RanXplain: Explaining Rankings in Recommendation Systems

Atakan Yilmaz, Nuray Eylul Erler, Emre Atilgan and Melisa Bal Aslan
Trendyol Group, Data Science
Istanbul, Turkey
Email: {atakan.yilmaz | eylul.erler | emre.atilgan | melisa.bal}@trendyol.com

***Abstract*—Recommendation systems are designed to rank items according to users' predicted interest. As these systems increasingly affect choices in domains like e-commerce and media, understanding the reasoning behind their rankings becomes essential. However, most existing approaches that explain recommendations focus on individual predictions, rather than explaining why one item is prioritized over another. To bridge this gap, this paper introduces RanXplain, an approach specifically designed to explain the ranking decisions produced by recommendation models. RanXplain operates as a separate machine learning model trained on pairs of items, using features that are derived from the original ranking model. The impact of different feature sets and model architectures on model performance is systematically investigated. Furthermore, a simulation based performance evaluation was presented on different breakdowns, specifically analyzing the proximity of item ranks and whether items belong to the same category to detect scenarios in which RanXplain yields superior performance. A practical insight is discussed regarding instances in which RanXplain fails to identify the ranking model's prioritization.**

Keywords-Recommendation System; Explainable AI (XAI); Machine Learning Explainability.

I. INTRODUCTION

Explainability in machine learning has become a cornerstone of responsible and trustworthy artificial intelligence, especially as these models are increasingly deployed in high-stakes and diverse domains, such as healthcare, finance, legal systems, and digital platforms. As predictive systems grow more complex, understanding how and why a model arrives at a particular decision is essential not only for debugging and improvement but also for ensuring fairness, accountability, and user trust. Therefore, developing effective methods to interpret machine learning models is crucial for aligning technical performance with ethical and practical expectations in real-world applications.

Recommendation systems, a key application of machine learning, have become integral in modern digital platforms, connecting users with relevant items across various domains, from e-commerce to entertainment. While traditional machine learning tasks provide precise point predictions, the core objective in recommendation systems is to accurately rank items based on users' predicted preferences. This change in focus underlines the need to adapt explainability techniques to better align with ranking based recommendation systems. Most of the existing explainability methods are effective for explaining individual predictions but they are often insufficient in expressing the comparative logic behind a generated ranked list. For instance, understanding why a model recommends "Item A" over "Item B" is crucial for user trust, system transparency, and even for identifying potential biases.

This paper introduces RanXplain, a methodology specifically designed to address this gap by explaining the comparative behavior of rankings generated by recommendation models. RanXplain functions as an independent machine learning model, trained on pairs of items recommended by the ranking model. It utilizes features derived from the original ranking model, enriched with additional comparison features that capture the differences between items. The application of both inherently explainable models and more complex, high-performing models were explored within RanXplain framework. The approach addresses the unique challenges of explaining rankings, offering flexible and detailed insights into why one item is placed above another in a recommendation list. By doing so, RanXplain aims to increase the transparency and interpretability of recommendation systems, promoting user understanding and trust.

The remainder of the paper is organized as follows: Section II reviews the related work on explainable AI and explanation methods. Section III introduces the RanXplain methodology in detail. Section IV offers the key results and experiments, along with a brief evaluation and discussion. Finally, Section V concludes the paper and outlines directions for future research.

II. RELATED WORK

Explainable Artificial Intelligence (XAI) is now one of the most important topics in many machine learning systems, due to the increasing need for transparency, trustfulness, and accountability [1][2]. With the high adoption of artificial intelligence in various fields, such as healthcare, banking, law, e-commerce, entertainment, interpreting predictions has been as important as creating the predictions themselves. Approaches to XAI may be categorized in terms of their usage with models and explaining the local or global behaviors.

1) Model-Intrinsic (or Inherently Interpretable) vs. Model-Agnostic (or Post-Hoc):
   - Model-intrinsic methods rely on the inherent transparency of certain machine learning algorithms, such as linear models or decision trees, whose internal structures make them naturally suitable for generating explanations.
   - On the other hand, model-agnostic methods are complementary for so-called black box models, such as neural networks, gradient boosting trees in a way that these methods are used after the predictions have been made.

