# ShaRP: Explaining Rankings and Preferences with Shapley Values

Venetia Pliatsika*
New York University
New York, NY, USA
venetia@nyu.edu

Joao Fonseca*
New York University
New York, NY, USA
jpm9748@nyu.edu

Kateryna Akhynko
Ukrainian Catholic University
Lviv, Ukraine
kateryna.akhynko@ucu.edu.ua

Ivan Shevchenko
Ukrainian Catholic University
Lviv, Ukraine
ivan.shevchenko@ucu.edu.ua

Julia Stoyanovich
New York University
New York, NY, USA
stoyanovich@nyu.edu

## ABSTRACT

Algorithmic decisions in critical domains such as hiring, college admissions, and lending are often based on rankings. Given the impact of these decisions on individuals, organizations, and population groups, it is essential to understand them—to help individuals improve their ranking position, design better ranking procedures, and ensure legal compliance. In this paper, we argue that explainability methods for classification and regression, such as SHAP, are insufficient for ranking tasks, and present ShaRP—Shapley Values for Rankings and Preferences—a framework that explains the contributions of features to various aspects of a ranked outcome.

ShaRP computes feature contributions for various ranking-specific profit functions, such as rank and top-$k$, and also includes a novel Shapley value-based method for explaining pairwise preference outcomes. We provide a flexible implementation of ShaRP, capable of efficiently and comprehensively explaining ranked and pairwise outcomes over tabular data, in score-based ranking and learning-to-rank tasks. Finally, we develop a comprehensive evaluation methodology for ranking explainability methods, showing through qualitative, quantitative, and usability studies that our rank-aware QoIs offer complementary insights, scale effectively, and help users interpret ranked outcomes in practice.

## KEYWORDS

ranking, interpretability, feature importance, Shapley values, evaluation, responsible data management, responsible AI

**PVLDB Reference Format:**
Venetia Pliatsika, Joao Fonseca, Kateryna Akhynko, Ivan Shevchenko, and Julia Stoyanovich. ShaRP: Explaining Rankings and Preferences with Shapley Values. PVLDB, 18(11): 4131 - 4143, 2025.
doi:10.14778/3749646.3749682

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/DataResponsibly/ShaRP.

*Both authors contributed equally to this research.

Proceedings of the VLDB Endowment, Vol. 18, No. 11 ISSN 2150-8097.
doi:10.14778/3749646.3749682

## 1 INTRODUCTION

Rankings produced by data-driven algorithmic systems now influence a myriad of socio-technical applications, as part of automated or semi-automated decision-making, and with direct consequences to people's lives and aspirations. An *algorithmic ranker*, or a *ranker* for short, takes a database of candidates as input and produces a permutation of these candidates as output, see Figure 1 for an example. We refer to the output of a ranker as a *ranked outcome* or simply a *ranking*. As an alternative to the full permutation, the best-ranked $k$ candidates, or the top-$k$, may be returned in rank order or as a set. In the latter case, we are dealing with a *selection* task, which is a special case of ranking.

Algorithmic rankers are broadly used to support decision-making in critical domains, including hiring and employment, school and college admissions, credit and lending, and, of course, college ranking. Because of the impact rankers have on individuals, organizations, and population groups, there is a need to understand them: to know whether the decisions are correct and legally compliant (*auditing* tasks), to help individuals improve their ranked outcomes (*recourse* tasks), and to design better ranking procedures (*design* tasks). To make progress towards these tasks, we need ways to explain and interpret ranked outcomes. In this paper, we present *ShaRP —Shapley for Rankings and Preferences*—a framework that explains the contributions of features to different aspects of a ranked outcome, and that can support all these critically important tasks.

There are two types of rankers: score-based and learned. In score-based ranking, a given set of candidates is sorted on a score, which is typically computed using a simple formula, such as a weighted sum of attribute values [41]. In supervised learning-to-rank (LtR), a preference-enriched set of candidates is used to train a model that predicts rankings of unseen candidates [20]. We motivate our work using score-based rankers and return to LtR later in the paper.

Score-based rankers are often seen as "interpretable models" [30]: their scoring functions, such as $Y_1 = 0.9 \times gpa + 0.1 \times essay$ in a college admissions setting, reflect a normative, a priori notion of merit. For instance, specifying $Y_1$ asserts that *gpa* matters more than the essay, while $Y_2 = 0.1 \times gpa + 0.9 \times essay$ asserts the opposite. Yet the apparent transparency—and sense of *control over outcomes*—that such rankers afford is often misleading. Even with full knowledge of the formula, designers or decision-makers may struggle to anticipate or explain its output [22, 24]. We illustrate this with an example.

| name | gpa | sat | essay | $f$ | $g$ |
|---|---|---|---|---|---|
| Bob | 4 | 5 | 5 | 4.6 | 5 |
| Cal | 4 | 5 | 5 | 4.6 | 5 |
| Dia | 5 | 4 | 4 | 4.4 | 4 |
| Eli | 4 | 5 | 3 | 4.2 | 3 |
| Fay | 5 | 4 | 3 | 4.2 | 3 |
| Kat | 5 | 4 | 2 | 4.0 | 2 |
| Leo | 4 | 4 | 3 | 3.8 | 3 |
| Osi | 3 | 3 | 3 | 3.0 | 3 |

(a)

| $r_{\mathcal{D},f}$ |
|---|
| Bob |
| Cal |
| Dia |
| Eli |
| Fay |
| Kat |
| Leo |
| Osi |

(b)

| $r_{\mathcal{D},g}$ |
|---|
| Bob |
| Cal |
| Dia |
| Eli |
| Fay |
| Leo |
| Osi |
| Kat |

(c)

**Figure 1: (a) Dataset $\mathcal{D}$ of college applicants, scored on *gpa*, *sat*, and *essay*. (b) Ranking $r_{\mathcal{D},f}$ of $\mathcal{D}$ on $f = 0.4 \times gpa + 0.4 \times sat + 0.2 \times essay$; the highlighted top-4 candidates will be interviewed and potentially admitted. (c) Ranking $r_{\mathcal{D},g}$ on $g = 1.0 \times essay$; the top-4 coincides with that of $r_{\mathcal{D},f}$, signifying that *essay* has the highest importance for $f$, despite carrying the lowest weight in the scoring function.**

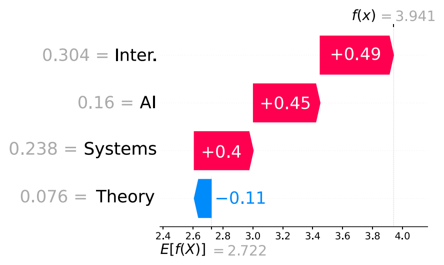
Example 1. *Consider a dataset $\mathcal{D}$ of college applicants in Figure 1, with scoring features gpa, sat, and essay. Very different scoring functions $f = 0.4 \times gpa + 0.4 \times sat + 0.2 \times essay$ and $g = 1.0 \times essay$ induce very similar rankings $r_{\mathcal{D},f}$ and $r_{\mathcal{D},g}$, with the same top-4 items appearing in the same order, apparently because essay is the feature that is best able to discriminate between the top-4 and the rest, and that determines the relative order among the top-4.*

This example illustrates that "intrinsically interpretable" score-based rankers do not always yield explainable outcomes. Even when both the formula and the dataset are fully known, it may be difficult to accurately anticipate how individual features influence the final ranking [22, 24]. This disconnect arises because a feature's weight in the scoring function does not necessarily correspond to its practical influence on the ranked outcome. For example, if *gpa* and *sat* scores are highly correlated, while *essay* scores are more variable and less correlated with the others, the *essay* component may exert disproportionate influence on rank positions *despite having lower nominal weight.* Conversely, a heavily weighted feature might have little effect if its values are tightly clustered across candidates.

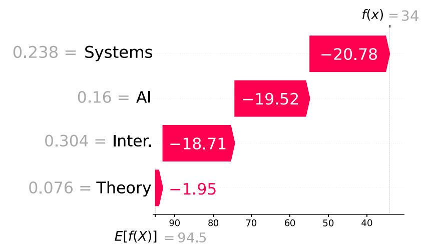
An additional nuance in ranking is that outcomes are inherently *relative*, whereas feature values and computed scores are *absolute*—an item's score reveals little about its position *relative to others.* The *lack of independence between per-item outcomes* makes feature importance methods developed for classification and regression [9, 11, 15, 21, 23, 29, 34] inadequate for ranking. These methods evaluate how a feature affects an item's score, but a feature can shift the score without altering the rank. Consider an example.

Example 2. *Consider Figure 1 and suppose that Dia's essay score increases from 4 to 5, thus increasing the scores computed with both $f$ (4.4 to 4.6) and $g$ (4 to 5). However, Dia's rank remains unchanged.*

Changes in score do not necessarily lead to changes in rank because, in selection and ranking, an item's outcome $\mathbf{v}$ depends on the outcomes of other items in $\mathcal{D} \setminus \{\mathbf{v}\}$. For example, only one item can occupy a given rank, and exactly $k$ items can appear in the top-$k$. Thus, any explainability method that measures score changes can only partially explain rank changes. This highlights that *interpretability for ranking tasks requires measuring the features' impact on quantities beyond the score*, such as rank or top-$k$ presence. We preview these results for CS Rankings in Figure 2, where feature importance for score in 2a and rank in 2b yield markedly different explanations. We discuss these findings in detail in Section 4.



**(a) Score QoI: Interdisciplinary is the most important feature, followed by AI and Systems, while Theory negatively impacts the score.**



**(b) Rank QoI: Systems is the most important feature, followed by AI and Interdisciplinary. Theory is minimally but positively impacting the rank.**

**Figure 2: Feature importance for Texas A&M in CS Rankings.**

*In summary,* ranking differs fundamentally from classification and regression, as noted in learning-to-rank and fairness-in-ranking work [20, 41, 42]. Interpretability methods must also be tailored to ranking, where scoring feature influence must account for the *interdependence of item outcomes.* We formalize and build on this insight, making four contributions.

*First,* we formalize several profit functions for computing Shapley values in ranking, capturing feature contributions to an item's score, rank, or top-$k$ presence. Building on the QII framework [11], which applies Shapley values [31] to classification, we adopt QII as a flexible foundation for defining ranked *Quantities of Interest (QoIs).*

*Second,* we propose a Shapley-based method for explaining pairwise outcomes. Unlike prior methods that use a fixed baseline [6, 21], we adapt the baseline dynamically for each pair $u \prec v$, yielding explanations that reflect relative differences.

*Third,* we release ShaRP —the first open-source library for explaining ranked outcomes over tabular data. ShaRP supports both score-based and learned rankers, includes exact and approximate QoI computation, and incorporates optimizations for scalability.

