Conformal Prediction without Nonconformity Scores

Jonas Hanselle[1,2,*] **Alireza Javanmardi**[1,2,*] **Tobias Oberkofler**[1,2] **Yusuf Sale**[1,2] **Eyke Hüllermeier**[1,2,3]

[1]Institute of Informatics, LMU Munich, Germany
[2]Munich Center for Machine Learning (MCML), Germany
[3]German Centre for Artificial Intelligence (DFKI), Kaiserslautern, Germany

Abstract

Conformal prediction (CP) is an uncertainty quantification framework that allows for constructing statistically valid prediction sets. Key to the construction of these sets is the notion of a nonconformity function, which assigns a real-valued score to individual data points: only those (hypothetical) data points contribute to a prediction set that sufficiently conform to the data. The point of departure of this work is the observation that CP predictions are invariant against (strictly) monotone transformations of the nonconformity function. In other words, it is only the ordering of the scores that matters, not their quantitative values. Consequently, instead of scoring individual data points, a conformal predictor only needs to be able to compare pairs of data points, deciding which of them is the more conforming one. This suggests an interesting connection between CP and preference learning, in particular learning-to-rank methods, and makes CP amenable to training data in the form of (qualitative) preferences. Elaborating on this connection, we propose methods for preference-based CP and show their usefulness in real-world classification tasks.

1 INTRODUCTION

Conformal prediction (CP) (Vovk, Gammerman, and Shafer, 2022) has recently emerged as a prominent tool for uncertainty quantification in machine learning, offering formal statistical guarantees (Angelopoulos and Bates, 2022). Instead of point predictions, which do not account for potential uncertainty, conformal prediction provides set-valued predictions (e.g., subsets of class labels in classification or intervals in regression). A key advantage over other uncertainty quantification methods is its ability to provide coverage guarantees without relying on distributional assumptions, making it applicable to any predictive model. At the core of conformal prediction is the notion of a nonconformity score, a real-valued function that quantifies how atypical a (hypothetical) data point is compared to previously observed data. During inference, given a new input as a query, each possible outcome is evaluated by comparing its nonconformity to those in a reference set, determining how well it aligns with past observations. Specifically, if the nonconformity score of the new input with a given outcome ranks before a certain quantile, that outcome is included in the prediction set.

An important observation here is that the exact values of the scores do not matter; rather, it is the relative ordering of the data points in terms of their nonconformities that ultimately determines the prediction set. Indeed, any monotonic transformation of the nonconformity function that preserves the *ranking* of the scores will leave the prediction set unchanged (Vovk, Gammerman, and Shafer, 2022). In this paper, we leverage this observation to make CP amenable to preference learning (Fürnkranz and Hüllermeier, 2011). Rather than assigning a numerical score to each data point independently, one can instead focus on pairwise comparisons to determine which data points are more conforming and which are less. In other words, a conformal predictor essentially requires the ability to learn a ranking over the data, thereby establishing a natural connection to preference learning and learning-to-rank frameworks.

More concretely, we propose novel methods for learning latent nonconformity functions directly from qualitative preference data. Our approach replaces the conventional pointwise scoring mechanism with a strategy that utilizes pairwise comparisons, making it well-suited for scenarios where training data are available in the form of relative judgments. This is particularly useful in fields where human judgment is the primary source of information, often collected through comparative assessments (Yannakakis and Martínez, 2015). Analysis on data of this type has a long-

* indicates equal contribution.

standing tradition in economics and psychology, where preference information is used to analyze consumer behavior and decision making. More recently, pairwise comparisons have become integral to machine learning, too, especially for fine-tuning large language models and aligning them with human preferences (Ouyang et al., 2022). This type of feedback is also promising for eliciting the human perception of nonconformity. Indeed, humans often struggle with quantitative assessments, such as assigning precise probabilities to events (Tversky and Kahneman, 1974). Comparing two events and deciding which one is more probable, or likewise comparing two data points and saying which of them is more conforming to the data, is arguably easier. Thus, the connection between CP and preference learning provides a means of harnessing human implicit understanding of nonconformity, making it accessible for the construction of prediction sets.

In Section 2, we first review the (theoretical) foundations that underpin the relationship between conformal prediction and preference learning. In Section 3, we then introduce our method for learning nonconformity relations represented by latent nonconformity functions from preference data. Afterward, we discuss the experiments we conducted to validate the proposed method, showing that it indeed manages to induce meaningful nonconformity scores and is applicable in the context of conformal classification. Through extensive evaluations, we confirm that our approach not only preserves the formal statistical guarantees of conformal prediction but also enhances its applicability in settings where qualitative preference data are more readily available.

2 REINTERPRETING CP VIA PREFERENCE RELATIONS

Consider a set of data points $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$ drawn from an unknown distribution P. Let (X_{n+1}, Y_{n+1}) be a future test point, also drawn from P, such that the combined collection $\mathcal{D} \cup \{(X_{n+1}, Y_{n+1})\}$ is exchangeable. Assuming that the outcome Y_{n+1} for the instance X_{n+1} is unobserved, conformal prediction seeks to construct a prediction set $\mathcal{C}(X_{n+1}) \subseteq \mathcal{Y}$ such that

$$\mathbb{P}(Y_{n+1} \in \mathcal{C}(X_{n+1})) \geq 1 - \alpha, \tag{1}$$

where $\alpha \in (0, 1)$ is a user-specified error rate and the probability is taken over all samples $\mathcal{D} \cup \{(X_{n+1}, Y_{n+1})\}$.

In essence, conformal prediction tests the hypothesis $Y_{n+1} = y$ for each possible label $y \in \mathcal{Y}$. To compute p-values for this hypothesis test, CP relies on a nonconformity score $s : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$ that quantifies how atypical a new pair (x, y) is relative to the previously observed data, with higher values indicating less conformity. This score can be predefined, but is normally derived from a model fitted on $\mathcal{D}$. For example, in classification, a common choice for the nonconformity score is $s_{\text{LAC}}(x, y) := 1 - \hat{p}(y \mid x)$ (Sadinle, Lei, and Wasserman, 2019), where $\hat{p}(y \mid x)$ is the predicted probability assigned to the class y.

There are two main variants of CP: full and split conformal prediction (Papadopoulos et al., 2002; Lei et al., 2018). In full CP, the entire dataset $\mathcal{D}$ is used for model training. Specifically, for each candidate $y \in \mathcal{Y}$, a model M_y is trained on the augmented dataset $\mathcal{D} \cup (X_{n+1}, y)$, and the nonconformity scores for all $n+1$ data points are computed based on the fitted model M_y. Finally, the p-value for the hypothesis $Y_{n+1} = y$ is calculated as

$$p^y_{\text{full}} = \frac{1 + \sum_{i=1}^n \mathbb{1}\{s_{M_y}(X_i, Y_i) \geq s_{M_y}(X_{n+1}, y)\}}{n+1}, \tag{2}$$

where s_{M_y} is the nonconformity score defined based on M_y. One immediate drawback of full conformal prediction is the need to train $|\mathcal{Y}|$ models, which makes it computationally expensive. In contrast, split conformal prediction requires only a single model training step, at the cost of partitioning the data $\mathcal{D}$ into separate training set $\mathcal{D}_{\text{train}}$ and calibration set $\mathcal{D}_{\text{calib}}$. Specifically, split CP trains a single model M on $\mathcal{D}_{\text{train}}$ and then uses this model to compute nonconformity scores on the calibration set. The p-value for the hypothesis $Y_{n+1} = y$ is then calculated as:

$$p^y_{\text{split}} = \frac{1 + \sum_{i \in \mathcal{D}_{\text{calib}}} \mathbb{1}\{s_M(X_i, Y_i) \geq s_M(X_{n+1}, y)\}}{|\mathcal{D}_{\text{calib}}| + 1}, \tag{3}$$

where s_M is the score defined based on M.

Given the p-values for each label in $\mathcal{Y}$ calculated using (2) or (3), the conformal prediction set is constructed as:

$$\mathcal{C}(X_{n+1}) = \{y \in \mathcal{Y} \colon p^y \geq \alpha\}. \tag{4}$$

Regardless of the variant of conformal prediction—full or split—and the choice of the nonconformity score, an important observation is the one we already highlighted in the introduction: what ultimately matters in the calculation of p-values and the construction of the prediction set is the relative ranking of the data points, rather than the exact values of the scores. This invariance to scale is formally stated in the following lemma.