These methods are, therefore, more flexible and may be used with any algorithm.

2) Local vs global explanations:
   - Local explanations aim to clarify individual input-output decisions, such as why a specific application was rejected or why a particular prediction probability was assigned.
   - Global explanations, however, try to give a general image of the behavior of the models and can be thought of as a summary of the model.

Local Interpretable Model-Agnostic Explanations (LIME) [3] and SHapley Additive exPlanations (SHAP) [4] are two popular model-agnostic local explanation approaches designed to explain any given black box classifier. Both of them work as feature attribution linear models, trying to understand the degree of change in predictions and particular features used to generate these predictions.

Even though they are extremely widely used and general, SHAP and similar feature attribution methods are basically limited [5][6][7], especially in ranking tasks. These methods are designed to explain instance-wise predictions by attributing the outcome to each feature one at a time. However, in recommendation systems, where the main task is to rank items relative to one another, such pointwise explanations are not able to capture the relative dynamics among items. For instance, the fact that the particular feature had a positive impact on the score of Item 1 tells us relatively little about the reasons why Item 1 outperformed Item 2. In Figure 1, row-wise SHAP-style feature attributions for the top four recommended items for a user are shown to illustrate this limitation. Each row corresponds to one item, with SHAP values color-coded based on their magnitude and impact within that row. Green indicates positive contribution toward the item's ranking score, and red indicates negative contribution. Though single-item contributions are formulated for each item, they do not provide insight into relative differences that cause the ensuing ranking order. A seemingly logical, yet misleading, approach would be to simply compare feature contributions between two items. For instance, the SHAP value for price feature (Feature 1) could be positive for Item 1 and negative for Item 2. This large, opposing difference in SHAP values might incorrectly suggest that price is the primary reason for the ranking disparity. In reality, Item 2 can be cheaper than Item 1 and other features like user affinity for specific categories (or brands) might be the true drivers, creating these conflicting individual attributions. This means a feature crucial for an item's individual score may be irrelevant when explaining its comparative rank.

Global explanation methods like Permutation Feature Importance [8] or Partial Dependence Plots [9] similarly fall short in explaining the behavior of ranking models. While they can identify influential features on average across predictions, they do not provide specific, contextual information. For example, price is generally the most important factor for ranking models in e-commerce; however, it does not explain why, for a particular user and context, a more expensive Item A might be ranked higher than a cheaper Item B, contrary to average user behavior. These gaps highlight that neither standard local nor global approaches are inherently suited to the comparative nature of ranking explanations, motivating the need for specialized pairwise or listwise approaches.

Item ID	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8	Score	Rank
1	0.73	0.92	0.56	0.36	0.25	0.11	-0.74	0.58	2.77	1
2	-0.46	0.14	0.10	0.95	-0.04	0.76	0.42	0.04	1.91	2
3	0.51	-0.08	-0.56	0.49	0.84	0.59	-0.24	-0.18	1.37	3
4	0.57	-0.58	-0.71	0.02	0.84	0.77	0.16	0.19	1.27	4

Figure 1. Row-wise SHAP-style feature attributions.

One of the most influential pairwise approaches is the Analytic Hierarchy Process (AHP) and its generalization, the Analytic Network Process (ANP), introduced by Saaty [10][11] for decision-making based on pairwise comparisons. In AHP/ANP, decision-makers explicitly provide judgments on the relative importance of alternatives or criteria, and a priority ranking is then derived using the principal eigenvector of the comparison matrix. This framework has been widely applied in domains, such as project selection, resource allocation, and policy evaluation. The RanXplain framework, however, addresses the inverse problem: instead of deriving rankings from human-provided comparisons, it seeks to explain rankings that have already been produced by machine learning models. While one might envision applying Saaty's eigenvector method directly to model-generated pairwise scores, several practical obstacles arise.