*Fourth,* we evaluate ranking explainability methods through qualitative, quantitative, and usability studies. Using established metrics, we show that rank-aware QoIs provide complementary insights beyond score-based explanations. A large-scale evaluation confirms the scalability and effectiveness of our methods, while a CS Rankings usability study shows it helps users make sense of ranked outcomes.

## 2 RELATED WORK

*Local feature-based explanations.* Ribeiro et al. [29] introduced LIME, which explains classifiers using local interpretable models. Lundberg and Lee [21] proposed SHAP, which uses Shapley values to explain predictions of classification and regression models. Both are implemented in software libraries and explain an item's score—what we refer to as the score QoI.

*Feature-based explanations for ranking.* Yang et al. [39] introduced a "nutritional label" for score-based rankers with two global explanation widgets: "Recipe" (scoring feature weights) and "Ingredients" (features with strongest rank-score correlation). They observed that a feature's weight often does not align with its correlation, highlighting the limits of global explanations. In contrast, we focus on local explanations for individual items or item pairs.

Gale and Marian [14] proposed "participation metrics" for score-based rankers, notably "weighted participation," which attributes an item's presence in the top-$k$ to its features, weights, and values. Their method aggregates over all top-$k$ items; ours provides per-item explanations using the top-$k$ QoI, which can be aggregated.

Yuan and Dasgupta [40] designed a sensitivity analysis tool for synthetic data with linear scoring, using mean-centered feature differences to approximate Shapley values. We re-implemented and extended their method to support arbitrary distributions, more features, and flexible scoring functions.

Anahideh and Mohabbati-Kalejahi [2] used local SHAP-based explanations for items near the one being explained, assuming rank stability across repeated competitions. While we also observe rank-stratum-specific feature effects, we show that small feature changes can cause large rank shifts, challenging their locality assumption.

Moskovitch et al. [25] introduced DEXER to detect group disparities in top-$k$ inclusion and explained causes via SHAP on ranks fitted by linear regression. In contrast, ShaRP fully adapts Shapley values to rank-specific QoIs. We compare with DEXER in Section 8.2.

Pastor et al. [26] used ranking-based profit functions to detect under- or overrepresented groups via attribute-level contributions, focusing on group fairness rather than individual explanations.

Hu et al. [17] proposed PrefSHAP to explain pairwise preferences in learned rankers, transforming item pairs into artificial items and applying Shapley analysis. We share the motivation for ranking-specific QoIs but target preferences induced by score-based rankers or LtR, not kernel-based preference models as in PrefSHAP.

*Shapley-based explanations in Information Retrieval (IR).* Concurrently with our work, Heuss et al. [16] and Chowdhury et al. [8] proposed Shapley-based methods for explaining ranked outcomes in IR. Both compute feature contributions for the *entire ranking* by perturbing all items simultaneously for each coalition. These methods are not applicable to settings that require explanations on a *per-item basis* (e.g., lending or hiring). In particular, Chowdhury et al. [8] define a profit function tied to query-specific rank-relevance, limiting generality. In contrast, our method supports per-item explanations while accounting for the interdependence of outcomes, using a general profit function that yields feature attributions analogous to SHAP in classification and regression. Other recent work in IR explored the use of LIME to explain ranked outcomes [7, 32, 37], and introduced baseline document construction techniques to improve explanation quality [13].

*In summary,* we share motivation with these lines of work but take a leap by presenting the first comprehensive Shapley-value-based framework for explaining rankings and pairwise preferences.

## 3 PRELIMINARIES AND NOTATION

*Ranking.* Let $\mathcal{A}$ denote an ordered collection of features (equiv. attributes), and let $\mathcal{D}$ denote a set of items (equiv. points or candidates). An item $\mathbf{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$ assigns values to $|\mathcal{A}| = d$ features, and may additionally be associated with a score. Score-based rankers use a scoring function $f(\mathbf{v})$ to compute the score of $\mathbf{v}$. For example, using $f_1(\mathbf{v}) = 0.4 \times gpa + 0.4 \times sat + 0.2 \times essay$, we compute $f(\texttt{Bob}) = 4.6$ and $f(\texttt{Leo}) = 3.8$.

A *ranking* $r_{\mathcal{D}}$ is a permutation over the items in $\mathcal{D}$. Letting $n = |\mathcal{D}|$, we denote by $r_{\mathcal{D}} = \langle \mathbf{v}_1, \ldots, \mathbf{v}_n \rangle$ a ranking that places item $\mathbf{v}_i$ at rank $i$. We denote by $r_{\mathcal{D}}(i)$ the item at rank $i$, and by $r_{\mathcal{D}}^{-1}(\mathbf{v})$ the rank of item $\mathbf{v}$ in $r_{\mathcal{D}}$. In score-based ranking, we are interested in rankings induced by some scoring function $f$. We denote these rankings $r_{\mathcal{D},f}$. For example, in Figure 1b, $r_{\mathcal{D},f}(1) = \texttt{Bob}$, $r_{\mathcal{D},f}^{-1}(\texttt{Leo}) = 7$. We assume that $r_{\mathcal{D},f}^{-1}(\mathbf{v}_1) < r_{\mathcal{D},f}^{-1}(\mathbf{v}_2) < \cdots < r_{\mathcal{D},f}^{-1}(\mathbf{v}_n)$, where smaller rank means better position in the ranking.

We are often interested in a sub-ranking of $r_{\mathcal{D},f}$ containing its best-ranked $k$ items, for some integer $k \leq n$, called the top-$k$. The top-4 of the ranking in Figure 1b is $\langle \texttt{Bob}, \texttt{Cal}, \texttt{Dia}, \texttt{Eli} \rangle$.

Our goal is to explain the importance of features $\mathcal{A}$ to the ranking $r_{\mathcal{D},f}$. We will do so using Shapley values [31].

*Shapley values.* For a set $\mathcal{N}$ of $n$ players, and a value function $f$ that assigns a profit to any subset (or coalition) $\mathcal{S}$ of players, $f : 2^n \to \mathbb{R}$, where $f(\emptyset) = 0$, the Shapley value of player $i$ is:

$$\phi_i(f) = \sum_{\mathcal{S}} \frac{|\mathcal{S}|!(n - |\mathcal{S}| - 1)!}{n!} (f(\mathcal{S} \cup \{i\}) - f(\mathcal{S})) \qquad (1)$$

We will use Shapley values to explain ranked outcomes using the set of features $\mathcal{A}$ as the players, and the outcome (or the quantity of interest, QoI) as the payoff function. In addition to the definition of players and the payoff function, Shapley values require the quantification of the payoff over a subset of the players. This, in turn, requires some way to estimate the payoff over a subset of the features. Consequently, for any Shapley value implementation, a method of feature removal or masking is required [6, 10].

A common method (e.g., used in SHAP [10, 21]), for a *coalition* (subset of features) $\mathcal{S} \subseteq \mathcal{A}$, is to marginalize out the features not in the coalition $\mathcal{A} \setminus \mathcal{S}$ and draw values from the marginal distributions of the subset of features in $\mathcal{S}$ jointly, often referred to as the "marginal" approach. Another alternative (e.g., used in QII [11]) is to draw values of each feature in $\mathcal{S}$ independently from its marginal distribution, often referred to as the "product of marginals" approach. Another approach is called "baseline" and instead of sampling the features not in the coalition, they are replaced with the feature values of a specific fixed sample [21]. Here, we choose the marginal approach for our implementation and take inspiration from the baseline approach for our pairwise method. In Section 7, we show how both can be implemented using one algorithm.

Let $\mathbf{v}_{\mathcal{S}}$ denote a projection of $\mathbf{v}$ onto $\mathcal{S}$. In the example in Figure 1, $(\texttt{Bob}, 4, 5, 5)_{\{name,gpa\}} = (\texttt{Bob}, 4)$. We define a random variable $\mathbf{U}$ that draws values from the marginal distributions of the subset

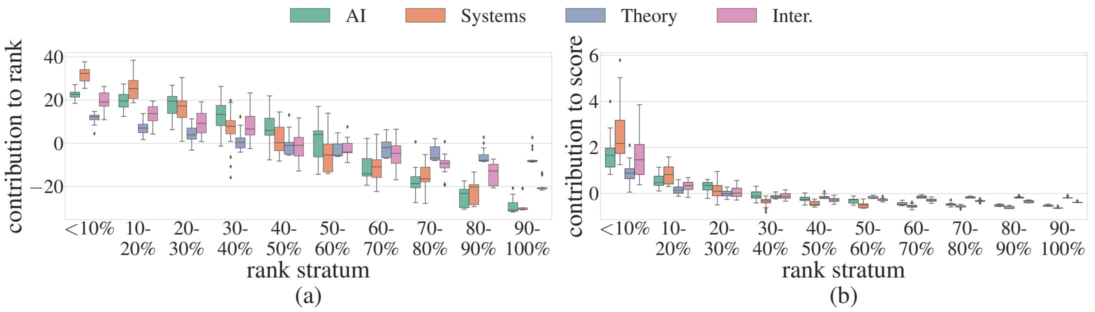


**Figure 3: Feature contributions to rank and score for the CSRankings dataset, aggregated over 10% strata. In this ranking, 189 computer science departments are ranked based on a normalized publication count of the faculty across 4 research areas: AI (green), Systems (orange), Theory (purple), and Interdisciplinary (pink). (a) Systems is the most important feature for an item's rank in the top-20%, followed by AI. AI becomes more important for the rest of the ranking strata. (b) Feature contributions to score are less informative than to rank: both capture the same relative feature importance for the top 20%; however, feature contributions become small and very similar as more items are tied for their score. (See rank vs. score plot on the top-right.)**

of features in $\mathcal{S}$. Let $\mathbf{U} = \langle \mathbf{u}_1, \ldots, \mathbf{u}_m \rangle$ denote a vector of $m$ items sampled from $\mathcal{D}$ using this method. For a subset of features $\mathcal{S} \in \mathcal{A}$, let $\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{U}_{\mathcal{S}} = \langle \mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}(\mathbf{u}_1)_{\mathcal{S}}, \ldots, \mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}(\mathbf{u}_m)_{\mathcal{S}} \rangle$ denote a vector of items, in which each $\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}(\mathbf{u}_i)_{\mathcal{S}}$ takes on the values of the features in $\mathcal{S}$ from $\mathbf{u}_i$, and the values of the remaining features $\mathcal{A} \setminus \mathcal{S}$ from $\mathbf{v}$. We calculate Shapley values using this set of features $\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{U}_{\mathcal{S}}$, note that if $m = |\mathcal{D} - 1|$ we use the entire dataset $\mathcal{D} \setminus \mathbf{v}$ to calculate the *exact* Shapley values.