Lemma 2.1 ((Vovk, Gammerman, and Shafer, 2022))**.** *Let $s : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$ be a nonconformity score and let $\mathcal{T} : \mathbb{R} \longrightarrow \mathbb{R}$ be a strictly increasing function. For any $\alpha \in (0, 1)$, the conformal prediction set constructed from s is identical to that constructed from $\mathcal{T}(s)$.*

According to this observation, one may ask whether, instead of defining a nonconformity score based on a model (e.g., a probabilistic classifier) that is ultimately used to rank data points, it would be sufficient to directly learn a relation over pairs that ranks them without explicitly defining a score. As

Algorithm 1 Full Preference-based Conformal Prediction

Input: data $\mathcal{D}$, error rate α, test instance X_{n+1}
for each $y \in \mathcal{Y}$ **do**
  Use data $\mathcal{D} \cup (X_{n+1}, y)$ to infer relation $\succsim_{s_y}$
  Calculate p^y according to (5) given $\succsim_{s_y}$
end for
Return prediction set $\mathcal{C}(X_{n+1}) = \{y \in \mathcal{Y} \colon p^y \geq \alpha\}$

Algorithm 2 Split Preference-based Conformal Prediction

Input: data $\mathcal{D}$, error rate α, test instance X_{n+1}
Partition $\mathcal{D}$ into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{calib}}$
Use $\mathcal{D}_{\text{train}}$ to infer preference relation $\succsim_s$
For each $y \in \mathcal{Y}$, calculate p^y according to (6) given $\mathcal{D}_{\text{calib}}$
Return prediction set $\mathcal{C}(X_{n+1}) = \{y \in \mathcal{Y} \colon p^y \geq \alpha\}$

already said, this is closely related to the concept of preference learning in machine learning, where the objective is to learn a ranking over data points (Fürnkranz and Hüllermeier, 2011). More specifically, learning-to-rank methods construct a (weak) preference relation $\succsim$ that, given a pair of objects Z_i and Z_j, determines which one is preferred over (or ranked before) the other. For example, $Z_i \succsim Z_j$ suggests that object Z_i is preferred to or as good as Z_j. More specifically, if objects are data points $Z = (X, Y)$, then $(X_i, Y_i) \succsim (X_j, Y_j)$ implies that instance X_i with label Y_i is (weakly) preferred to instance X_j with label Y_j. We will return to the learning of preferences later in Section 3.

Having access to such a relation, we can modify the conformal prediction framework as follows: in the case of full CP, for each candidate $y \in \mathcal{Y}$, we use the augmented datasets $\mathcal{D} \cup (X_{n+1}, y)$ to infer a preference relation $\succsim_{s_y}$. This allows us to redefine the notion of a p-value in (2) as follows:

$$p^y_{\text{full}} = \frac{1 + \sum_{i=1}^{n} \mathbb{1}\{(X_{n+1}, y) \succsim_{s_y} (X_i, Y_i)\}}{n+1}. \quad (5)$$

Accordingly, for the split CP, we use the training data $\mathcal{D}_{\text{train}}$ to learn a single relation $\succsim_s$ and modify (3) to have

$$p^y_{\text{split}} = \frac{1 + \sum_{i \in \mathcal{D}_{\text{calib}}} \mathbb{1}\{(X_{n+1}, y) \succsim_s (X_i, Y_i)\}}{|\mathcal{D}_{\text{calib}}| + 1}. \quad (6)$$

The pseudo-code of the proposed methods, which we refer to as *preference-based* conformal prediction, is presented in Algorithms 1 and 2, respectively.

Beyond the natural connection to the notion of nonconformity, an additional advantage of this approach is its ability to learn such a ranking relation using weaker supervision, in particular, pairwise comparisons of the form $\{(X_{i_1}, Y_{i_1}) \succsim (X_{i_2}, Y_{i_2})\}_{i=1}^{n}$. This highlights the benefit of our approach compared to standard CP in scenarios where a large amount of pairwise comparisons is available, but supervision in the form of labeled data is limited, as outlined empirically in Section 4.3.

2.1 VALIDITY OF THE PROPOSED METHOD

As mentioned earlier, the desirable property of CP lies in its marginal coverage, as stated in (1). Hence, it is necessary to provide sufficient conditions under which the proposed method in the previous section still preserves this guarantee. We begin with the following definitions on a relation $\succsim$.

Definition 2.2. Consider a relation $\succsim$ defined on $\mathcal{X} \times \mathcal{Y}$.

1. **Transitivity:** $\succsim$ is transitive if and only if for all $(X_1, Y_1), (X_2, Y_2), (X_3, Y_3) \in \mathcal{X} \times \mathcal{Y}$,

$$\begin{cases} (X_1, Y_1) \succsim (X_2, Y_2) \\ \text{and} \\ (X_2, Y_2) \succsim (X_3, Y_3) \end{cases} \Rightarrow (X_1, Y_1) \succsim (X_3, Y_3).$$

2. **Completeness:** $\succsim$ is complete if and only if for all $(X_1, Y_1), (X_2, Y_2) \in \mathcal{X} \times \mathcal{Y}$, either

$$(X_1, Y_1) \succsim (X_2, Y_2) \quad \text{or} \quad (X_2, Y_2) \succsim (X_1, Y_1)$$

holds.

3. **Continuity:** $\succsim$ is continuous if and only if for every $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, the sets

$$\{(W, V) \in \mathcal{X} \times \mathcal{Y} \mid (X, Y) \succsim (W, V)\}$$

and

$$\{(W, V) \in \mathcal{X} \times \mathcal{Y} \mid (W, V) \succsim (X, Y)\}$$

are closed.

The following theorem states that a relation, under certain assumptions, can be represented by a real-valued (unknown) utility function, where the preference of one point over another can be translated into a numerical comparison of their latent utilities.

Theorem 2.3 ((Debreu, 1954))**.** *Let $\succsim$ be a binary relation on $\mathcal{X} \times \mathcal{Y}$ that is transitive, complete, and continuous. Then there exists a utility function $u : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$ such that, for all $(X_1, Y_1), (X_2, Y_2) \in \mathcal{X} \times \mathcal{Y}$,*

$$(X_1, Y_1) \succsim (X_2, Y_2) \iff u(X_1, Y_1) \geq u(X_2, Y_2). \quad (7)$$

Next, we show that if there exists more than one utility function that represents a relation, they must all be comonotonic, meaning that they must agree on the relative comparisons.

Lemma 2.4 (Comonotonicity)**.** *Let $\mathcal{U}$ be the set of utility functions representing a transitive, complete, and continuous relation $\succsim$. Then for every $u, u' \in \mathcal{U}$ and every pair $(X_1, Y_1), (X_2, Y_2) \in \mathcal{X} \times \mathcal{Y}$*

$$u(X_1, Y_1) \geq u(X_2, Y_2) \iff u'(X_1, Y_1) \geq u'(X_2, Y_2).$$

Proof. By (7), we have for each $u, u' \in \mathcal{U}$ and all $(X_1, Y_1), (X_2, Y_2) \in \mathcal{X} \times \mathcal{Y}$

$$\begin{aligned} u(X_1, Y_1) \geq u(X_2, Y_2) &\iff (X_1, Y_1) \succsim (X_2, Y_2) \\ &\iff u'(X_1, Y_1) \geq u'(X_2, Y_2). \end{aligned}$$

□

Now, we are ready to present our main theorems.

Theorem 2.5. *Let $\mathcal{D} \cup \{(X_{n+1}, Y_{n+1})\}$ be a set of exchangeable data points, and let $\{\succsim_{s_y} \colon y \in \mathcal{Y}\}$ be the set of relations as in Algorithm 1. If every $\succsim_{s_y}$ is transitive, complete, and continuous, and the algorithm used to derive them is invariant to data permutation, then the conformal prediction sets constructed using Algorithm 1 are valid.*

Proof. Consider $y \in \mathcal{Y}$. According to Theorem 2.3, let u_y be a utility function that represents $\succsim_{s_y}$. We can therefore rewrite (5) as follows:

$$p^y_{\text{full}} = \frac{1 + \sum_{i=1}^n \mathbb{1}\{u_y(X_{n+1}, y) \geq u_y(X_i, Y_i)\}}{n+1}.$$

Hence, the conformal prediction procedure provided in Algorithm 1 can be seen as the standard full CP with the conformity function u_y for each $y \in \mathcal{Y}$, which is a valid set predictor as long as $\succsim_{s_y}$ does not depend on the ordering of the data points (see Angelopoulos, Foygel Barber, and Bates, 2025, Proposition 3.8 for more details). Also, note that in the case where more than one utility function can represent the relation $\succsim_{s_y}$, Lemma 2.4 ensures that p-values remain the same. □

Theorem 2.6. *Let $\mathcal{D}_{calib} \cup \{(X_{n+1}, Y_{n+1})\}$ be a set of exchangeable data points, and let $\succsim_s$ be the relation as in Algorithm 2 that is transitive, complete, and continuous. The conformal prediction sets constructed using Algorithm 2 are valid.*

We omit the proof for this case, as it follows the same reasoning as in the previous theorem. Our results show that the preference-based CP is valid when the three conditions in Definition 2.2 are met for the preference relation. What our theorem does not account for is the case where the relation does not satisfy some of these properties. Indeed, although such a relation is easy to obtain from a preference learning perspective, it is not clear whether one can achieve valid prediction sets with it.

