(bmnlog(n) + k^2)$

It is worth noting that these theoretical computational complexities align with the results obtained from the experimental evaluation, confirming the expected performance patterns. The same conclusions are drawn for incomplete rankings but with lower times, given that fewer instances are used to train the models (tables omitted due to space restrictions).

6 Conclusions

This paper has investigated MOR for the PLR problem. As this can be done in various forms, we have identified a class of approaches for this purpose and elaborated on the critical components. Our experiments show that with a suitable choice of components, it is possible to keep up with state-of-the-art PLR

Table 2: Time results (in seconds) for complete rankings

Problem	Dataset	RPC	ST-RR	ST-PI	Chain-RR	Chain-PI	Native-RR	Native-PI
LR	authorship	3.215 ± 2.143	4.413 ± 3.083	8.111 ± 8.202	14.229 ± 0.111	13.944 ± 0.109	1.230 ± 0.056	1.297 ± 0.052
	glass	2.647 ± 0.201	2.358 ± 0.140	6.081 ± 3.058	6.811 ± 0.085	6.398 ± 0.079	1.131 ± 0.044	1.185 ± 0.051
	iris	1.268 ± 0.127	1.627 ± 0.087	1.642 ± 0.086	3.213 ± 0.066	2.977 ± 0.051	1.072 ± 0.048	1.110 ± 0.043
	letter	118.040 ± 3.901	84.247 ± 2.017	86.550 ± 6.024	78.141 ± 0.381	80.041 ± 0.329	10.694 ± 0.087	11.401 ± 0.083
	libras	12.479 ± 0.312	5.680 ± 0.275	5.795 ± 0.203	35.195 ± 0.129	33.872 ± 0.156	1.298 ± 0.048	1.314 ± 0.050
	movies	12.746 ± 0.213	5.374 ± 0.234	5.449 ± 0.244	26.577 ± 0.110	25.325 ± 0.144	1.188 ± 0.047	1.241 ± 0.044
	pendigits	12.231 ± 0.170	16.263 ± 0.171	16.580 ± 0.225	19.845 ± 0.091	19.393 ± 0.104	3.907 ± 0.056	4.013 ± 0.062
	political	5.209 ± 1.008	3.324 ± 2.430	9.638 ± 11.207	13.526 ± 0.141	13.150 ± 0.130	1.196 ± 0.052	1.288 ± 0.071
	segment	4.417 ± 0.159	5.134 ± 0.144	5.260 ± 0.164	9.855 ± 0.089	9.306 ± 0.084	1.588 ± 0.051	1.623 ± 0.052
	vehicle	1.981 ± 0.102	2.416 ± 0.123	2.451 ± 0.125	5.382 ± 0.079	5.110 ± 0.071	1.201 ± 0.045	1.269 ± 0.039
	vowel	7.173 ± 0.217	4.454 ± 0.207	4.532 ± 0.169	13.136 ± 0.084	12.268 ± 0.087	1.295 ± 0.046	1.317 ± 0.046
	wine	1.251 ± 0.116	1.674 ± 0.081	1.700 ± 0.062	3.549 ± 0.075	3.341 ± 0.055	1.065 ± 0.041	1.101 ± 0.045