Shapley values satisfy several natural axioms, including efficiency, symmetry, dummy, and additivity [31], with additional useful properties, such as monotonicity, following from these axioms[21]. Efficiency states that the sum of the contributions of all features for item $\mathbf{v}$ equals the difference between the outcome $f(\mathbf{v})$ and the average outcome: $\sum_{i \in \mathcal{A}} \varphi_i(f, \mathbf{v}) = f(\mathbf{v}) - \mathbb{E}_{\mathbf{X}}[f(\mathbf{X})]$ [9, 24]. Using this property, explanation can be used to reconstruct the outcome. We will use the efficiency property to define the fidelity metric for comparing explanations (Section 6).

## 4 QUANTITIES OF INTEREST FOR RANKING

The first contribution of our work is that we define QoIs that are appropriate for ranked outcomes. In addition to the expected score, we introduce rank and top-$k$ QoI. We use the notation for the marginal feature removal approach in this section, but note that the QoIs we introduce can be used with any feature removal approach.

*Score QoI.* The Shapley value function for the score QoI is:

$$QoI_{f,\mathbf{v}}(S) = \mathop{\mathbb{E}}_{\mathbf{U}_{\mathcal{S}}}[f(\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{U}_{\mathcal{S}})] \tag{2}$$

This QoI captures the impact of an item's features on its score. This is the QoI used by the popular feature-based explanation methods such as SHAP [21] and LIME [28]. To get the contribution of a set of features $\mathcal{A} \setminus \mathcal{S}$, we take the expected value of the score over a random variable $\mathbf{U}_{\mathcal{S}}$ that draws values from the marginal distributions of the set of features in $\mathcal{S}$.

*Rank QoI.* The Shapley value payoff function for the rank QoI is:

$$QoI_{f,\mathbf{v},\mathcal{D}}(S) = \mathop{\mathbb{E}}_{\mathbf{U}_{\mathcal{S}}}[r^{-1}_{\mathcal{D}',f}(\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{U}_{\mathcal{S}})] \tag{3}$$

where $\mathcal{D}'$ is $\mathcal{D} \cup \{\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{U}_{\mathcal{S}}\} \setminus \mathbf{v}$. This QoI evaluates the impact of an item's features on its rank. To get the contribution of a set of features $\mathcal{A}\setminus\mathcal{S}$, we take the expected value of the rank over a random variable $\mathbf{U}_{\mathcal{S}}$ that draws values from the marginal distributions of the set of features in $\mathcal{S}$.

*Top-k QoI.* The Shapley value payoff function to quantify the impact of an item's features on its presence or absence among the top-$k$ is stated similarly as rank QoI:

$$QoI_{f,\mathbf{v},\mathcal{D}}(S) = \mathop{\mathbb{E}}_{\mathbf{U}_{\mathcal{S}}}[\mathbb{1}_{r_{\mathcal{D}',f}(1\ldots k)}(\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{U}_{\mathcal{S}})] \tag{4}$$

where $\mathcal{D}'$ is $\mathcal{D} \cup \{\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{U}_{\mathcal{S}}\} \setminus \mathbf{v}$. The difference with rank QoI (Equation 3) is that here we compute the expectation over the indicator function that returns 1 if an items' rank is at most $k$ and 0 otherwise. This QoI allows us to quantify how each feature contributed to getting the item into the top-$k$.

*Shapley values for ranking.* To compute Shapley values for the QoIs we defined, we need to apply Equation 1 on the QoIs. Following the QII notation, we define the iota function $\iota$ as the difference between the QoI including feature $i$ and excluding it.

$$\iota_{f,\mathbf{v},\mathcal{D}}(i, \mathcal{S}) = \alpha(QoI_{f,\mathbf{v},\mathcal{D}}(\mathcal{S} \cup i) - QoI_{f,\mathbf{v},\mathcal{D}}(\mathcal{S})) \tag{5}$$

Here, the QoI can be any defined earlier in this section, and $\alpha \in \{-1, 1\}$ is a multiplier that adjusts the order of QoI terms. In this work, we consider QoIs beyond the score. For some, like rank, where smaller values are preferable, we set $\alpha = -1$ to adjust the $\iota$ function accordingly.

Using this notation, we can define Shapley values for ShaRP:

$$\phi_i(f, \mathbf{v}, \mathcal{D}) = \sum_{\mathcal{S}} \frac{|\mathcal{S}|!(n - |\mathcal{S}| - 1)!}{n!} \iota_{f,\mathbf{v},\mathcal{D}}(i, \mathcal{S}) \tag{6}$$

*Case Study: QoIs for CSRankings.* We review local feature-based explanations generated by ShaRP for CS Rankings, a real dataset ranking 189 U.S. Computer Science departments based on normalized faculty publication counts in four areas: AI, Systems, Theory, and Interdisciplinary [3]. See the full version [27] for dataset and ranker details. Our goal is to illustrate how ShaRP reveals meaningful insights about the data—and how those insights vary depending on the outcome being explained.

Figure 3 shows feature contributions to the rank and score QoIs for CS Rankings, aggregated by 10% rank strata. As shown in Figure 3a, Systems is the most important feature across all strata, followed by AI. Both contribute most positively in the top strata and most negatively in the bottom. *Score-based explanations are less informative*: while they capture similar relative importance in the top 20%, feature contributions flatten in lower strata, where many departments have near-tied scores, making comparisons difficult.

Figure 4a presents aggregated feature contributions to the top-$k$ QoI, stratified by deciles. Systems again dominates in placing departments in the top-10, followed by AI. This trend is consistent with Figure 3b (score QoI), but more pronounced. Unlike the score QoI, the top-$k$ QoI also highlights Theory as impactful for top-$k$ inclusion. Notably, only the rank and top-$k$ QoIs capture a shift in relative importance between Systems and AI across strata.

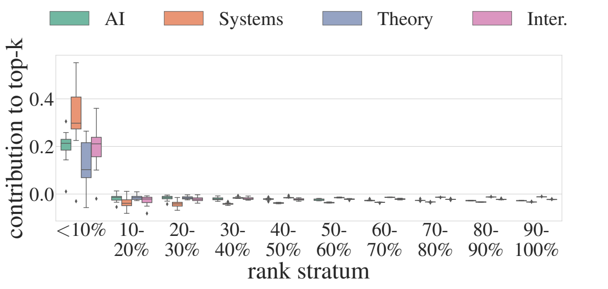
Figure 2, previewed in the Introduction, shows a local explanation for Texas A&M, ranked 34th with a score of 3.941. Waterfall plots in Figures 2a (score QoI) and 2b (rank QoI) break down feature contributions relative to the mean outcome $\mathbb{E}[f(X)]$. In Figure 2a, Interdisciplinary is the top contributor to Texas A&M's score, followed by AI and Systems; Theory contributes negatively. For rank QoI, all features contribute positively, with Systems as the most impactful. This illustrates that different QoIs support different goals. To improve score, Texas A&M should focus on Interdisciplinary and AI. To improve rank, prioritizing Systems is more effective. The difference arises because *increases in score do not always translate to changes in rank*—a score must exceed that of the next-highest item to affect position.

Another key aspect of these plots is the color of each feature, which indicates whether a feature contributes positively or negatively to the outcome. This is determined by the average feature value. Since the average score is influenced by outliers, while rank is not, the interpretation of contributions varies depending on the QoI. For example, in CS Rankings, over 70% of departments have scores below the mean. As a result, when using the score QoI, many or all of their features appear to contribute negatively. This highlights that the meaning of positive and negative contributions is dependent on the chosen QoI. See Figure 9b and Appendix C for the score vs. rank distribution for this dataset, and a more detailed comparison between the score-QoI-based and the rank-QoI-based explanations for CS Rankings.

## 5 PAIRWISE EXPLANATIONS

We developed a method for computing feature importance for the relative order between a pair of items $\mathbf{u}$ and $\mathbf{v}$, to answer the question of why $\mathbf{v}$ is ranked higher than $\mathbf{u}$ (i.e., $\mathbf{v} \succ \mathbf{u}$). Our method is based on baseline Shapley value methods.

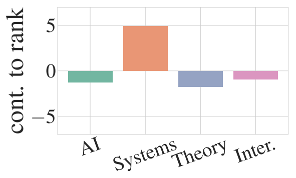
In Eq. 1 we provided the definition of game-theoretic Shapley values. This equation uses a profit function defined over subsets $\mathcal{S}$ of the players. In the ML context, we use methods that take as input all features (players) - not a subset. Different Shapley value methods in ML take different approaches for addressing this problem, often referred to as the "feature removal approach" in the literature [6].



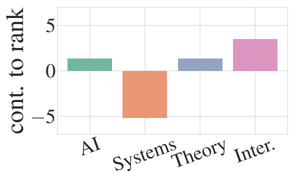
**(a) Feature contribution to the top-$k$ QoI, for $k = 10\%$. Systems is the most important feature, followed by Interdisciplinary and AI.**

| Institution | AI | Systems | Theory | Inter. | Rank |
|---|---|---|---|---|---|
| Georgia Tech | 28.5 | 7.8 | 6.9 | 10.2 | 5 |
| Stanford | 36.7 | 5.4 | 13.3 | 11.5 | 6 |
| UMich | 30.4 | 9.0 | 9.3 | 5.9 | 7 |

**(b) Feature values and rank of three highly ranked departments: Georgia Tech, Stanford, and UMich.**



**(c) Pairwise QoI: Georgia Tech ranks higher than Stanford because of its relative strength in Systems.**



**(d) Pairwise QoI: Stanford ranks higher than UMich despite Stanford's relative weakness in Systems.**

**Figure 4: Feature importance for the top-$k$ QoI (i.e., selection) for CS Rankings in 4a, with further analysis of the relative orders among two pairs of departments in 4c and 4d.**

One feature removal method is creating hybrid samples using the marginal distributions of the missing features and drawing values jointly. In Sec. 3 we took this approach. We defined the items that we will be using in the Shapley value computations when using this marginal approach as $\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{U}_{\mathcal{S}}$ where $\mathbf{U} = \langle \mathbf{u}_1, \ldots, \mathbf{u}_m \rangle$ is a vector of $m$ items sampled from $\mathcal{D}$.

For pairwise preferences, we will be using a different feature removal technique that uses a "baseline" item to create hybrid items instead of the feature distributions. Baseline feature removal techniques select one item as the baseline item and then compare all other items to it. The benefit of these methods is that the exact feature contributions can be computed without any sampling. The disadvantage is that often it is hard to select the baseline sample because different baseline samples create different feature attributions and, in most contexts, it is hard to identify a "neutral" or "average" item. As an example, in related work, we mentioned [33] that attempts to identify a good baseline input document for DeepSHAP in IR. As another example, the baseline implementation of SHAP [21] uses the all-zeroes item as the baseline sample. While selecting a baseline sample is not simple in most cases, we find that the baseline feature removal technique is a natural fit when we are explaining the difference in outcomes between two items $\mathbf{v}$ and $\mathbf{u}$.