3 LEARNING RELATIONS FROM PREFERENCE DATA

There are two main approaches to modeling preferences on a set of objects $\mathcal{Z}$, namely in terms of (binary) preference relations $R \subseteq \mathcal{Z} \times \mathcal{Z}$ and in terms of utility functions $u : \mathcal{Z} \longrightarrow \mathbb{R}$ — informally speaking, these approaches correspond, respectively, to comparing pairs of objects and evaluating individual objects (Hüllermeier and Słowiński, 2024a; Hüllermeier and Słowiński, 2024b). Mathematically, the relational approach is more general: while a utility function induces a preference relation in a straightforward way, not every preference relation can be represented in terms of a utility function. This is also the reason why we presented our extension of CP in the previous section in terms of preference relations.

From a machine learning perspective, the relational approach is closely related to binary classification, as it comes down to learning a binary preference predicate (Rigutini et al., 2011). For example, this predicate could be realized in the form of a binary classifier $\mathcal{Z} \times \mathcal{Z} \longrightarrow \{-1, +1\}$, which accepts a tuple (Z, Z') as input and returns $+1$ if $Z \succsim Z'$ and -1 otherwise. As one disadvantage of this approach, note that a (binary) predicate trained in this way does not necessarily guarantee any specific properties (such as transitivity) of the induced preference relation. As explained before, such properties are naturally required in the context of ranking. The alternative of learning a (latent) utility, which in our case corresponds to a *conformity* function $u \colon \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$, is appealing because the preference relation $\succsim$ induced by such a function guarantees desirable properties right away. Therefore, although the relational approach would be even more in line with the idea of CP without nonconformity scores (see the discussion in Section 5), we focus on learning latent utility functions in the following.

Among the various alternatives for learning a latent utility function from preference data, we opt for a generalized Bradley-Terry (BT) model (Bradley and Terry, 1952). Under this model, the probability of a pairwise preference is modeled as

$$P(Z_i \succ Z_j) = \frac{\exp(u(Z_i))}{\exp(u(Z_i)) + \exp(u(Z_j))}. \tag{8}$$

Having access to training data $\mathcal{D}_{\text{train}} = \{(X_{i_1}, Y_{i_1}) \succ (X_{i_2}, Y_{i_2})\}_{i=1}^n$ of pairwise comparisons, model parameters of u can be learned via maximum likelihood estimation, where the negative log-likelihood function is given as

$$l(i_1, i_2) = \log\big(\exp(u(Z_{i_1})) + \exp(u(Z_{i_2}))\big) - u(Z_{i_1}), \tag{9}$$

where $Z_j = (X_j, Y_j)$. This can be used as a loss function for training models of u with standard gradient-based methods. Due to its probabilistic nature, the BT model deals gracefully with noisy preference labels and is an appropriate choice for the task of learning a preference relation for CP. It is worth mentioning that one can also incorporate standard labeled data into the BT model training by converting it into preference data. To that end, for a given labeled instance (X_i, Y_i), $|\mathcal{Y}| - 1$ pairwise comparisons between the instance with its true label and the same instance with other incorrect

Algorithm 3 Bradley-Terry Neural Network Training

Input: Training data $\mathcal{D}_{\text{train}}$, number of epochs N , learning rate η
Initialize parameters
for epoch = 1 to N **do**
  for each $(X_{i_1}, Y_{i_1}) \succ (X_{i_2}, Y_{i_2}) \in \mathcal{D}_{\text{train}}$ **do**
    $u(Z_{i_1}) \leftarrow u(X_{i_1}, Y_{i_1})$ ▷ First forward pass
    $u(Z_{i_2}) \leftarrow u(X_{i_2}, Y_{i_2})$ ▷ Second forward pass
    $Loss \leftarrow l(i_1, i_2)$ ▷ Compute loss from (9)
    $Gradients \leftarrow$ BackwardPropagation$(Loss)$
    UpdateParameters$(Gradients, \eta)$
  end for
end for

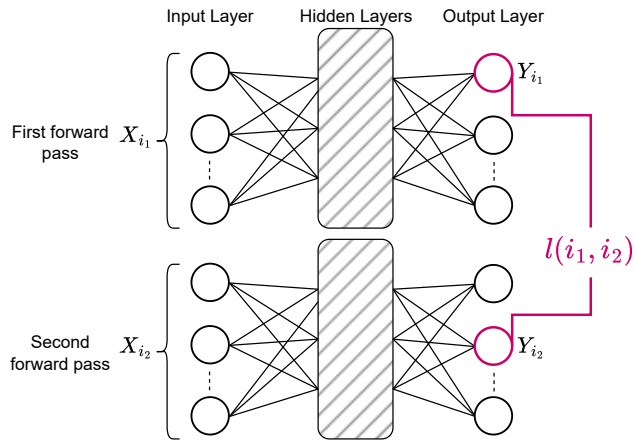

Figure 1: Illustration of the neural network architecture and loss computation. Only the outputs for the labels of the pair in the training examples (highlighted in magenta) are contributing to the loss.

labels, i.e., $\{(X_i, Y_i) \succ (X_i, y), \forall y \neq Y_i\}$, can be added to the training set. For datasets with a large number of classes, this can be done more efficiently, for example, by sampling.

There exist numerous alternatives to BT that are worth discussing. Note that the BT model assumes a clear winner for each comparison ($\succ$ instead of $\succsim$). There exist extensions that explicitly incorporate the possibility of ties, like the Davidson model (Davidson, 1970) and the extension by Rao and Kupper (Rao and Kupper, 1967). We adhere to the original BT model within this work due to its simplicity and robustness in situations with limited data. Another interesting alternative is the Plackett-Luce model (Plackett, 1975; Luce, 1959) for scenarios where preference data can be obtained in the form of full or partial rankings instead of only pairwise comparisons. In fact, the BT model is a special case of the Plackett-Luce model for rankings of length 2. While the Plackett-Luce model is more appealing from a data efficiency perspective, pairwise comparisons are typically easier to collect.

In this work, we focus on conformal classification and model u in terms of a neural network. The architecture comprises one output neuron per class label $y \in \mathcal{Y}$. Given a training example $(X_{i_1}, Y_{i_1}) \succ (X_{i_2}, Y_{i_2})$, we perform two forward passes, one for X_{i_1} and the other for X_{i_2}. The negative log-likelihood of the BT model is then computed between the output of the neuron corresponding to Y_{i_1} for input X_{i_1} in the first forward pass and the output of the neuron corresponding to Y_{i_2} for input X_{i_2} in the second forward pass. An illustration of this procedure is depicted in Figure 1. Note that neither $X_{i_1} \neq X_{i_2}$ nor $Y_{i_1} \neq Y_{i_2}$ is required. Consequently, we can learn from preferences with varying class labels for the same instance as well as preferences (for potentially the same class label) across varying instances. A description of BT neural network training in terms of pseudocode is given in Algorithm 3.

4 EXPERIMENTAL RESULTS

In this section, we will experimentally examine the proposed method, focusing on split conformal prediction as outlined in Algorithm 2 for the sake of computational efficiency. After discussing the experimental setup and baselines, we are specifically interested in answering the following research questions:

(A) Can existing nonconformity functions be replicated by the BT model from preference feedback? ($\rightsquigarrow$ Section 4.2.)

(B) Since our method is able to exploit preference data, does it bring any benefit in cases where both preference and standard labelled data are available for training? ($\rightsquigarrow$ Section 4.3)

(C) How does preference-based conformal prediction perform in real-world classification tasks? ($\rightsquigarrow$ Section 4.4.)

Setup. All of the following experiments are conducted with neural networks as the model class. The models are implemented in `PyTorch` and conformalized with `TorchCP` (Huang, Song, et al., 2024). Whenever our method is compared against a classification baseline (i.e., standard CP), both models share the same architecture, optimizer, and learning rate, and are just trained with different loss functions and a different type of data. The code that was used for carrying out the following experiments is made publicly available [1].

[1] `https://github.com/JonasHanselle/preference-based-cp`

4.1 BASELINE NONCONFORMITY SCORES

There are many nonconformity scores in the literature designed for the classification problem (Sadinle, Lei, and Wasserman, 2019; Romano, Sesia, and Candes, 2020; Angelopoulos, Bates, et al., 2021; Huang, Xi, et al., 2024). For the sake of comparison, we consider as baselines the two most commonly used ones, namely the *least ambiguous set-valued classifier* (LAC) (Sadinle, Lei, and Wasserman, 2019) and the *adaptive prediction set* (APS) (Romano, Sesia, and Candes, 2020). Let $\hat{p}(\cdot|x)$ denote the predicted conditional probability distribution over labels. The LAC score directly relates the nonconformity of a class label y to its negated (estimated) probability:

$$s_{\text{LAC}}(x, y) = 1 - \hat{p}(y \mid x). \tag{10}$$

The APS score, on the other hand, computes as nonconformity the cumulative probability of class labels that are equally or more likely than the label y:

$$s_{\text{APS}}(x, y) = \sum_{y' \in \mathcal{Y}} \hat{p}(y' \mid x) \cdot \mathbb{1}\{\hat{p}(y' \mid x) \geq \hat{p}(y \mid x)\}. \tag{11}$$

However, this score, in its current format, can result in many ties, deviating the distribution of p-values from uniformity. This, in turn, makes the CP framework more conservative, leading to coverage higher than the desired level. To address this, the randomized version of APS, aka *smoothed* APS, was introduced by modifying (11) as follows:

$$s_{\text{APS}}(x, y, \xi) = \sum_{y' \in \mathcal{Y}} \hat{p}(y' \mid x) \cdot \mathbb{1}\{\hat{p}(y' \mid x) > \hat{p}(y \mid x)\} + \xi \cdot \hat{p}(y \mid x),$$

with $\xi \sim \text{Unif}[0, 1]$. An illustration of the behavior of these scores is given in Figure 2 for a simple binary classification problem with a one-dimensional feature. Here, the true probability of each class, as well as the scores for each class given these probabilities, are plotted accordingly.