	yeast	6.456 ± 0.157	4.735 ± 0.137	4.932 ± 0.175	12.440 ± 0.088	11.556 ± 0.090	1.520 ± 0.049	1.569 ± 0.045
PLR	algae	3.744 ± 0.218	2.955 ± 0.152	3.129 ± 0.170	8.940 ± 0.088	8.331 ± 0.087	1.174 ± 0.048	1.231 ± 0.043
	authorship	3.504 ± 2.121	4.713 ± 3.303	10.335 ± 7.374	14.262 ± 0.103	14.029 ± 0.094	1.249 ± 0.052	1.309 ± 0.056
	blocks	7.822 ± 0.143	8.480 ± 0.184	9.102 ± 1.738	7.399 ± 0.090	6.868 ± 0.081	1.790 ± 0.053	1.858 ± 0.051
	breast	2.732 ± 0.213	2.309 ± 0.139	2.479 ± 0.154	6.806 ± 0.085	6.351 ± 0.080	1.119 ± 0.047	1.170 ± 0.049
	ecoli	4.204 ± 0.189	3.039 ± 0.143	3.274 ± 0.159	9.299 ± 0.085	8.635 ± 0.083	1.182 ± 0.040	1.254 ± 0.050
	glass	2.795 ± 0.208	2.466 ± 0.149	2.660 ± 0.183	7.002 ± 0.086	6.531 ± 0.081	1.158 ± 0.051	1.206 ± 0.043
	iris	1.292 ± 0.101	1.627 ± 0.070	1.679 ± 0.065	3.270 ± 0.061	3.014 ± 0.059	1.106 ± 0.041	1.140 ± 0.046
	letter	216.145 ± 24.093	78.308 ± 2.347	76.169 ± 1.286	67.072 ± 0.354	69.422 ± 0.307	4.878 ± 0.088	5.100 ± 0.110
	libras	13.837 ± 2.705	5.806 ± 0.915	5.864 ± 0.167	35.210 ± 0.120	34.024 ± 0.141	1.267 ± 0.049	1.284 ± 0.049
	movies	13.628 ± 0.261	4.700 ± 0.149	5.048 ± 0.196	26.368 ± 0.118	25.073 ± 0.146	1.163 ± 0.041	1.231 ± 0.041
	pendigits	21.594 ± 0.419	19.399 ± 0.190	19.692 ± 0.188	19.053 ± 0.092	19.055 ± 0.097	2.646 ± 0.073	2.907 ± 0.059
	political	6.087 ± 1.056	3.770 ± 2.473	8.846 ± 3.545	15.217 ± 0.124	14.848 ± 0.091	1.422 ± 0.049	1.506 ± 0.068
	satimage	8.997 ± 0.167	10.833 ± 0.171	11.192 ± 0.170	15.281 ± 0.108	14.951 ± 0.102	2.206 ± 0.067	2.333 ± 0.061
	segment	5.545 ± 0.171	5.489 ± 0.141	5.698 ± 0.165	9.899 ± 0.099	9.305 ± 0.091	1.459 ± 0.052	1.522 ± 0.050
	vehicle	2.214 ± 0.113	2.623 ± 0.130	2.664 ± 0.123	5.495 ± 0.079	5.172 ± 0.073	1.220 ± 0.046	1.282 ± 0.046
	vowel	7.778 ± 0.235	4.471 ± 0.185	4.654 ± 0.199	13.238 ± 0.078	12.272 ± 0.091	1.290 ± 0.047	1.305 ± 0.045
	wine	1.308 ± 0.114	1.699 ± 0.089	1.739 ± 0.068	3.647 ± 0.069	3.416 ± 0.066	1.104 ± 0.041	1.143 ± 0.043
	yeast	7.554 ± 0.184	5.141 ± 0.176	5.410 ± 0.143	12.668 ± 0.088	11.773 ± 0.088	1.441 ± 0.056	1.503 ± 0.045