When explaining the pairwise outcome of two items $\mathbf{v}$ and $\mathbf{u}$, we are going to generate an explanation for one item using the other as the baseline. In other words, for coalition $\mathcal{S}$, we will be creating the hybrid sample $\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{u}_{\mathcal{S}}$. Note that we do not need the feature distributions or any other parameters for this method. Additionally, note that we are not selecting a fixed item as the baseline but we dynamically change it depending on the pair of items we want to compare. This definition has a natural interpretation, the feature importance of a pairwise explanation amounts to the difference between the outcome of the two items. According to the property of efficiency (see Section 3) we have: $\sum_{i\in\mathcal{A}} \varphi_i(f,\mathbf{v}) = f(\mathbf{v}) - \mathbb{E}_{\mathbf{X}}[f(\mathbf{X})] = f(\mathbf{v}) - \frac{1}{2}(f(\mathbf{v}) + f(\mathbf{u})) = \frac{1}{2}(f(\mathbf{v}) - f(\mathbf{u}))$.

The Shapley value of $\mathbf{v}$ in comparison to $\mathbf{u}$ is defined as:

$$\phi_i(f, \mathbf{v} \succ \mathbf{u}) = \sum_{\mathcal{S}} \frac{|\mathcal{S}|!(n - |\mathcal{S}| - 1)!}{n!} \iota_{f,\mathbf{v},\mathbf{u}}(i, \mathcal{S}) \tag{7}$$

Note that Eq. 7 differs from Eq. 6 in setting $\mathcal{D} = \{\mathbf{u}\}$. Note also that any QoI from Section 4 can be used when calculating the pairwise explanation. Because pairwise preferences are of especial interest to ranking tasks, we will only be using rank as the QoI for the pairwise method in the rest of the paper.

*Case Study: Explanations of Pairwise Outcomes in CS Rankings.* In Figure 4b- 4d we continue our analysis of the top-$k$ and consider the relative ranking of three universities: Georgia Tech in rank 5, Stanford in rank 6, and UMich in rank 7. We wish to understand why Georgia Tech is ranked higher than Stanford (Figure 4c), and why Stanford is ranked higher than UMich (Figure 4d). In both cases, Georgia Tech and UMich have lower values for all features except Systems. The Systems value of Georgia Tech is high enough to overcome the contributions of other features and rank it higher than Stanford. However, for UMich, we see that, while Systems is the most important feature in the top-10% stratum, it is not important enough to move UMich above Stanford.

Pairwise Shapley explanations can clarify rank differences between two items. In Fig. 4c, we explain the pairwise outcome for Georgia Tech vs. Stanford. For $\mathbf{v}_{\text{Georgia Tech}} = (28.5, 7.8, 6.9, 10.2)$, we use $\mathbf{u}_{\text{Stanford}} = (36.7, 5.4, 13.3, 11.5)$ as the baseline. For coalition $\mathcal{S} = \{\text{AI}, \text{Systems}\}$, we construct $\mathbf{v}_{\mathcal{A}\setminus\mathcal{S}}\mathbf{u}_{\mathcal{S}} = (36.7, 5.4, 6.9, 10.2)$, enabling a direct feature comparison. The pairwise explanation from ShaRP is intuitive: in the same figure, Systems improves Georgia Tech's rank by 5 compared to Stanford. Feature contributions sum to half the rank difference between these universities, aligning with Fig. 4a, which highlights Systems as particularly influential for top-$k$ universities.

## 6 EMPIRICAL EVALUATION

Multiple metrics for evaluating explanation methods across key dimensions have been proposed [22, 24], including for ranking [4, 8, 33, 37]. In this work, we use such metrics to compare explanation methods and adapt or define several others for evaluating feature importance in ranking. We aim to formulate these metrics as generally as possible to support broader applicability.

Our focus is on explanation methods that return a numerical vector of feature attributions explaining the outcome for a given item. We can assess *pair-wise explanation agreement* by comparing the feature vectors of a pair of explanations. Furthermore, we can use an explanation to compute the outcome for the item being explained (e.g., its rank), and compare it to the actual observed outcome for that item. This allows us to assess *fidelity of an explanation.* Below, we describe explanation agreement and fidelity metrics and also explain how these primitives can be aggregated to assess *sensitivity* and *fidelity* of an explanation method, and to quantify inter-method *explanation agreement.*

*Notation.* In Section 3, we have been using $\phi(\mathbf{v})$ to represent the vector of feature weights, computed using Shapley values. We generalize our definition here to $g(\mathbf{v})$ to represent the output of any feature-based explanation method $g$, regardless of whether it consists of Shapley values or of some other numerical quantification of feature importance. For all methods we consider, $g(\mathbf{v})$ is a vector of numerical contributions of each feature towards the outcome for item $\mathbf{v}$.

### 6.1 Fidelity Metrics

*Explanation Fidelity.* A useful property of feature-based explanations is that the actual outcome can be computed from them. For Shapley-value-based explanations, this follows from the efficiency property of Shapley values, see Section 3. Fidelity measures how well the explanation $g(\mathbf{v})$ matches the model prediction $f(\mathbf{v})$ being explained, see [7, 24]. SHAP and LIME explanations can be used to compute an item's score (score QoI in our terminology) [7, 21], with feature importance indicating the displacement due to that feature from the mean score, either positively or negatively. ShaRP explanations can be used to compute the outcome for all supported QoIs, including score, rank, and top-$k$, and for the pairwise method.

For QoIs that concern a single item, namely, score, rank, and top-$k$, we compute fidelity of explanation $g$ for item $\mathbf{v}$ as:

$$F(g, \mathbf{v}, \text{QoI}()) = 1 - \frac{1}{Z}\left|\text{QoI}(\mathbf{v}) - \sum_{i=1}^{d} g(i, \mathbf{v})\right| \tag{8}$$

Here, $\text{QoI}(\mathbf{v})$ returns the value of the quantity of interest (i.e., the outcome being explained by $g$), such as $\mathbf{v}$'s score, rank, or presence in the top-$k$, while $g(i, \mathbf{v})$ is the contribution of the i-th feature of $v$. Finally, $Z$ is the normalizer set to the maximum distance between a pair of outcomes for the given dataset $\mathcal{D}$ and ranker $f$ (omitted to simplify notation), and for the specified QoI. Note that, for pairwise explanations, fidelity $F(g, \mathbf{u} \succ \mathbf{v}) = 1$ if $u$ is ranked higher than $u$ and if $g$ predicts that relative order among the items, and is 0 otherwise.

Example 3. *Consider, for example, the explanation of Texas A&M University's rank* $= 34$ *in CS Rankings, presented as a waterfall plot in Figure 2b. The sum of feature weights* $-20.78 - 19.52 - 18.71 - 1.95 = -60.96$ *captures the displacement of Texas A&M University in the ranking relative to the middle of the ranked list (position* 94.5 *out of* 189*), up to rounding:* $94.5 - 60.96 = 33.54$. *This explanation has near-perfect fidelity* $1 - \frac{0.16}{189} = 0.998$. *We use the length of the ranked list* $Z = 189$ *as the normalizer for rank QoI.*

*Method Fidelity.* We aggregate per-item fidelity (per Equation 8) to quantify the fidelity of an explanation method as:

$$F(g, \mathcal{D}) = \mathbb{E}_{\mathbf{v}\in\mathcal{D}} F(g, \mathbf{v}) \tag{9}$$

For pairwise, we compute $F(g, \mathcal{D})$ as the expectation of $F(g, \mathbf{u} \succ \mathbf{v})$ over all pairs of distinct items $\mathbf{u}, \mathbf{v} \in \mathcal{D}$.

## 6.2 Agreement Metrics

*Explanation Agreement.* When comparing explanation methods, we may be interested in knowing how similar their explanations are for *the same item.* Alternatively, when analyzing an explanation method, we may want to know how similar its explanations are for *some pair of items* (e.g., those that are similar in feature space, or that have similar outcomes, or both).

We define explanation agreement, based on three distance metrics often used for comparing rankings [8, 33], (1) Kendall's tau distance, (2) Jaccard distance of the top-2 features, and (3) Euclidean distance between the explanation vectors. For each of these distance metrics, we normalize them to the $[0, 1]$ range and then transform their output so that 1 means full agreement (similarity) and 0 means full disagreement. For dataset $\mathcal{D}$ and ranker $f$, we define explanation agreement as:

$$A(g, q, \mathbf{u}, \mathbf{v}, \mathsf{sim}()) = \mathsf{sim}(g(\mathbf{u}), q(\mathbf{v})) \quad (10)$$

Here, $g$ and $q$ are explanation methods, $\mathbf{u}$ and $\mathbf{v}$ are points being explained, and sim() is a function that computes the specified similarity metric over the explanations. Two important cases are: when $g = q$ and $\mathbf{u} \neq \mathbf{v}$, we are comparing explanations generated by the same method for different points. Conversely, when $g \neq q$ and $\mathbf{u} = \mathbf{v}$, we are comparing explanations of the same point generated by different methods.

Example 4. *For example, consider the explanations of Texas A&M's score and rank, produced by ShaRP for score QoI 2a and rank QoI 2b, respectively. These explanations are similar in the sense that they explain two related outcomes (score and rank) of the same item. However, they are dissimilar in that the relative importance of Texas A&M's features is different. For rank QoI, the explanation ranks features as ⟨Systems, AI, Inter, Theory⟩. However, for score QoI, the explanation ranks features differently as ⟨Inter, AI, Systems, Theory⟩. These lists are dissimilar in terms of the relative order of the features, with 3 out of 6 possible pairs appearing in the opposite relative order. An explanation agreement metric that uses Kendall's tau distance as a sub-routine allows us to quantify this.*

*Method Agreement.* To compute agreement for a pair of explanation methods $g$ and $q$, for a dataset $\mathcal{D}$, we compute explanations for each item using each method, compute pair-wise explanation agreement per Eq. 10, and aggregate it across $\mathcal{D}$.

$$A(g, q, \mathcal{D}, \mathsf{sim}()) = \mathbb{E}_{\mathbf{v} \in \mathcal{D}} A(g, q, \mathbf{v}, \mathbf{v}, \mathsf{sim}()) \quad (11)$$

*Method Sensitivity.* The Sensitivity of an explanation method quantifies the similarity between explanations of similar items [4]. We will use $\mathsf{nbr}(\mathbf{v})$ (as in "neighbor") to refer to a function that retrieves items that are in some sense similar to $\mathbf{v}$, noting that this similarity may be based on items' features, their outcomes for some QoI, or both. For each $\mathbf{v}$, we retrieve its neighbors $\mathsf{nbr}(\mathbf{v})$, compute pair-wise explanation agreement between $\mathbf{v}$ and each of its neighbors per Eq. 10, and aggregate this value over $\mathcal{D}$:

$$S(g, \mathcal{D}, \mathsf{sim}, \mathsf{nbr}()) = \mathbb{E}_{\mathbf{v} \in \mathcal{D}, \mathbf{u} \in \mathsf{nbr}(\mathbf{v})} A(g, g, \mathbf{v}, \mathbf{u}, \mathsf{sim}()) \quad (12)$$

## 7 THE SHARP LIBRARY

ShaRP is implemented in Python, follows an API structure similar to scikit-learn [5], and is parallelized. The library can be used both to compute exact feature importance values and to approximate them to improve running times.