4.2 REPLICATING NONCONFORMITY SCORES FROM ORACLE FEEDBACK

Before evaluating preference-based conformal prediction, we will assess our method's capability of replicating the discussed nonconformity scores from oracle feedback. By oracle feedback, we mean that a preference relation between two pairs results from comparing their nonconformity scores. The nonconformity scores under consideration pose different challenges. As illustrated in Figure 2, the LAC score corresponds (in the binary case) to the probability of the opposite class. The APS score in its non-randomized version is simply a cumulative probability. Consequently, it exhibits plateaus in the regions where the label y is the least probable class. This poses a challenge for learning the score from pairwise data, as all comparisons within these regions result in ties (because the scores are all one). Additionally, this score has discontinuities at the decision boundaries. Due to the inherent stochasticity and global lack of smoothness, learning to replicate the randomized version of APS is conceptually intractable. Thus, we restrict ourselves to the cases of LAC and non-randomized APS.

In order to assess whether the proposed method can indeed replicate the LAC and APS nonconformity scores, we generate synthetic data with two features and two classes drawn from two multivariate Gaussian distributions for which the ground truth conditional distributions $p(\cdot|x)$ are known. This is done in a two-stage process: First, the class label c is sampled according to a prior distribution $p(y)$. Then, the features are sampled from corresponding multivariate normal distributions $X \sim \mathcal{N}(\boldsymbol{m}_c, \boldsymbol{\Sigma}_c)$, fully defined by a mean vector $\boldsymbol{m}_c$ and a covariance matrix $\boldsymbol{\Sigma}_c$. The overall posterior probability of observing a class label c given a feature vector X is thus given by

$$p(y = c \mid X) = \frac{p(X \mid y = c)p(y = c)}{\sum_{c'} p(X \mid y = c')p(y = c')},$$

where the denominator serves for normalization. Figure 3 depicts an example of data sampled from this data-generating process.

We construct preference training data $\mathcal{D}_{\text{train}}$ by drawing n instances from the data-generating process and pairing them with each of the $k = 2$ possible class labels. We compute nonconformity scores $s(x, y)$ for each of these observations and build all $\binom{n \cdot k}{2}$ ordered pairs in agreement with the corresponding scores.[2] In case two observations have the same nonconformity score $s(X_i, Y_i) = s(X_j, Y_j)$, we include both possible pairs, that is, $(X_i, Y_i) \succsim (X_j, Y_j)$ and its symmetric counterpart $(X_j, Y_j) \succsim (X_i, Y_i)$ in the training data. This is particularly relevant for APS, because of the presence of ties. To assess how well the learned preference relation reflects the original nonconformity score, we generate additional 100 data points from the data generating process and sort them according to s and $\succsim_s$, resulting in rankings σ_s and $\sigma_{\succsim_s}$. We compute the Kendall's Tau-b rank correlation coefficient (Kendall, 1945) between said rankings:

$$\tau_b(\sigma_s, \sigma_{\succsim_s}) = \frac{C - D}{\sqrt{((C + D + T_1) \cdot (C + D + T_2)}} \tag{12}$$

where C is the number of concordant pairs, D the number of discordant pairs and T_1 and T_2 are the number of ties in σ_s and $\sigma_{\succsim_s}$ respectively. Figure 4 shows the rank correlation as a function of the number of training instances n averaged over 5 runs.

We observe that the ranker quickly learns to replicate LAC, while it takes more training instances in order to replicate

[2]Note that we assume a noiseless oracle here, which returns the pair strictly ordered according to the nonconformity scores.

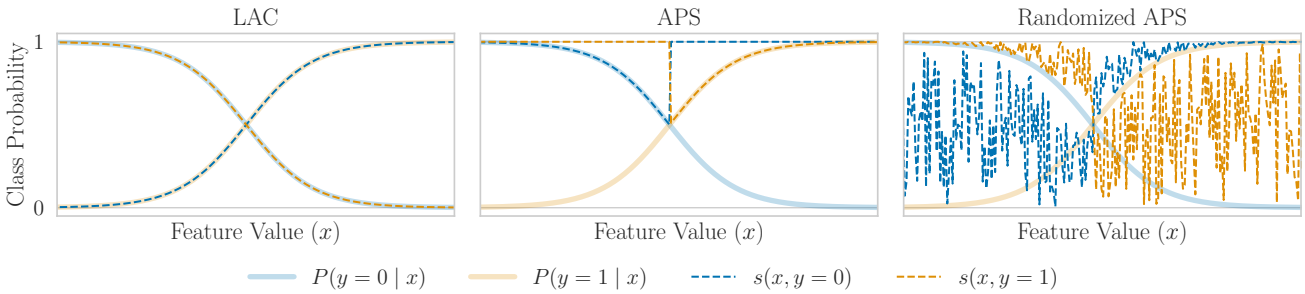

Figure 2: Illustration of the LAC and APS nonconformity scores.

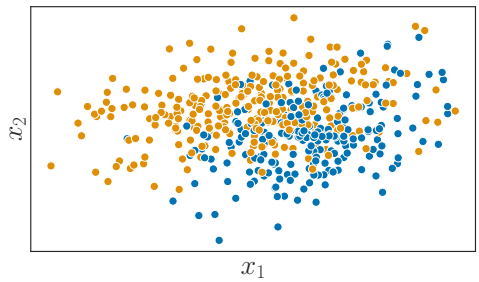

Figure 3: Example data drawn from the data-generating process described in Section 4.2.

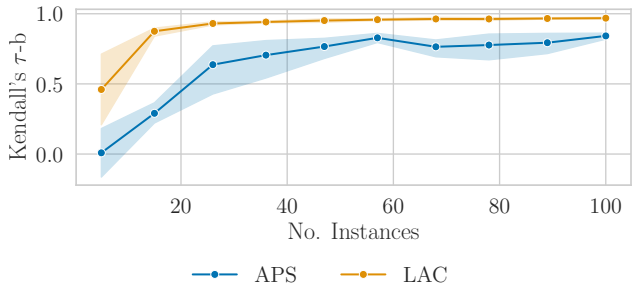

Figure 4: Rank correlation between $\mathcal{D}_{val}$ sorted according to the ground truth conformity score s and the preference relation $\succsim_s$ learned from pairwise annotations averaged over 5 runs. The shaded region indicates the 95% confidence interval.

APS. We attribute this behavior to the presence of ties in the APS training data. When two data points of the least probable class are compared, it is a tie in APS as both exhibit an APS score of 1 (see the plateaus in Figure 2). LAC, in contrast, almost always generates a strict ordering with its scores. As a result, the training data for LAC can be leveraged more effectively as each pairwise comparison contains strict order information. This structural difference is important, as the BT model assumes a winner for each comparison and does not natively handle ties. The ranking induced by APS, with its inherent ties, is therefore a poor match for the model. Hence, because Kendall's Tau-b formula accounts for ties in its denominator (see (12)), the score's theoretical maximum for the APS ranker is strictly less than 1, even if all non-tied pairs are ordered perfectly.

4.3 MIXED SETUP: EXISTENCE OF CLASSIFICATION AND PREFERENCE DATA

In the following, we demonstrate the advantage that preference-based CP has over standard CP in cases where qualitative preference data is also available that can be used by the former but not by the latter. To this end, we conduct a 3-class classification experiment involving a mixture of 100 pairwise comparison data points and a varying number of n labeled data points, with n ranging from 10 to 150. The data-generating process follows the setup used in Section 4.2. The pairwise comparisons are constructed by drawing two samples and the preference is determined based on their conditional probabilities, i.e. $(X_1, Y_1) \succ (X_2, Y_2) \iff p(Y_1 \mid X_1) > p(Y_2 \mid X_2)$. We also generate 100 labeled data points for calibration and another 100 for testing. While the classifier is trained solely on the n labeled data points, our ranker is trained using both the 100 pairwise comparisons and the same n labeled data transformed into pairwise comparisons as described in Section 3. We repeated the experiment with varying n and 20 different random seeds and report the results in Figure 5.