methods and, in some cases, in less time. Especially on LR problems, this approach performs well, suggesting that the PLR post-hoc layer component still has potential for improvement when it comes to PLR problems.

In future research, an interesting question would be the influence of the type of rank position vector on performance, as we used only one variant in our experiments. One potential improvement, particularly for PLR, could involve replacing the basic RR layer with an ϵ-RR layer that includes a learnable parameter ϵ. This parameter would define an acceptance region for values. Furthermore, having different-sized acceptance regions $\epsilon_1, \ldots, \epsilon_k$ for each target could be beneficial.

Acknowledgments

This work is partially funded by the following projects: SBPLY/21/180225/000062 (Junta de Comunidades de Castilla-La Mancha and ERDF A way of making Europe), PID2022-139293NB-C32 (MICIU/AEI/10.13039/501100011033 and ERDF, EU), and 2022-GRIN-34437 (Universidad de Castilla-La Mancha and ERDF A way of making Europe).

References

1. Aledo, J.A., Gámez, J.A., Molina, D.: Tackling the supervised label ranking problem by bagging weak learners. Information Fusion **35**, 38–50 (2017)
2. Aledo, J.A., Gámez, J.A., Rosete, A.: Utopia in the solution of the bucket order problem. Decision Support Systems **97**, 69–80 (2017)
3. Alfaro, J.C., Aledo, J.A., Gámez, J.A.: Learning decision trees for the partial label ranking problem. International Journal of Intelligent Systems **36**, 890–918 (2021)
4. Alfaro, J.C., Aledo, J.A., Gámez, J.A.: Pairwise learning for the partial label ranking problem. Pattern Recognition **140**, 109590 (2023)
5. Alfaro, J.C., Aledo, J.A., Gámez, J.A.: Ensemble learning for the partial label ranking problem. Mathematical Methods in the Applied Sciences **46**, 1–21 (2023)
6. Angelopoulos, A.N., Bates, S., et al.: Conformal prediction: A gentle introduction. Foundations and Trends® in Machine Learning **16**(4), 494–591 (2023)
7. Arias, J., Cózar, J.: exreport: Fast, reliable and elegant reproducible research (2015), `https://cran.r-project.org/web/packages/exreport/index.html`
8. Borchani, H., Varando, G., Bielza, C., Larranaga, P.: A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **5**(5), 216–233 (2015)
9. Cheng, W., Henzgen, S., Hüllermeier, E.: Labelwise versus pairwise decomposition in label ranking. In: Proceedings of the Workshop on Lernen, Wissen & Adaptivität. pp. 129–136 (2013)
10. Cheng, W., Hühn, J., Hüllermeier, E.: Decision tree and instance-based learning for label ranking. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 161–168 (2009)
11. De'Ath, G.: Multivariate regression trees: A new technique for modeling species–environment relationships. Ecology **83**(4), 1105–1117 (2002)
12. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006)
13. Dery, L., Shmueli, E.: BoostLR: A boosting-based learning ensemble for label ranking tasks. IEEE Access **8**, 176023 – 176032 (2020)
14. Emond, E.J., Mason, D.W.: A new rank correlation coefficient with application to the consensus ranking problem. Journal of Multi-Criteria Decision Analysis **11**, 17–28 (2002)
15. Fotakis, D., Kalavasis, A., Psaroudaki, E.: Label ranking through nonparametric regression. In: Proceedings of the 39th International Conference on Machine Learning. pp. 6622–6659 (2022)
16. Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. Annals of Mathematical Statistics **11**, 86–92 (1940)
17. Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine learning **73**, 133–153 (2008)
18. García, S., Herrera, F.: An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research **9**, 2677–2694 (2008)
19. Hanselle, J., Tornede, A., Wever, M., Hüllermeier, E.: Hybrid ranking and regression for algorithm selection. In: German Conference on Artificial Intelligence. pp. 59–72. Springer (2020)
20. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification for multiclass classification and ranking. In: Proceedings of the 15th International Conference on Neural Information Processing Systems. pp. 785–792 (2002)

21. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6**, 65–70 (1979)
22. Hüllermeier, E., Fürnkranz, J., Cheng, W., Brinker, K.: Label ranking by learning pairwise preferences. Artificial Intelligence **172**, 1897–1916 (2008)
23. Hüllermeier, E., Słowiński, R.: Preference learning and multiple criteria decision aiding: differences, commonalities, and synergies–part I. 4OR (2024)
24. Jakkala, K.: Deep gaussian processes: A survey. Computing Research Repository **abs/2106.12135** (2021)
25. Kaufmann, T., Weng, P., Bengs, V., Hüllermeier, E.: A survey of reinforcement learning from human feedback. arXiv preprint arXiv:2312.14925 (2023)
26. Lienen, J., Hüllermeier, E., Ewerth, R., Nommensen, N.: Monocular depth estimation via listwise ranking using the Plackett-Luce model. In: Proceedings of the 2021 Conference on Computer Vision and Pattern Recognition. pp. 14590–14599 (2021)
27. Ribeiro, G., Duivesteijn, W., Soares, C., Knobbe, A.J.: Multilayer perceptron for label ranking. In: Proceedings of the 22nd International Conference on Artificial Neural Networks and Machine Learning. pp. 25–32 (2012)
28. de Sá, C.R., Soares, C., Jorge, A.M., Azevedo, P., Costa, J.: Mining association rules for label ranking. In: Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 432–443 (2011)
29. de Sá, C.R., Soares, C., Knobbe, A., Cortez, P.: Label ranking forests. Expert Systems **34** (2017)
30. Ukkonen, A., Puolamäki, K., Gionis, A., Mannila, H.: A randomized approximation algorithm for computing bucket orders. Information Processing Letters **109**, 356–359 (2009)
31. Vembu, S., Gärtner, T.: Label ranking algorithms: A survey. In: Preference Learning, pp. 45–64. Springer Science+Business Media (2010)
32. Vogel, R., Clémen, S.: A multiclass classification approach to label ranking. In: 23rd International Conference on Artificial Intelligence and Statistics. pp. 1421–1430 (2020)
33. Zhou, Y., Liu, Y., Gao, X., Qiu, G.: A label ranking method based on Gaussian mixture model. Knowledge-Based Systems **72**, 108–113 (2014)
34. Zhou, Y., Liu, Y., Yang, J., He, X., Liu, L.: A taxonomy of label ranking algorithms. Journal of Computers **9**(3), 557–565 (2014)