*Implementation of QoIs for ranking.* We provide Algorithm 1 to showcase the flexibility of ShaRP . Using this implementation, we can 1) easily switch between QoIs, 2) calculate both marginal and baseline Shapley values, and 3) approximate Shapley values for efficiency. The algorithm relies on black-box access to the model that generates the outcome (i.e., specifying an input and observing the outcome used in the QoI). Specifically, Algorithm 1 takes as input a dataset $\mathcal{D}$, a reference set $\mathcal{D}' \subseteq \mathcal{D}$ from which samples are drawn, an item $\mathbf{v}$ for which the explanation is generated, the number of samples $m$, the maximum coalition size $c$, and the $\iota()$ function (Equation 5) used to quantify feature importance.

To change the QoI, we modify the input $\iota()$ function. To switch to the pairwise baseline method, we set $\mathcal{D}' = \mathbf{u}$ and $m = 1$, where $\mathbf{u}$ is the baseline item to compare against $\mathbf{v}$. To approximate feature importance, we control the parameters $m$ and $c$. Passing in the full set of items as the reference set ($\mathcal{D}' = \mathcal{D}$), and setting $m = |\mathcal{D}| - 1$ and $c = |\mathcal{A}| - 1$, yields exact Shapley value computation—i.e., each feature of $\mathbf{v}$ is quantified against all other items in $\mathcal{D}$ using all possible coalitions of features except the one being evaluated.

Because we compute the rank of each item relative to the entire dataset $\mathcal{D}$, the dataset must be provided along with the reference set. We provide an empirical analysis of the impact of $m$ and $c$ on performance in Section 8.3.2.

We now describe the algorithm for marginal exact computation, which generalizes all cases discussed above. By definition, Shapley values compute feature importance using all possible coalitions of features and all items in the dataset—referred to here as the *exact computation* of local feature-based explanations. For illustrative purposes, we explicitly include the construction of the random variable $\mathbf{U}$ in lines 4–7 of Algorithm 1. For each feature $i \in \mathcal{A}$, the algorithm considers all coalitions $\mathcal{S} \subseteq \mathcal{A} \setminus \{i\}$. For each $\mathcal{S}$, it draws $m = |\mathcal{D}| - 1$ samples from $\mathcal{D}$. Two vectors of items are then constructed: $\mathbf{U}_1$, where features in $\mathcal{S}$ vary as in $\mathbf{U}$ and the rest are fixed to their values in $\mathbf{v}$; and $\mathbf{U}_2$, where features in $\mathcal{S} \cup \{i\}$ vary as in $\mathbf{U}$, with the remaining features again fixed to $\mathbf{v}$. The importance of coalition $\mathcal{S}$ for feature $i$, denoted $\phi_{i_\mathcal{S}}(\mathbf{v})$, is computed using the QoI function $\iota()$, which measures the difference between $\mathbf{U}_1$ and $\mathbf{U}_2$. This quantity is then weighted by the number of coalitions of size $|\mathcal{S}|$—specifically, $\binom{d-1}{|\mathcal{S}|}$—and accumulated into the final contribution $\phi_i(\mathbf{v})$, normalized over all possible coalition sizes $d$.

In practice, one of the main bottlenecks in computing feature contributions, especially with complex black-box models, is inference time. To mitigate this, we cache inference results in a hash map, allowing repeated inputs to return cached outputs in constant time ($O(1)$). This significantly speeds up computation as more tuples are processed. Initially, the explainer experiences a "cold start" with no cached results, but performance improves to a "warm start" as the cache builds, reducing the need for repeated model inference.

Evaluating the $\iota()$ function, is straightforward for the *score QoI* but not for the ranking-specific QoIs. Specifically, for the score QoI, using the definition in Section 4, we take the mean of the

**Algorithm 1** Local feature importance using ShaRP

```
Require: Dataset $\mathcal{D}'$, reference set $\mathcal{D}'$, item $\mathbf{v}$, number of samples $m$, maximum coalition size c, $\iota()$
1: $\phi(\mathbf{v}) = \langle 0, \ldots, 0 \rangle$
2: for $i \in \mathcal{A}$ do
3:     for $\mathcal{S} \subseteq \mathcal{A} \setminus \{i\}$ and $|\mathcal{S}| \leq c$ do
4:         $\mathbf{U} \sim \mathcal{D}' \setminus \mathbf{v}, m$
5:         $\mathbf{U}_1 = \mathbf{v}_{\mathcal{A} \setminus \mathcal{S}} \mathbf{U}_{\mathcal{S}}$
6:         $\mathbf{U}_2 = \mathbf{v}_{\mathcal{A} \setminus \{\mathcal{S} \cup i\}} \mathbf{U}_{\mathcal{S} \cup i}$
7:         $\phi_{i_{\mathcal{S}}}(\mathbf{v}) = \iota(\mathbf{U}_1, \mathbf{U}_2)$
8:         $\phi_i(\mathbf{v}) = \phi_i(\mathbf{v}) + \frac{1}{d} \frac{1}{\binom{d-1}{|S|}} \phi_{i_{\mathcal{S}}}(\mathbf{v})$
9:     end for
10: end for
11: return $\phi(\mathbf{v})$, the Shapley values $\mathbf{v}$'s features
```

**Algorithm 2** $\iota_{Rank}$

```
Require: Dataset $\mathcal{D}'$, scoring function $f$, item $\mathbf{v}$, $\mathbf{U}_1$, $\mathbf{U}_2$, number of samples $m$
1: $\phi = 0$
2: for $i \in \{1, \ldots, m\}$ do
3:     $\mathbf{u}_1 = \mathbf{U}_1(i)$
4:     $\mathbf{u}_2 = \mathbf{U}_2(i)$
5:     $\mathcal{D}_1 = \mathcal{D} \setminus \{\mathbf{v}\} \cup \{\mathbf{u}_1\}$
6:     $\mathcal{D}_2 = \mathcal{D} \setminus \{\mathbf{v}\} \cup \{\mathbf{u}_2\}$
7:     $\phi = \phi + r^{-1}_{\mathcal{D}_2,f}(\mathbf{u}_2) - r^{-1}_{\mathcal{D}_1,f}(\mathbf{u}_1)$
8: end for
9: return $\phi / |\mathbf{U}_1|$
```

(per-element) difference of $f(\mathbf{U}_1)$ and $f(\mathbf{U}_2)$. However, this is not the case for ranking-specific QoIs. The rank of an item is computed with respect to all other items in the sample. This adds two steps to calculating the rank QoI compared to the score QoI. The item we are explaining needs to be removed from $\mathcal{D}'$, and the score of each item $\mathbf{u}_i \in \mathbf{U}_1$ (and equivalently $\mathbf{u}_j \in \mathbf{U}_2$) needs to be compared to the scores of all items in $\mathcal{D}'$. The computation of $\iota_{Rank}$ is summarized in Algorithm 2.

To compute feature importance that explains whether an item appears at the top-$k$, for some given $k$, we use a similar method as for rank QoI. The difference is that, rather than computing the difference in rank positions for a given pair of items $\mathbf{u}_1$ and $\mathbf{u}_2$, we instead check whether one, both, or neither of them is at the top-$k$. As in Algorithm 2, we work with $\mathcal{D}_1 = \mathcal{D} \setminus \{\mathbf{v}\} \cup \{\mathbf{u}_1\}$ and $\mathcal{D}_2 = \mathcal{D} \setminus \{\mathbf{v}\} \cup \{\mathbf{u}_2\}$ for each sample. We increase the contribution to $\phi$ by 1 if only $\mathbf{u}_1$ is in the top-$k$, and decrease it by 1 if only $\mathbf{u}_2$ is in the top-$k$. We omit pseudocode due to space constraints.

*Visualizing feature importance.* We use three visualization methods. First, waterfall plots (Figure 2) show feature importance for a single item, following [21]. Second, box-and-whisker plots (Figures 3, 4a, 5) aggregate local importance across 10%-width ranking strata, showing median and variance per feature. Third, bar charts (Figures 4c, 4d) display pairwise contributions from the perspective of the first item in each pair.

## 8 EXPERIMENTAL EVALUATION OF SHARP

We ran extensive experiments on real and synthetic datasets with score-based ranking tasks to demonstrate the utility and performance of ShaRP . Section 8.3 presents efficiency results, Section 8.1 provides a qualitative evaluation, and Section 8.2 compares ShaRP to other methods using the metrics from Section 6. All experiments were run on a 14-core Intel Xeon Platinum 8268 (2.90GHz) machine with 128GB RAM. We evaluate the performance of ShaRP and compare it to other local feature importance methods, using several real and synthetic datasets, with the corresponding ranking tasks. Dataset properties, along with ranker type (score-based or learned) are summarized in Table 1, see Appendix A for details.

### 8.1 Qualitative Analysis

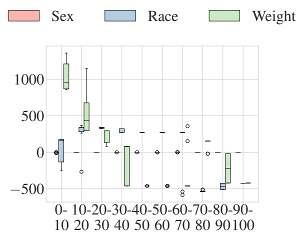
We already presented a detailed case study of CS Rankings presented as an example across the previous sections. To evaluate ShaRP across different settings, we conducted two additional experiments. First, we analyzed a set of simple synthetic datasets coming from multiple different distributions and studied how each distribution affects the ranking. Secondly, we compared the explanations resulting from two different LtR rankers for the Moving Company dataset.

*8.1.1 Score-based Ranking with Synthetic Data.* In this set of experiments (see Appendix B for details), we use simple two-feature datasets to study how feature distributions and scoring functions interact with ranking. We consider two settings: (1) fixed scoring function with varying distributions, and (2) fixed distributions with varying scoring functions.

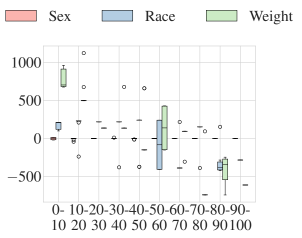
When the scoring function is fixed, *feature importance depends on both distribution and stratum.* Features with higher variance dominate at the top, while in the middle either feature may prevail, increasing variability. For negatively correlated features, the pattern holds with opposite contribution signs. Discrete features (e.g., Bernoulli) split the ranking into segments, with the second feature determining order within each. When distributions are fixed and scoring functions vary, *importance varies by stratum*, depending on both weight and variance. A low-variance feature can dominate if its weight is high. Finally, we show that under certain distributions, *low-ranked items can jump to the top-k*, contradicting the locality assumption in Anahideh and Mohabbati-Kalejahi [2]. Even items in the top-50% can move into the top-10% with specific value changes.