We observe that the ranker initially has an advantage, which diminishes as more classification data becomes available. This highlights the usefulness of our approach in scenarios where a large amount of weakly supervised preference data is available, but supervision in the form of labeled classification data is limited. This, of course, comes at an extra computational cost. More specifically, in the case where we have access to m pairwise comparisons and n labeled instances, we can construct $(k-1) \cdot n$ pairwise comparisons between the correct and incorrect class labels from the labeled instances, where k is the number of classes. The overall number of forward passes required for BT model training is then in $\mathcal{O}(m + k \cdot n)$, compared to classifier training, which is $\mathcal{O}(n)$.

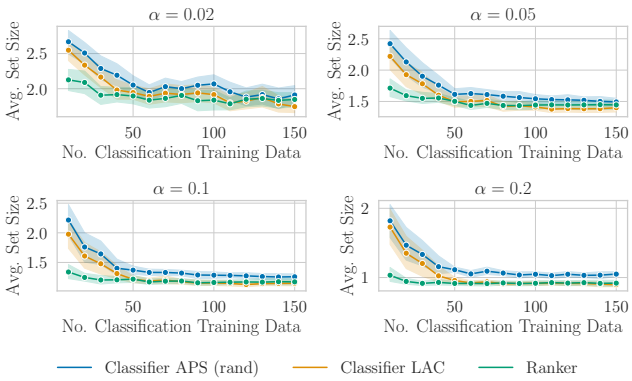

Figure 5: Average prediction set sizes of a classifier using the LAC and randomized APS nonconformity scores and a ranker for a varying number of available classification training data and different α. The results are averaged over 20 runs and the shaded region indicates the 95% confidence interval.

4.4 PREFERENCE-BASED CONFORMAL PREDICTION IN CLASSIFICATION

In the following, we will examine the applicability of our method for the task of tabular classification. To this end, we utilize benchmark datasets from the OpenML database (Vanschoren et al., 2013), which are summarized in Table 1. For all experiments, we learn a feed-forward neural network with two hidden layers of 20 units each. For each dataset, we train a classifier model with cross-entropy loss on the original training data as a baseline. In order to learn a ranking model, we transfer the classification data into preference data by deriving all pairs between the true and wrong labels for each observation in the training set, as described in Section 3. This yields a preference dataset consisting of all $(k-1) \cdot n$ pairwise comparisons, where k is the number of classes and n the number of instances in the original dataset.

We employ a 10-times Monte Carlo cross-validation, in which we reserve a fraction of $\frac{1}{5}$ of the original dataset as test data. From the remaining data, we reserve $\frac{1}{4}$ of the data points as calibration data for split conformal prediction and use the rest for training the classifier and ranker. Results for various calibration set sizes are given in Appendix A. We employ the LAC and the randomized APS nonconformity scores for building conformal prediction sets with the classifier.

The results for the tabular classification are summarized in Figure 6, and a detailed tabular representation is given in Appendix B. We report the empirical coverage rate

$$\frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{i \in \mathcal{D}_{\text{test}}} \mathbb{1}\{Y_i \in \mathcal{C}(X_i)\}$$

Table 1: Overview of OpenML Datasets.

ID	Name	# Feat.	# Classes	# Inst.
15	`breast-w`	10	2	699
31	`credit-g`	21	2	1000
4534	`PhishingWebsites`	31	2	11055
61	`iris`	5	3	150
187	`wine`	14	3	178
54	`vehicle`	19	4	846
35	`dermatology`	35	6	366

and average set size

$$\frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{i \in \mathcal{D}_{\text{test}}} |\mathcal{C}(X_i)|$$

averaged over the 10 cross-validation folds for different error rates $\alpha \in \{0.02, 0.05, 0.1, 0.2\}$, where $\mathcal{D}_{\text{test}}$ denote the test set. Additionally, we report the accuracy of the model in a non-conformalized context, where class predictions are given by the argmax of the probabilities returned by the classifier. For computing the accuracy of the ranker, the class that has the lowest latent nonconformity is predicted.

For the datasets under consideration, we observe a similar performance between the conventional CP methods using a classifier equipped with a nonconformity score and the preference-based CP using the ranker. Interestingly, even the accuracies coincide for `breast-w`, `credit-g`, and `PhishingWebsites` datasets. While this may be surprising at first glance, note that the BT model can be seen as a special case of logistic regression, and in the case of binary classification, its negative log-likelihood corresponds to the cross-entropy loss. Additionally, both the classifier and ranker share the same neural architecture and optimization procedure. Albeit the ranker receives more training data (at least in the multi-class case), the information is also identical to the classification data: The decomposition of one classification data point into $k-1$ preferences uniquely identifies the true class label but carries no further information.

For the CP-specific evaluation metrics, we see that the preference-based CP achieves comparable performance to the conventional CP. The ranker fulfills the specified coverage rates while not producing larger prediction sets than the other approaches within acceptable statistical fluctuations. We conclude that for tabular classification tasks, a ranker that inferred a nonconformity relation achieves comparable performance to the conventional approach of learning a probabilistic classifier and conformalizing it via a specific nonconformity function.

5 DISCUSSION

In this work, we established a connection between conformal prediction and preference learning. We have shown

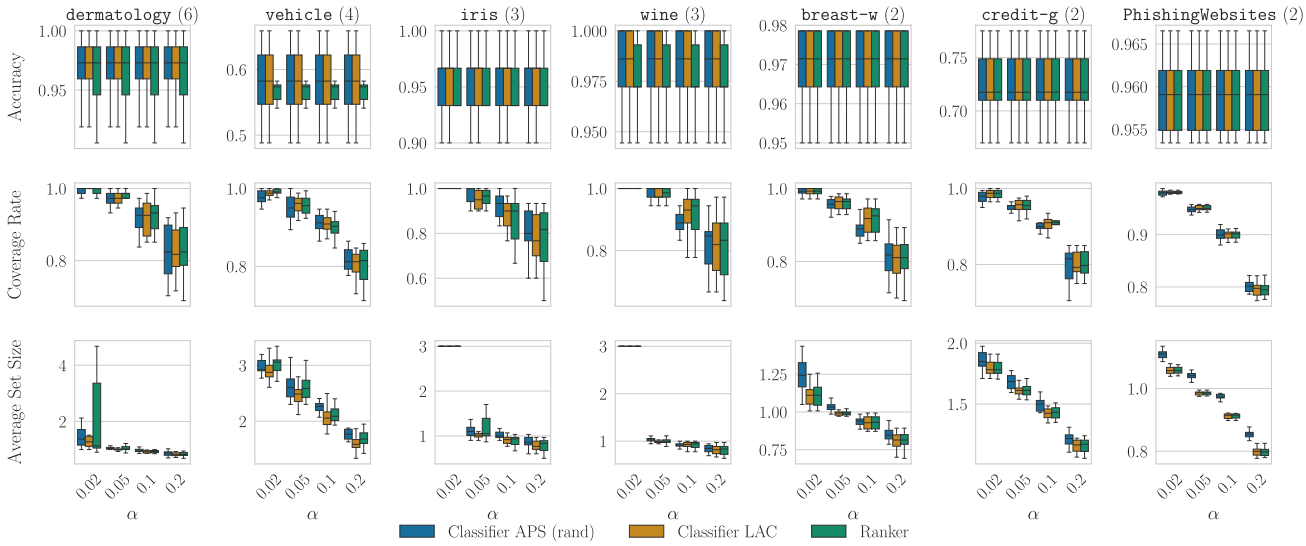

Figure 6: Accuracy, coverage rate, and average set size of the baselines APS and LAC as well as the ranking method.

that nonconformity functions can be equivalently replaced with preference relations, suggesting the use of PL methods to construct valid prediction sets without nonconformity scores. Building upon this observation, we proposed a concrete method for directly inferring nonconformity functions from preference data based on the Bradley-Terry model. The experiments carried out empirically validate that

1. established nonconformity functions can be replicated by our method, given appropriate data;
2. when preference data is abundant but classification data is scarce, preference-based CP is advantageous over standard CP;
3. preference-based CP attains a level of performance on par with standard CP on downstream classification tasks.

These results highlight the potential for deriving valid conformal prediction sets solely from preferential feedback.

Limitations and future work. While our proposed methods are promising, they are not without limitations. First, the calibration data must still consist of observations $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, which restricts the approach to datasets where such structured inputs and labels are available. Another (potential) limitation is that our current work has been confined to (multi-class) classification problems, so one immediate extension would be to adapt it to the regression setting. Additionally, incorporating dyad-ranking techniques (Schäfer and Hüllermeier, 2018) appears to be a promising future direction, as this could enable zero-shot predictions.