**Table 1: Datasets, sorted by # tuples. S stands for score-based ranked task and LtR for learning-to-rank.**

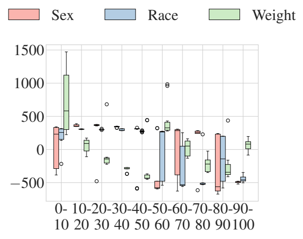
| name | source | # tuples | # features | task |
|---|---|---|---|---|
| Tennis (ATP) | [19] | 86 | 6 | S |
| CS Rankings (CSR) | [3] | 189 | 5 | S |
| Times Higher Education (THE) | [18] | 1,397 | 5 | S |
| Synthetic (SYN) | here | 2,000 | 2 or 3 | S |
| ACS Income - Alaska (ACS-AK) | [12] | 3,546 | 10 | LtR |
| Moving company (MOV) | [38] | 4,000 | 3 | LtR |
| ACS Income - Texas (ACS-TX) | [12] | 135,924 | 10 | LtR |



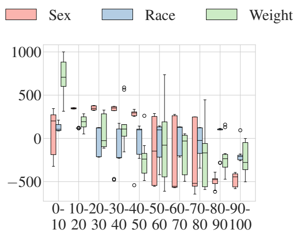
**(a) XGB (original data)**



**(b) XGB (fairness interv.)**



**(c) LGB (original data)**



**(d) LGB (fairness interv.)**

**Figure 5: Feature contribution to the rank QoI for (a) XGB over the original moving company dataset, (b) XGB over the unbiased version, (c) LGB over the original moving company dataset and (d) LGB over the unbiased version.**

*8.1.2 Learning to Rank.* We now showcase how ShaRP can be used to audit black-box rankers and understanding their decision process. We use an XGB ranker with a pairwise ranking objective and an LGB ranker with a LambdaRank objective. Both are trained on training sets and evaluated on test sets of 2,000 tuples each. We use ShaRP to explain 100 items (10 per stratum) of each test set, with no approximations and the rank QoI. In Figure 5, we observe that the two LtR models behave significantly differently.

XGB rankers do not appear to rely on the Sex feature, regardless of whether the de-biasing intervention from [38] is applied. However, Race remains influential; in Figure 5a, it boosts applicants' rankings by roughly 400 positions up to the 70th percentile. This is notable given that Weight Lifting contributes positively in the 70–80th percentile range but negatively in the 60–70th range. Ideally, its impact should be more monotonic, as partially achieved in Figure 5b. Although Race shows slightly reduced influence after the intervention, it remains an important feature.

In contrast, LGB rankers tend to rely on all features. In the original model (Figure 5c), Sex and Race are highly influential across all strata, often ranking as the top features for applicants in the lower percentiles (50th and below). Analysis of the 10–20th, 60–70th, and 90–100th percentiles shows that Weight Lifting has minimal impact on decisions, with Sex and Race largely determining rank. The fairness intervention reduces this effect somewhat (Figure 5d) by increasing the influence of Weight Lifting, but Race and Sex remain dominant features, occasionally outweighing Weight Lifting.

In summary, results indicate that XGB relies more on Race, while LGB emphasizes Sex. Bias mitigation is effective up to the 10th percentile but fails to correct bias across the remaining strata.

*8.1.3 ACS Income.* We use the 2018 ACS Income dataset (10 features, 6 categorical) from Alaska (3,546 records) and Texas (135,924 records) as a secondary case study. The task is to predict whether an individual's income exceeds \$50,000, using a pipeline with one-hot encoding and a Random Forest Classifier (RFC). Unlike other methods, ShaRP can generate explanations at any pipeline stage, including over raw features. Individuals are ranked by classification score, with explanations shown in Figures 6 and 19 (Appendix F).

Figure 6b shows overall feature importance in Alaska. Hours worked (WKHP), marital status (MAR), age (AGEP), and race (RAC1P) are most influential, followed by education (SCHL), which only matters in the top 20%. Marital status impacts rank across all strata, while race, marital status, and sex dominate in the top 60%, 50%, and 10% respectively. The top 10% are mostly white, married, and male; in contrast, education and hours worked vary more but are less important. Feature importance shifts notably in Texas. Education becomes key—especially in the top 10% and bottom 30%. Age plays a smaller role, marital status remains influential at both extremes, race has limited impact, and sex is relevant but rarely dominant.

This experiment shows the effectiveness of ShaRP on higher-dimensional data and highlights nuanced differences in feature importance across data subsets.

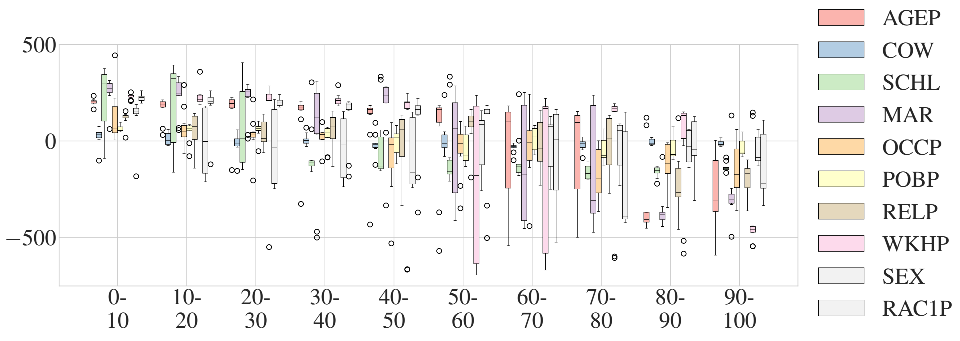
## 8.2 Comparison to Other Methods

In this section, we compare explainability methods using the metrics from Section 6, focusing on a subset from Section 2. Since ShaRP and Shapley values target individual explanations, we exclude global methods such as those by Yang et al. and Gale and Marian [14, 39]. To compare with HIL [40], we adapt their code to support real data and arbitrary score-based rankers (see full version [27]), and focus on their weight-based methods, as their Shapley approximation is already covered by SHAP. We exclude PrefShap [17], which is restricted to pairwise data with a specialized kernel model.

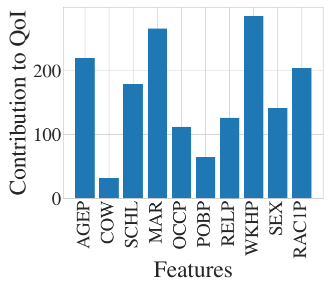
We compare to HRE [2] but use only four of their internal methods as provided by their public code base (Decision Trees (DT), Linear Regression (LR), Ordinary Least Squares (OLS), and Partial Least Squares (PLS)) and their default neighborhood settings (5-10 consecutive positions above and below the item being explained). We compare to DEXER which fits a linear regression model to the ranks and explains this model using the score-based SHAP instead of the original blackbox, treating rank as a score. Finally, we compare to SHAP [21] and LIME [29], due to their wide use and availability, even though they are not designed for ranking.

*8.2.1 Sensitivity.* Figure 7 compares the sensitivity of all methods by evaluating explanation similarity for pairs of similar items. For each pair, we compute: (1) Euclidean distance between explanations (x-axis), (2) rank difference (y-axis), and (3) feature distance (hue; lighter means more similar). Each plot centers the reference item at (0,0), with scatter points showing neighbors' distances. Results are overlaid across all items, each used in turn as the reference point.

Intuitively, items with similar features and close rankings should have similar explanations—points should lie near the diagonal $y = x$, with hue darkening as distance grows. In practice, this often fails: a dominant feature may decouple feature and explanation similarity, and dissimilar items can yield similar outcomes. Ideally, explanations should vary for closely ranked items with distinct



(a) ACS strata



(b) Overall

**Figure 6: Feature contribution on ACS Income (Alaska) to the rank QoI (a) across strata and (b) overall.**

features and differ significantly for distant ranks, filling the space below $y = x$ with hue darkening outward.

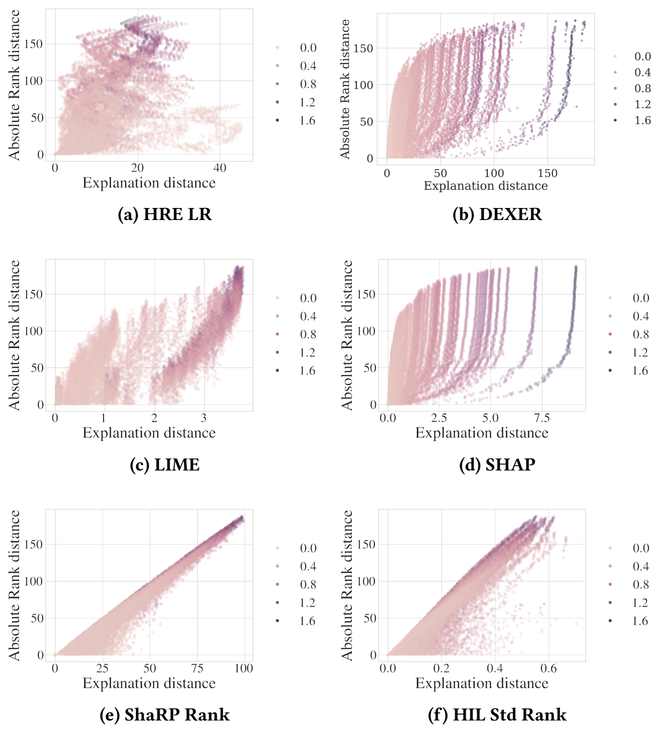


**Figure 7: Sensitivity results for CS Rankings. Each dot represents a neighbor of the reference item; the x-axis shows Euclidean explanation distance, the y-axis rank difference, and hue indicates feature similarity. Methods using rank as the profit function (ShaRP and HIL Std rank) perform best, with ShaRP leading. These are the only methods that consistently produce similar explanations for items with similar features and outcomes.**

In Figure 7, only the rank QoI methods produce the expected shape. Both ShaRP (Figure 7e) and HIL-Std-Rank (Figure 7f in our implementation) generate similar explanations for similarly ranked, feature-similar items, with ShaRP forming slightly denser clusters. In contrast, SHAP (Figure 7d), a score-based method, reflects primarily feature distance: its plot shows darkening bands away from the origin, but assigns nearly identical explanations to items with similar features even when their ranks differ substantially.