Last but not least, it would be interesting to explore the "genuinely relational" approach to preference learning, which is even more in the spirit of doing CP without nonconformity scores. Thus, instead of expressing a preference relation R through a latent utility (nonconformity) function, this relation could be learned more directly, essentially by training a classifier to predict pairwise preferences. There are (at least) two ways in which this problem could be tackled. First, the learning procedure could assure that R fulfills all desired mathematical properties (completeness, transitivity, continuity), so that Theorems 2.5 and 2.6 apply. This is challenging from a (preference) learning point of view, as it means, for example, that individual pairwise comparisons cannot be predicted independently of each other. Second, one may allow the learning procedure to produce relations R of more general nature, which may violate some of the properties. While this simplifies the learning part, CP cannot be done in the standard way anymore, so this approach requires a generalization of conformal prediction (e.g., making it amenable to partial order relations on data points).

Acknowledgements

This work has received funding from the European Union's Horizon Europe Research and Innovation Programme under the Marie Sklodowska-Curie grant agreement No 101073307. Alireza Javanmardi was supported by the Klaus Tschira Stiftung. Yusuf Sale was supported by the DAAD program Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research.

Co-funded by the European Union

REFERENCES

Angelopoulos, A. N. and S. Bates (2022). *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*. arXiv: 2107.07511.

Angelopoulos, A. N., R. Foygel Barber, and S. Bates (2025). *Theoretical Foundations of Conformal Prediction*. arXiv: 2411.11824.

Angelopoulos, A. N., S. Bates, M. I. Jordan, and J. Malik (2021). "Uncertainty Sets for Image Classifiers using Conformal Prediction." In: *9th International Conference on Learning Representations (ICLR)*. OpenReview.net. URL: https://openreview.net/forum?id=eNdiU%5C_DbM9.

Bradley, R. A. and M. E. Terry (1952). "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons." In: *Biometrika* 39.3/4, p. 324. DOI: 10.2307/2334029.

Davidson, R. R. (1970). "On Extending the Bradley-Terry Model to Accommodate Ties in Paired Comparison Experiments." In: *Journal of the American Statistical Association* 65.329, pp. 317–328. DOI: 10.2307/2283595.

Debreu, G. (1954). "Representation of a preference ordering by a numerical function." In: *Decision Processes* 3, pp. 159–165.

Fürnkranz, J. and E. Hüllermeier (2011). *Preference Learning*. SpringerLink Books. Springer. DOI: 10.1007/978-3-642-14125-6.

Huang, J., J. Song, X. Zhou, B. Jing, and H. Wei (2024). *TorchCP: A Python Library for Conformal Prediction*. arXiv: 2402.12683.

Huang, J., H. Xi, L. Zhang, H. Yao, Y. Qiu, and H. Wei (2024). "Conformal Prediction for Deep Classifier via Label Ranking." In: *41st International Conference on Machine Learning, (ICML)*.

Hüllermeier, E. and R. Słowiński (2024a). "Preference learning and multiple criteria decision aiding: Differences, commonalities, and synergies – Part I." In: *4OR*. DOI: 10.1007/s10288-023-00560-6.

– (2024b). "Preference learning and multiple criteria decision aiding: Differences, commonalities, and synergies – Part II." In: *4OR*. DOI: 10.1007/s10288-023-00561-5.

Kendall, M. G. (1945). "The Treatment of Ties in Ranking Problems." In: *Biometrika* 33.3, pp. 239–251. DOI: 10.2307/2332303.

Lei, J., M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman (2018). "Distribution-free predictive inference for regression." In: *Journal of the American Statistical Association* 113.523, pp. 1094–1111.

Luce, R. D. (1959). *Individual choice behavior.* Individual Choice Behavior. Wiley.

Ouyang, L., J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe (2022). "Training language models to follow instructions with human feedback." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., pp. 27730–27744.

Papadopoulos, H., K. Proedrou, V. Vovk, and A. Gammerman (2002). "Inductive confidence machines for regression." In: *Machine Learning: ECML 2002: 13th European Conference on Machine Learning*. Springer.

Plackett, R. L. (1975). "The Analysis of Permutations." In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 24.2, pp. 193–202. ISSN: 00359254, 14679876.

Rao, P. V. and L. L. Kupper (1967). "Ties in Paired-Comparison Experiments: A Generalization of the Bradley-Terry Model." In: *Journal of the American Statistical Association* 62.317, pp. 194–204. DOI: 10.2307/2282923.

Rigutini, L., T. Papini, M. Maggini, and F. Scarselli (2011). "SortNet: Learning to Rank by a Neural Preference Function." In: *IEEE Transactions on Neural Networks* 22.9, pp. 1368–1380.

Romano, Y., M. Sesia, and E. Candes (2020). "Classification with Valid and Adaptive Coverage." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 3581–3591.

Sadinle, M., J. Lei, and L. Wasserman (2019). "Least ambiguous set-valued classifiers with bounded error levels." In: *Journal of the American Statistical Association*.

Schäfer, D. and E. Hüllermeier (2018). "Dyad ranking using Plackett-Luce models based on joint feature representations." In: *Machine Learning* 107.5, pp. 903–941. DOI: 10.1007/S10994-017-5694-9.

Tversky, A. and D. Kahneman (1974). "Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty." In: *Science* 185.4157, pp. 1124–1131.

Vanschoren, J., J. N. v. Rijn, B. Bischl, and L. Torgo (2013). "OpenML: networked science in machine learning." In: *SIGKDD Explorations* 15.2, pp. 49–60. DOI: 10.1145/2641190.2641198.

Vovk, V., A. Gammerman, and G. Shafer (2022). *Algorithmic Learning in a Random World*. Springer Nature.

Yannakakis, G. N. and H. P. Martínez (2015). "Ratings are Overrated!" In: *Frontiers ICT* 2, p. 13. DOI: 10.3389/FICT.2015.00013.

A INFLUENCE OF CALIBRATION SET SIZE

In order to assess the influence of the calibration set size on the performance of preference-based conformal prediction, we conducted a small experimental study using the binary classification dataset `PhishingWebsites` and the multi-class dataset `vehicle`. We reserved varying portions of the original training data for calibration, ranging from 10% to 50%, and used the remainder for training the ranking model. Apart from this, the experimental setup is identical to the one used for the classification experiments in Section 4.4. We report our results in Table 2. Overall, there is no clear tendency that increasing the calibration set size is beneficial. Using smaller portions of calibration data allows the learner to use more data for training, which typically results in a better classification performance.

Table 2: Performance of Preference-based Conformal Prediction on Classification Tasks with a Varying Percentage of Training Data Reserved for Calibration, Ranging from 10% to 50%.

Dataset	α	Calibration Data	Accuracy	Coverage Rate	Avg. Set Size
`PhishingWebsites` (2)	0.02	10%	$\mathbf{0.962 \pm 0.003}$	$\mathbf{0.983 \pm 0.004}$	$\mathbf{1.055 \pm 0.010}$
		20%	0.960 ± 0.007	0.981 ± 0.006	1.055 ± 0.010
		30%	0.958 ± 0.006	0.982 ± 0.004	1.062 ± 0.011
		40%	0.957 ± 0.004	0.983 ± 0.003	1.071 ± 0.011
		50%	0.955 ± 0.007	0.981 ± 0.004	1.068 ± 0.009
	0.05	10%	$\mathbf{0.962 \pm 0.003}$	0.953 ± 0.009	$\mathbf{0.984 \pm 0.013}$
		20%	0.960 ± 0.007	0.953 ± 0.007	0.987 ± 0.010
		30%	0.958 ± 0.006	0.953 ± 0.006	0.990 ± 0.008
		40%	0.957 ± 0.004	$\mathbf{0.954 \pm 0.005}$	0.995 ± 0.009
		50%	0.955 ± 0.007	0.952 ± 0.007	0.995 ± 0.006
	0.1	10%	$\mathbf{0.962 \pm 0.003}$	0.902 ± 0.013	$\mathbf{0.912 \pm 0.016}$
		20%	0.960 ± 0.007	0.903 ± 0.010	0.915 ± 0.012
		30%	0.958 ± 0.006	0.903 ± 0.009	0.916 ± 0.009
		40%	0.957 ± 0.004	$\mathbf{0.904 \pm 0.009}$	0.919 ± 0.009
		50%	0.955 ± 0.007	0.903 ± 0.007	0.919 ± 0.008
	0.2	10%	$\mathbf{0.962 \pm 0.003}$	$\mathbf{0.808 \pm 0.014}$	0.810 ± 0.015
		20%	0.960 ± 0.007	0.802 ± 0.012	$\mathbf{0.804 \pm 0.012}$
		30%	0.958 ± 0.006	0.802 ± 0.009	0.805 ± 0.009
		40%	0.957 ± 0.004	0.804 ± 0.015	0.808 ± 0.015
		50%	0.955 ± 0.007	0.804 ± 0.010	0.808 ± 0.010
`vehicle` (4)	0.02	10%	0.594 ± 0.045	0.989 ± 0.007	3.018 ± 0.159
		20%	$\mathbf{0.599 \pm 0.043}$	$\mathbf{0.991 \pm 0.008}$	3.069 ± 0.229
		30%	0.574 ± 0.032	0.982 ± 0.013	$\mathbf{2.902 \pm 0.143}$
		40%	0.555 ± 0.047	0.986 ± 0.013	2.994 ± 0.172
		50%	0.571 ± 0.050	0.987 ± 0.011	3.045 ± 0.178
	0.05	10%	0.594 ± 0.045	0.958 ± 0.031	$\mathbf{2.497 \pm 0.225}$
		20%	$\mathbf{0.599 \pm 0.043}$	0.960 ± 0.021	2.588 ± 0.236
		30%	0.574 ± 0.032	0.943 ± 0.030	2.542 ± 0.163
		40%	0.555 ± 0.047	0.963 ± 0.018	2.672 ± 0.199
		50%	0.571 ± 0.050	$\mathbf{0.964 \pm 0.018}$	2.692 ± 0.140
	0.1	10%	0.594 ± 0.045	0.901 ± 0.027	2.085 ± 0.161
		20%	$\mathbf{0.599 \pm 0.043}$	0.902 ± 0.024	$\mathbf{2.074 \pm 0.231}$
		30%	0.574 ± 0.032	0.893 ± 0.030	2.184 ± 0.107
		40%	0.555 ± 0.047	0.907 ± 0.023	2.233 ± 0.179
		50%	0.571 ± 0.050	$\mathbf{0.911 \pm 0.019}$	2.266 ± 0.135
	0.2	10%	0.594 ± 0.045	0.786 ± 0.042	1.605 ± 0.113
		20%	$\mathbf{0.599 \pm 0.043}$	0.804 ± 0.047	$\mathbf{1.595 \pm 0.163}$
		30%	0.574 ± 0.032	0.776 ± 0.052	1.680 ± 0.102
		40%	0.555 ± 0.047	0.805 ± 0.036	1.793 ± 0.186
		50%	0.571 ± 0.050	$\mathbf{0.819 \pm 0.027}$	1.779 ± 0.102