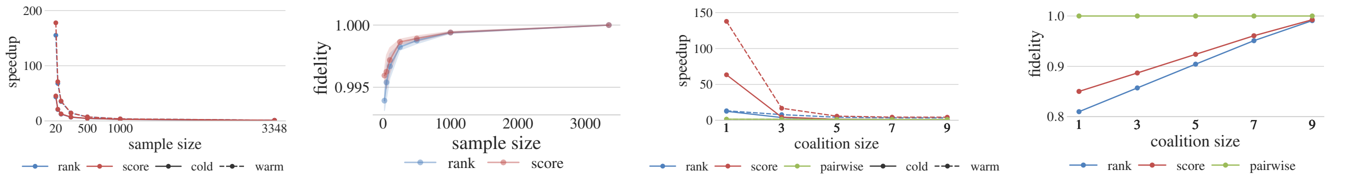
LIME (Figure 7c), another score-based method, reflects both feature and rank distance but fails to distinguish explanations as clearly as rank-based methods. Score-based methods generally struggle to capture the nonlinear relationship between score and rank. DEXER (Figure 7b), which uses linear regression to predict rank and SHAP for explanations, performs similarly to other score-based approaches. While non-linear models might better approximate rank, our approach directly integrates rank into the Shapley value utility. HRE (Figure 7a) shows no clear pattern with respect to rank or feature distance; similar and dissimilar explanations appear across all ranks and hues. This is expected, as HRE depends on local neighbors, which can vary widely in features and outcomes.

In the full version [27], we provide additional sensitivity results, comparing ShaRP with rank QoI to HIL Std rank and analyzing a score-based task. We show that ShaRP outperforms HIL across datasets and that ShaRP with score QoI aligns well with the diagonal in score-based tasks—underscoring the importance of choosing a QoI aligned with the explanation goal.

In summary, explanations for the rank QoI, which we are introducing in this paper, are able to more accurately explain ranking tasks compared to other local feature-based explanation methods.

We also quantified agreement between explanations produced by different methods. We show these results in Appendix E.2.

*8.2.2 Fidelity.* It is possible to calculate Fidelity for SHAP, LIME, ShaRP, and the HIL-score. It is impossible to compute Fidelity for HIL-rank and all the HRE methods. All methods except HIL-score perform very well. We compute the Fidelity averaged across all items in all datasets. All methods are executed using their recommended settings to compute explanations for score QoI. Additionally, we compute fidelity for ShaRP for the rank QoI. Recall that ShaRP is the only method that can compute an explanation for



**(a) Speedup vs. sample size, ACS** **(b) Fidelity vs. sample size, ACS** **(c) Speedup vs. max coalition size, ACS** **(d) Fidelity vs. max coalition size, ACS**

**Figure 8: Running time of approximation for ACS Income (AK). In (a) and (b), max coalition size is 9; in (c) and (d), sample size is 100. Speedup is computed vs. to exact times in Table 2, see Table 5 in the Appendix for additional information. Due to a slight difference in the tie breaking method, the dataset's size (and maximum sample size) was set to 3,348.**

this QoI. LIME, SHAP and ShaRP are all achieving high explanation fidelity, on average ranging from 0.94-0.98, 0.97-1.00 and 1.00 correspondingly. HIL has reasonable fidelity for CSR (0.85) but does not perform consistently on other datasets ranging from 0.14-0.64. See Table 4 in the full version of the paper [27] for details.

## 8.3 Efficiency and Approximation

*8.3.1 Running time of exact computation.* In our first experiment, we measure the exact computation time for the rank and score QoIs, and the pairwise method with rank QoI, on three real and one synthetic dataset from Table 1. We include only one synthetic dataset, as all have the same size ($m$ = 2,000) and at most three features; differences in correlation structure do not affect runtime. We omit the top-$k$ QoI, as its implementation mirrors the rank QoI, resulting in indistinguishable runtimes.

Table 2 presents the results, reporting the time to generate an explanation per point, averaged over 100 points for CSR, THE, SYN, and ACS-AK, and over 83 points (dataset size) for ATP. Runtime for rank and score QoIs increases with both the number of items ($m$ in Algorithm 1) and features ($d$), as exact computation scales linearly with $m$ and exponentially with $d$ ($2^d - 1$ coalitions). Pairwise methods involve only two items, so their runtime is independent of $m$ but remains exponential in $d$. Our pairwise method for rank QoI also requires recomputing ranks after each intervention (line 7, Algorithm 2), which scales linearly with $m$ in our implementation. This explains why pairwise QoI for THE ($m$ = 1,397, $d$ = 5) runs slower than for ATP ($m$ = 86, $d$ = 6). Exact computation is particularly challenging for ACS-AK due to its higher feature count. We next demonstrate how approximations can mitigate this cost.

**Table 2: Running time of exact computation, cold start.**

| | | | avg. time (sec) | | |
|---|---|---|---|---|---|
| dataset | # tuples | # features | score | rank | pair |
| ATP | 86 | 6 | 0.004 | 0.026 | 0.004 |
| CSR | 189 | 5 | 0.002 | 0.022 | 0.003 |
| THE | 1,397 | 5 | 0.011 | 0.423 | 0.007 |
| SYN | 2,000 | 3 | 0.002 | 0.126 | 0.003 |
| ACS-AK | 3,546 | 10 | 1,960.7 | 1,956.8 | 2.53 |

*8.3.2 Running time and quality of approximation.* To reduce runtime, we implement two approximation methods: limiting the number of samples and bounding coalition size. We report running time and fidelity (Eq. 8) to assess approximation quality. Figure 8 shows results for ACS-AK, see Appendix G for ATP and CS Rankings.

Figure 8a shows the speed-up achieved by reducing the number of samples $m$. Lowering $m$ from 1,348 (exact) to 20, while maintaining a maximum of size 9 coalitions (the largest possible for 10 features), accelerates rank QoI by a factor of 79, reducing runtime from 1956 sec to 45 sec. Crucially, this performance gain does not compromise fidelity, which remains above 0.99 (out of 1) across all sample sizes in all experiments(detailed in the full paper [27]). Figures 8c and 8d show speed-up and fidelity when bounding coalition size. The largest speed-up occurs for coalition size 1, though fidelity is lower: at least 0.81 for rank and 0.85 for score (fidelity is 1 for pairwise). Fidelity improves with coalition size 3, reaching 0.86 for rank and 0.89 for score.

Table 3 shows per-tuple explanation times across different maximum coalition and sample sizes, highlighting the trade-off between runtime and fidelity. For large datasets, approximate methods yield substantial speedups with minimal fidelity loss. In ACS (AK), for example, a ranking can be explained in 9.45 seconds (vs. 1,956 seconds for exact computation). Warm start is typically 3 times faster than cold start, and pairwise explanations are the fastest overall. Figures 8b and 8a illustrate how fidelity and runtime vary with sample size. As shown in Table 5, runtime grows linearly with sample size, while fidelity decreases gradually, reflecting a favorable accuracy–efficiency trade-off.

In summary, reducing the number of samples and bounding coalition size improves runtime while maintaining high explanation fidelity. Computing Shapley values is exponential in the number of features, and it is common to develop model-specific approximations for explainers like SHAP [21]. Designing more sophisticated custom optimizations for our QIIs is in our immediate plans.

## 9 USER STUDY

We conducted an IRB-approved study (NYU IRB-FY2025-9983) to explore how users interpret rank-based vs. score-based explanations, using CS Rankings. We summarize the study protocol and the results, see Appendix H and I in the full paper [27] for details.

*Participant recruitment and study protocol.* Through our institution, we recruited 13 participants: 6 PhD students, 3 postdocs, 2

**Table 3: Running time of optimized computation. Running times are reported per data point, in seconds. Parameter optimization was performed separately for each dataset. The optimal RFC for ACS (AK) used 100 estimators, compared to 10 for ACS (TX), resulting in faster cold-start inference per tuple for ACS (TX).**

| | | | | | | avg. time (sec) | | | fidelity | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| dataset | # tuples | # features | start | max coal. size | sample size | score | rank | pair | score | rank | pair |
| `ACS (AK)` | 3,348 | 10 | cold | 9 | 100 | 143.54 | 151.28 | 1.97 | 0.997 | 0.997 | 1.0 |
| `ACS (AK)` | 3,348 | 10 | warm | 9 | 100 | 40.42 | 41.56 | 1.64 | 0.997 | 0.997 | 1.0 |
| `ACS (AK)` | 3,348 | 10 | warm | 9 | 20 | 8.09 | 9.45 | 1.64 | 0.996 | 0.994 | 1.0 |
| `ACS (AK)` | 3,348 | 10 | warm | 7 | 20 | 7.95 | 9.28 | 1.64 | 0.960 | 0.951 | 1.0 |
| `ACS (AK)` | 3,348 | 10 | warm | 5 | 20 | 6.07 | 7.37 | 1.56 | 0.923 | 0.904 | 0.9 |
| `ACS (AK)` | 3,348 | 10 | warm | 3 | 20 | 2.08 | 3.39 | 1.35 | 0.886 | 0.856 | 0.9 |
| `ACS (AK)` | 3,348 | 10 | warm | 2 | 20 | 0.74 | 2.05 | 1.27 | 0.868 | 0.833 | 0.9 |
| `ACS (TX)` | 135,924 | 10 | cold | 9 | 100 | 126.39 | 139.69 | 7.69 | 0.998 | 0.997 | 1.0 |
| `ACS (TX)` | 135,924 | 10 | warm | 9 | 100 | 40.42 | 48.79 | 7.65 | 0.998 | 0.997 | 1.0 |
| `ACS (TX)` | 135,924 | 10 | warm | 9 | 20 | 8.07 | 16.28 | 7.59 | 0.992 | 0.989 | 1.0 |
| `ACS (TX)` | 135,924 | 10 | warm | 7 | 20 | 7.95 | 16.35 | 7.69 | 0.973 | 0.959 | 0.9 |
| `ACS (TX)` | 135,924 | 10 | warm | 5 | 20 | 6.27 | 14.33 | 7.50 | 0.944 | 0.913 | 0.9 |
| `ACS (TX)` | 135,924 | 10 | warm | 3 | 20 | 2.52 | 12.46 | 7.84 | 0.911 | 0.864 | 0.8 |
| `ACS (TX)` | 135,924 | 10 | warm | 2 | 20 | 0.93 | 10.98 | 7.13 | 0.894 | 0.839 | 0.8 |

professors, and 2 research staff. All completed forms detailing their academic backgrounds and familiarity with explainability and the dataset. Students and postdocs, all from CS, reported moderate to high familiarity with explainability. Professors and staff, with social science backgrounds applied to AI, showed varied familiarity with explainability. CS Rankings familiarity ranged from high to low, independent of seniority.

Participants were divided into Rank-Group (7 people) and Score-Group (6 people). Both groups received an introductory document corresponding to their group, completed a range of tasks that included either rank-based or score-based explanations, and then participated in a discussion. Each participants answered 22 questions, divided into 3 categories: understanding the rank of a specific department (3 departments × 4 questions), understanding why one department is ranked higher than another (3 department pairs × 2 questions), and understanding feature importance trends across the ranking (2 sets of 6 departments × 2 questions).