B DETAILED CLASSIFICATION RESULTS

In the following, we present the classification results of Section 4.4 that were summarized in Figure 6 in detail in Table 3 and Table 4.

Table 3: Detailed Classification Results

Dataset	α	Method	Accuracy	Coverage Rate	Avg. Set Size
dermatology (6)	0.02	Classifier APS	$\mathbf{0.968 \pm 0.023}$	0.985 ± 0.025	2.383 ± 1.356
		Classifier APS (rand)	$\mathbf{0.968 \pm 0.023}$	$\mathbf{0.993 \pm 0.010}$	1.889 ± 1.352
		Classifier LAC	$\mathbf{0.968 \pm 0.023}$	0.992 ± 0.022	$\mathbf{1.811 \pm 1.388}$
		Ranker	0.965 ± 0.026	0.986 ± 0.030	2.055 ± 1.476
	0.05	Classifier APS	$\mathbf{0.968 \pm 0.023}$	0.967 ± 0.033	1.845 ± 0.243
		Classifier APS (rand)	$\mathbf{0.968 \pm 0.023}$	0.970 ± 0.026	1.068 ± 0.081
		Classifier LAC	$\mathbf{0.968 \pm 0.023}$	0.969 ± 0.031	$\mathbf{1.016 \pm 0.059}$
		Ranker	0.965 ± 0.026	$\mathbf{0.974 \pm 0.032}$	1.197 ± 0.486
	0.1	Classifier APS	$\mathbf{0.968 \pm 0.023}$	0.910 ± 0.059	1.778 ± 0.253
		Classifier APS (rand)	$\mathbf{0.968 \pm 0.023}$	0.915 ± 0.038	0.968 ± 0.057
		Classifier LAC	$\mathbf{0.968 \pm 0.023}$	0.919 ± 0.047	$\mathbf{0.928 \pm 0.048}$
		Ranker	0.965 ± 0.026	$\mathbf{0.925 \pm 0.044}$	0.931 ± 0.046
	0.2	Classifier APS	$\mathbf{0.968 \pm 0.023}$	0.816 ± 0.092	1.624 ± 0.333
		Classifier APS (rand)	$\mathbf{0.968 \pm 0.023}$	0.824 ± 0.078	0.862 ± 0.088
		Classifier LAC	$\mathbf{0.968 \pm 0.023}$	$\mathbf{0.832 \pm 0.067}$	$\mathbf{0.832 \pm 0.067}$
		Ranker	0.965 ± 0.026	0.831 ± 0.071	0.833 ± 0.071
iris (3)	0.02	Classifier APS	$\mathbf{0.948 \pm 0.034}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{3.000 \pm 0.000}$
		Classifier APS (rand)	$\mathbf{0.948 \pm 0.034}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{3.000 \pm 0.000}$
		Classifier LAC	$\mathbf{0.948 \pm 0.034}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{3.000 \pm 0.000}$
		Ranker	0.940 ± 0.037	$\mathbf{1.000 \pm 0.000}$	$\mathbf{3.000 \pm 0.000}$
	0.05	Classifier APS	$\mathbf{0.948 \pm 0.034}$	0.945 ± 0.052	1.160 ± 0.113
		Classifier APS (rand)	$\mathbf{0.948 \pm 0.034}$	$\mathbf{0.957 \pm 0.067}$	1.133 ± 0.189
		Classifier LAC	$\mathbf{0.948 \pm 0.034}$	0.950 ± 0.041	$\mathbf{1.043 \pm 0.128}$
		Ranker	0.940 ± 0.037	0.955 ± 0.046	1.169 ± 0.253
	0.1	Classifier APS	$\mathbf{0.948 \pm 0.034}$	0.848 ± 0.076	0.967 ± 0.137
		Classifier APS (rand)	$\mathbf{0.948 \pm 0.034}$	$\mathbf{0.914 \pm 0.080}$	1.007 ± 0.106
		Classifier LAC	$\mathbf{0.948 \pm 0.034}$	0.888 ± 0.066	0.905 ± 0.086
		Ranker	0.940 ± 0.037	0.857 ± 0.110	$\mathbf{0.881 \pm 0.123}$
	0.2	Classifier APS	$\mathbf{0.948 \pm 0.034}$	0.764 ± 0.097	0.845 ± 0.132
		Classifier APS (rand)	$\mathbf{0.948 \pm 0.034}$	$\mathbf{0.793 \pm 0.119}$	0.855 ± 0.141
		Classifier LAC	$\mathbf{0.948 \pm 0.034}$	0.781 ± 0.111	$\mathbf{0.783 \pm 0.115}$
		Ranker	0.940 ± 0.037	0.781 ± 0.134	0.788 ± 0.138
vehicle (4)	0.02	Classifier APS	$\mathbf{0.579 \pm 0.054}$	$\mathbf{0.995 \pm 0.009}$	3.784 ± 0.314
		Classifier APS (rand)	$\mathbf{0.579 \pm 0.054}$	0.978 ± 0.017	2.989 ± 0.185
		Classifier LAC	$\mathbf{0.579 \pm 0.054}$	0.982 ± 0.019	$\mathbf{2.910 \pm 0.199}$
		Ranker	0.565 ± 0.027	0.990 ± 0.010	3.030 ± 0.167
	0.05	Classifier APS	$\mathbf{0.579 \pm 0.054}$	$\mathbf{0.971 \pm 0.022}$	3.097 ± 0.468
		Classifier APS (rand)	$\mathbf{0.579 \pm 0.054}$	0.951 ± 0.031	2.608 ± 0.229
		Classifier LAC	$\mathbf{0.579 \pm 0.054}$	0.957 ± 0.023	$\mathbf{2.484 \pm 0.215}$
		Ranker	0.565 ± 0.027	0.957 ± 0.021	2.608 ± 0.226
	0.1	Classifier APS	$\mathbf{0.579 \pm 0.054}$	0.909 ± 0.027	2.464 ± 0.122
		Classifier APS (rand)	$\mathbf{0.579 \pm 0.054}$	$\mathbf{0.911 \pm 0.025}$	2.296 ± 0.233
		Classifier LAC	$\mathbf{0.579 \pm 0.054}$	0.908 ± 0.024	$\mathbf{2.072 \pm 0.201}$
		Ranker	0.565 ± 0.027	0.900 ± 0.027	2.125 ± 0.149
	0.2	Classifier APS	$\mathbf{0.579 \pm 0.054}$	$\mathbf{0.816 \pm 0.042}$	2.027 ± 0.121
		Classifier APS (rand)	$\mathbf{0.579 \pm 0.054}$	0.810 ± 0.045	1.803 ± 0.172
		Classifier LAC	$\mathbf{0.579 \pm 0.054}$	0.800 ± 0.037	$\mathbf{1.596 \pm 0.169}$
		Ranker	0.565 ± 0.027	0.800 ± 0.051	1.684 ± 0.145

Table 4: Detailed Classification Results

Dataset	α	Method	Accuracy	Coverage Rate	Avg. Set Size
PhishingWebsites(2)	0.02	Classifier APS	$\mathbf{0.959 \pm 0.004}$	$\mathbf{0.999 \pm 0.001}$	1.987 ± 0.002
		Classifier APS (rand)	$\mathbf{0.959 \pm 0.004}$	0.980 ± 0.005	1.109 ± 0.014
		Classifier LAC	$\mathbf{0.959 \pm 0.004}$	0.980 ± 0.003	$\mathbf{1.057 \pm 0.012}$
		Ranker	$\mathbf{0.959 \pm 0.004}$	0.981 ± 0.003	1.058 ± 0.011
	0.05	Classifier APS	$\mathbf{0.959 \pm 0.004}$	$\mathbf{0.967 \pm 0.018}$	1.212 ± 0.421
		Classifier APS (rand)	$\mathbf{0.959 \pm 0.004}$	0.948 ± 0.006	1.042 ± 0.013
		Classifier LAC	$\mathbf{0.959 \pm 0.004}$	0.951 ± 0.005	$\mathbf{0.985 \pm 0.006}$
		Ranker	$\mathbf{0.959 \pm 0.004}$	0.951 ± 0.005	0.985 ± 0.006
	0.1	Classifier APS	$\mathbf{0.959 \pm 0.004}$	$\mathbf{0.951 \pm 0.033}$	1.196 ± 0.430
		Classifier APS (rand)	$\mathbf{0.959 \pm 0.004}$	0.900 ± 0.011	0.974 ± 0.013
		Classifier LAC	$\mathbf{0.959 \pm 0.004}$	0.899 ± 0.008	$\mathbf{0.912 \pm 0.008}$
		Ranker	$\mathbf{0.959 \pm 0.004}$	0.900 ± 0.008	0.912 ± 0.008
	0.2	Classifier APS	$\mathbf{0.959 \pm 0.004}$	$\mathbf{0.895 \pm 0.082}$	1.140 ± 0.463
		Classifier APS (rand)	$\mathbf{0.959 \pm 0.004}$	0.801 ± 0.011	0.854 ± 0.012
		Classifier LAC	$\mathbf{0.959 \pm 0.004}$	0.796 ± 0.015	$\mathbf{0.799 \pm 0.015}$
		Ranker	$\mathbf{0.959 \pm 0.004}$	0.796 ± 0.014	0.799 ± 0.015
breast-w(2)	0.02	Classifier APS	$\mathbf{0.967 \pm 0.014}$	$\mathbf{0.999 \pm 0.003}$	1.947 ± 0.026
		Classifier APS (rand)	$\mathbf{0.967 \pm 0.014}$	0.990 ± 0.012	1.245 ± 0.108
		Classifier LAC	$\mathbf{0.967 \pm 0.014}$	0.991 ± 0.010	$\mathbf{1.113 \pm 0.077}$
		Ranker	$\mathbf{0.967 \pm 0.014}$	0.991 ± 0.010	1.116 ± 0.082
	0.05	Classifier APS	$\mathbf{0.967 \pm 0.014}$	$\mathbf{0.966 \pm 0.029}$	1.216 ± 0.407
		Classifier APS (rand)	$\mathbf{0.967 \pm 0.014}$	0.953 ± 0.027	1.027 ± 0.043
		Classifier LAC	$\mathbf{0.967 \pm 0.014}$	0.956 ± 0.027	$\mathbf{0.984 \pm 0.030}$
		Ranker	$\mathbf{0.967 \pm 0.014}$	0.956 ± 0.028	0.984 ± 0.031
	0.1	Classifier APS	$\mathbf{0.967 \pm 0.014}$	$\mathbf{0.922 \pm 0.048}$	0.956 ± 0.040
		Classifier APS (rand)	$\mathbf{0.967 \pm 0.014}$	0.887 ± 0.034	0.935 ± 0.031
		Classifier LAC	$\mathbf{0.967 \pm 0.014}$	0.914 ± 0.038	$\mathbf{0.927 \pm 0.043}$
		Ranker	$\mathbf{0.967 \pm 0.014}$	0.915 ± 0.037	0.929 ± 0.042
	0.2	Classifier APS	$\mathbf{0.967 \pm 0.014}$	$\mathbf{0.818 \pm 0.071}$	0.851 ± 0.064
		Classifier APS (rand)	$\mathbf{0.967 \pm 0.014}$	0.814 ± 0.052	0.851 ± 0.048
		Classifier LAC	$\mathbf{0.967 \pm 0.014}$	0.808 ± 0.056	$\mathbf{0.811 \pm 0.056}$
		Ranker	$\mathbf{0.967 \pm 0.014}$	0.808 ± 0.057	$\mathbf{0.811 \pm 0.057}$
credit-g(2)	0.02	Classifier APS	$\mathbf{0.726 \pm 0.029}$	$\mathbf{0.994 \pm 0.005}$	1.969 ± 0.012
		Classifier APS (rand)	$\mathbf{0.726 \pm 0.029}$	0.978 ± 0.014	1.852 ± 0.084
		Classifier LAC	$\mathbf{0.726 \pm 0.029}$	0.985 ± 0.010	1.800 ± 0.060
		Ranker	$\mathbf{0.726 \pm 0.029}$	0.985 ± 0.010	$\mathbf{1.799 \pm 0.060}$
	0.05	Classifier APS	$\mathbf{0.726 \pm 0.029}$	$\mathbf{0.994 \pm 0.005}$	1.969 ± 0.012
		Classifier APS (rand)	$\mathbf{0.726 \pm 0.029}$	0.952 ± 0.012	1.685 ± 0.063
		Classifier LAC	$\mathbf{0.726 \pm 0.029}$	0.955 ± 0.017	$\mathbf{1.611 \pm 0.054}$
		Ranker	$\mathbf{0.726 \pm 0.029}$	0.955 ± 0.017	1.615 ± 0.054
	0.1	Classifier APS	$\mathbf{0.726 \pm 0.029}$	$\mathbf{0.994 \pm 0.005}$	1.969 ± 0.012
		Classifier APS (rand)	$\mathbf{0.726 \pm 0.029}$	0.901 ± 0.021	1.487 ± 0.071
		Classifier LAC	$\mathbf{0.726 \pm 0.029}$	0.906 ± 0.018	$\mathbf{1.424 \pm 0.045}$
		Ranker	$\mathbf{0.726 \pm 0.029}$	0.906 ± 0.021	1.432 ± 0.051
	0.2	Classifier APS	$\mathbf{0.726 \pm 0.029}$	$\mathbf{0.974 \pm 0.073}$	1.909 ± 0.220
		Classifier APS (rand)	$\mathbf{0.726 \pm 0.029}$	0.800 ± 0.044	1.210 ± 0.059
		Classifier LAC	$\mathbf{0.726 \pm 0.029}$	0.801 ± 0.032	$\mathbf{1.159 \pm 0.052}$
		Ranker	$\mathbf{0.726 \pm 0.029}$	0.801 ± 0.034	1.161 ± 0.058
wine(3)	0.02	Classifier APS	$\mathbf{0.982 \pm 0.021}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{3.000 \pm 0.000}$
		Classifier APS (rand)	$\mathbf{0.982 \pm 0.021}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{3.000 \pm 0.000}$
		Classifier LAC	$\mathbf{0.982 \pm 0.021}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{3.000 \pm 0.000}$
		Ranker	0.974 ± 0.023	$\mathbf{1.000 \pm 0.000}$	$\mathbf{3.000 \pm 0.000}$
	0.05	Classifier APS	$\mathbf{0.982 \pm 0.021}$	0.954 ± 0.059	1.379 ± 0.494
		Classifier APS (rand)	$\mathbf{0.982 \pm 0.021}$	$\mathbf{0.982 \pm 0.026}$	1.040 ± 0.056
		Classifier LAC	$\mathbf{0.982 \pm 0.021}$	0.966 ± 0.045	$\mathbf{0.988 \pm 0.067}$
		Ranker	0.974 ± 0.023	0.974 ± 0.035	1.000 ± 0.052
	0.1	Classifier APS	$\mathbf{0.982 \pm 0.021}$	0.895 ± 0.077	1.159 ± 0.245
		Classifier APS (rand)	$\mathbf{0.982 \pm 0.021}$	0.895 ± 0.038	0.923 ± 0.045
		Classifier LAC	$\mathbf{0.982 \pm 0.021}$	$\mathbf{0.915 \pm 0.068}$	0.917 ± 0.069
		Ranker	0.974 ± 0.023	0.907 ± 0.075	$\mathbf{0.911 \pm 0.079}$
	0.2	Classifier APS	$\mathbf{0.982 \pm 0.021}$	0.817 ± 0.074	1.038 ± 0.233
		Classifier APS (rand)	$\mathbf{0.982 \pm 0.021}$	$\mathbf{0.819 \pm 0.080}$	0.839 ± 0.086
		Classifier LAC	$\mathbf{0.982 \pm 0.021}$	0.817 ± 0.097	0.817 ± 0.097
		Ranker	0.974 ± 0.023	0.812 ± 0.106	$\mathbf{0.812 \pm 0.106}$