*Results.* Rank-Group outperformed Score-Group in terms of accurately answering questions (73% vs. 67%), and also reported higher confidence (4.15 vs. 3.90 on a 5-point Likert scale), see Table 6 in Appendix H). Notably, Score-Group expressed greater distrust in the ranking and the dataset, echoing findings from [1], for example: *"Maybe my mind started looking for some kind of [...] preconceived biases and wondering? [...] There was one figure [...] towards the end. The difference was almost imperceptible, and I kept thinking, why is one ranked few points higher than the other?"*

Several Score-Group participants noted needing multiple explanations to understand the ranking, as score-based explanations lack rank context. For example: *"At first [for the items at the top of the ranking], the differences were so big that [the answer] was very clear, and then at the end, you know which one is better 1.05 or 1.08 [...]? So it makes you want to go back to the earlier questions and makes you question your initial impression and understanding of [the ranking]."*.

While further study is needed to understand the sources of mistrust and validate findings with more participants, our results provide preliminary evidence that rank-based explanations better support understanding and trust in ranking tasks as compared to score-based explanations. Most importantly, several participants underscored that they found feature-based explanations useful. For example: *"I thought that the experience is successful on raising awareness and provoking critical thinking about using rankings."*

## 10 CONCLUSIONS

We introduced a comprehensive framework for quantifying feature importance in selection and ranking. Given the impact of rankers on individuals, organizations, and populations, understanding their decisions is crucial for *auditing and compliance* (ensuring legal adherence), *recourse* (helping individuals improve outcomes), and *design* (optimizing ranking procedures). Our work addresses the interpretability needs of these tasks.

We demonstrated the effectiveness of ShaRP through a qualitative analysis of an impactful real-world task—the ranking of Computer Science departments. This was complemented by an evaluation on real and synthetic datasets, revealing that our defined profit functions provide valuable and complementary insights beyond simple score-rank relationships. We showed that feature importance varies with data distribution *even when* the scoring function is fixed and exhibits locality. Finally, we compared ShaRP to other local feature-based explanation methods, showing it performs favorably. ShaRP is an open-source Python library, and is the only available library for explaining ranked outcomes in tabular data.

## 11 ACKNOWLEDGMENTS

This research was supported in part by NSF Awards No. 2326193 and 2312930. Ivan Shevchenko and Kateryna Akhynko conducted this work through the RAI for Ukraine program of the NYU Center for Responsible AI and supported in part by the Simons Foundation (SFARI Award #1280457, JS). We thank Tilun Wang for contributions to the earlier code base, and Lucius Bynum and Falaah Arif Khan for helpful discussions.

## REFERENCES

[1] Jonathan Aechtner, Lena Cabrera, Dennis Katwal, Pierre Onghena, Diego Penroz Valenzuela, and Anna Wilbik. 2022. Comparing User Perception of Explanations Developed with XAI Methods. In *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 1–7. https://doi.org/10.1109/FUZZ-IEEE55066.2022.9882743

[2] Hadis Anahideh and Nasrin Mohabbati-Kalejahi. 2022. Local Explanations of Global Rankings: Insights for Competitive Rankings. *IEEE Access* 10 (2022), 30676–30693. https://doi.org/10.1109/ACCESS.2022.3159245

[3] Emery Berger. 2023. CSRankings: Computer Science Rankings. https://csrankings.org/.

[4] Umang Bhatt, Adrian Weller, and José M. F. Moura. 2021. Evaluating and aggregating feature-based model explanations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence* (Yokohama, Yokohama, Japan) *(IJCAI'20)*. Article 417, 7 pages.

[5] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[6] Hugh Chen, Ian C Covert, Scott M Lundberg, and Su-In Lee. 2023. Algorithms to estimate Shapley value feature attributions. *Nature Machine Intelligence* 5, 6 (2023), 590–601.

[7] Tanya Chowdhury, Razieh Rahimi, and James Allan. 2023. Rank-LIME: Local Model-Agnostic Feature Attribution for Learning to Rank. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval* (Taipei, Taiwan) *(ICTIR '23)*. Association for Computing Machinery, New York, NY, USA, 33–37. https://doi.org/10.1145/3578337.3605138

[8] Tanya Chowdhury, Yair Zick, and James Allan. 2024. RankSHAP: Shapley Value Based Feature Attributions for Learning to Rank. arXiv:2405.01848 [cs.IR] https://arxiv.org/abs/2405.01848

[9] Ian Covert, Scott M. Lundberg, and Su-In Lee. 2020. Understanding Global Feature Contributions With Additive Importance Measures. In *NeurIPS*. https://proceedings.neurips.cc/paper/2020/hash/c7bf0b7c1a86d5eb3be2c722cf2cf746-Abstract.html

[10] Ian C. Covert, Scott Lundberg, and Su-In Lee. 2021. Explaining by removing: a unified framework for model explanation. *J. Mach. Learn. Res.* 22, 1, Article 209 (Jan. 2021), 90 pages.

[11] Anupam Datta, Shayak Sen, and Yair Zick. 2016. Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 598–617. https://doi.org/10.1109/SP.2016.42

[12] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2022. Retiring Adult: New Datasets for Fair Machine Learning. http://arxiv.org/abs/2108.04884 arXiv:2108.04884 [cs, stat].

[13] Zeon Trevor Fernando, Jaspreet Singh, and Avishek Anand. 2019. A study on the Interpretability of Neural Retrieval Models using DeepSHAP. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19)*. ACM, 1005–1008. https://doi.org/10.1145/3331184.3331312

[14] Abraham Gale and Amélie Marian. 2020. Explaining Ranking Functions. *Proc. VLDB Endow.* 14, 4 (2020), 640–652. https://doi.org/10.14778/3436905.3436922

[15] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2019. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.* 51, 5 (2019), 93:1–93:42. https://doi.org/10.1145/3236009

[16] Maria Heuss, Maarten de Rijke, and Avishek Anand. 2024. RankingSHAP – Listwise Feature Attribution Explanations for Ranking Models. arXiv:2403.16085 [cs.IR] https://arxiv.org/abs/2403.16085

[17] Robert Hu, Siu Lun Chau, Jaime Ferrando Huertas, and Dino Sejdinovic. 2022. Explaining Preferences with Shapley Values. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2022/hash/b1656d20067ca7c84a33785c4083a75e-Abstract-Conference.html

[18] Ivan Shevchenko Joao Fonseca. 2024. Times Higher Education: World University Rankings. https://zenodo.org/records/11235321.

[19] Kateryna Akhynko Joao Fonseca. 2024. ATPTennis: Male Tennis Players in 2020-2023. https://zenodo.org/records/10245175.

[20] Hang Li. 2014. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00607ED2V01Y201410HLT026

[21] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[22] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence* 267 (2019), 1–38.

[23] George Mohler, Michael Porter, Jeremy Carter, and Gary LaFree. 2020. Learning to rank spatio-temporal event hotspots. *Crime Science* 9, 1 (2020), 1–12.

[24] Christoph Molnar. 2020. *Interpretable machine learning*. Lulu. com.

[25] Yuval Moskovitch, Jinyang Li, and H. V. Jagadish. 2023. Dexer: Detecting and Explaining Biased Representation in Ranking. In *Companion of the 2023 International Conference on Management of Data* (Seattle, WA, USA) *(SIGMOD '23)*. Association for Computing Machinery, New York, NY, USA, 159–162. https://doi.org/10.1145/3555041.3589725

[26] Eliana Pastor, Luca de Alfaro, and Elena Baralis. 2021. Identifying biased subgroups in ranking and classification. *arXiv preprint arXiv:2108.07450* (2021).

[27] Venetia Pliatsika, João Fonseca, Kateryna Akhynko, Ivan Shevchenko, and Julia Stoyanovich. 2025. ShaRP: Explaining Rankings and Preferences with Shapley Values. (2025). arXiv:2401.16744 [cs.AI] https://arxiv.org/abs/2401.16744

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-Agnostic Interpretability of Machine Learning. (2016). https://arxiv.org/pdf/1606.05386.pdf

[29] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1135–1144. https://doi.org/10.1145/2939672.2939778

[30] Cynthia Rudin. 2019. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. arXiv:1811.10154 [stat.ML]

[31] Lloyd S Shapley et al. 1953. A value for n-person games. (1953).

[32] Jaspreet Singh and Avishek Anand. 2018. EXS: Explainable Search Using Local Model Agnostic Interpretability. arXiv:1809.03857 [cs.IR] https://arxiv.org/abs/1809.03857

[33] Jaspreet Singh and Avishek Anand. 2018-06-29. Posthoc Interpretability of Learning to Rank Models Using Secondary Training Data. (2018-06-29). arXiv:1806.11330 [cs] http://arxiv.org/abs/1806.11330

[34] Erik Strumbelj and Igor Kononenko. 2010. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research* 11 (2010), 1–18.

[35] Erik Strumbelj and Igor Kononenko. 2014. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.* 41, 3 (2014), 647–665. https://doi.org/10.1007/s10115-013-0679-x

[36] Joao Fonseca Venetia Pliatsika. 2023. CSRankings: CSRankings data (2023). https://zenodo.org/records/11234896.

[37] Manisha Verma and Debasis Ganguly. 2019. LIRME: Locally Interpretable Ranking Model Explanation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 1281–1284. https://doi.org/10.1145/3331184.3331377

[38] Ke Yang, Joshua Loftus, and Julia Stoyanovich. 2021. Causal Intersectionality and Fair Ranking. In *Symposium on the Foundations of Responsible Computing FORC*. https://doi.org/10.4230/LIPIcs.FORC.2021.7

[39] Ke Yang, Julia Stoyanovich, Abolfazl Asudeh, Bill Howe, H. V. Jagadish, and Gerome Miklau. 2018. A Nutritional Label for Rankings. In *Proceedings of International Conference on the Management of Data, SIGMOD*. ACM, 1773–1776. https://doi.org/10.1145/3183713.3193568

[40] Jun Yuan and Aritra Dasgupta. 2023. A Human-in-the-loop Workflow for Multi-Factorial Sensitivity Analysis of Algorithmic Rankers. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA 2023, Seattle, WA, USA, 18 June 2023*. ACM, 5:1–5:5. https://doi.org/10.1145/3597465.3605221

[41] Meike Zehlike, Ke Yang, and Julia Stoyanovich. 2022. Fairness in Ranking, Part I: Score-Based Ranking. *ACM Comput. Surv.* (apr 2022). https://doi.org/10.1145/3533379

[42] Meike Zehlike, Ke Yang, and Julia Stoyanovich. 2022. Fairness in Ranking, Part II: Learning-to-Rank and Recommender Systems. *ACM Comput. Surv.* (apr 2022). https://doi.org/10.1145/3533380