First, the scale of modern recommender systems far exceeds the typical scope of AHP/ANP: a single user session may involve thousands of candidate items (e.g., in e-commerce with catalogs exceeding 10 million products) and hundreds of input features (e.g., user–item embeddings, contextual features, temporal recency signals). Constructing and processing complete $n \times n$ pairwise matrices under such conditions becomes computationally intractable. Second, the eigenvector solution yields overall item priorities but does not provide feature-level contributions to rankings, which are essential for transparency in explainable AI. Third, while AHP assumes relatively stable and consistent comparison judgments, machine-learned rankings are highly context-dependent, with the relative importance of features varying substantially across users and sessions. These distinctions underscore why classical AHP/ANP methods are not directly applicable to explaining large-scale AI ranking systems.

In the following sections, a comparative RanXplain methodology will be discussed in detail on how to mitigate the gaps of the current methods of XAI.

III. Methodology

The methodological framework for the RanXplain model outlined in this section, addresses the aforementioned limitations of existing explainability methods in ranking. RanXplain provides explanations for pairwise preferences within a ranked list of items, clarifying the comparative reasoning of the original ranking model. Effectively, RanXplain operates as a seperate

machine learning model, trained to explain the primary ranking system's comparative behavior.

A. Data Generation for RanXplain: The Pairwise Paradigm

RanXplain focuses on enhancing a personalized recommendation system in terms of explainability. The underlying notion behind RanXplain is to transform the complex problem of explaining the ranking of the whole item list into a series of manageable binary classification problems based on pairwise comparisons. A training instance is constructed for RanXplain, for each relevant pair of items derived from the output of the primary ranking model.

In personalized recommendation systems, the common approach involves generating pointwise predictions for individual user-item pairs. Items are then ranked for each user based on these scores. However, while it might be possible to explain why a single item received a particular prediction score (although even this is often challenging with typical ranking models), it's rarely clear why "Item A is ranked higher than Item B." This explanation is often more intuitive for users trying to understand their preferences.

This lack of clarity regarding relative rankings makes it difficult for both users and developers to grasp the underlying behavior of the recommendation framework. RanXplain addresses this explanatory deficiency by evaluating ranking model behavior through considering combinations of items.

1) Selection of Pairs: RanXplain relies on modeling pairwise preferences to effectively explain the comparative logic of the primary ranking model. However, it is computationally challenging to generate every possible combination from a large set of items. Therefore, a strategic approach to sampling these pairs is vital, not only for practical implementation but also to ensure the most informative pairs of items are included.

The preferred methodology for generating these pairwise comparisons involves two main strategies, both beginning by determining top K items for each user from their recommendation lists.

The first strategy for generating user-item-item indices involves randomly selecting a subset of k ($k < K$) items for each user, from their selected top K recommendations. All possible pairwise combinations are then created from this subset. This ensures that each item within the chosen subset appears in multiple comparisons for that user, providing a substantial set of data for learning specific comparative preferences of the ranking model.

The second strategy initially forms all possible combinations from the entire set of K top items for each user. Then, a random sampling is applied to obtain a comprehensive collection of pairwise comparisons from this potentially vast dataset. This strategy differs from the first as it creates a subset of the original dataset rather than representing the full data. While this can make the model more robust, it has a key drawback: it might miss some pairwise comparisons between items. For example, if we consider three items (i_1, i_2, and i_3) recommended to a user, the first strategy includes all pairwise comparisons (i_1 vs. i_2, i_2 vs. i_3, and i_1 vs. i_3). In contrast, this strategy might include only some of these pairs, which makes it harder to capture three-way (or higher-order) relationships. Furthermore, this approach may introduce greater imbalance in the number of data points per user, which can lead to biased training or decreased generalization performance.

2) Features of RanXplain: Creating meaningful features is crucial for the RanXplain model to learn from and explain the comparative relations. Original feature set $F = \{f_1, f_2, \ldots, f_N\}$ which were used by the primary ranking model to make pointwise predictions are added to the feature set for both items in each pair (i_1, i_2), so that the feature set of RanXplain contains $2N$ item features for each index since both items have N features.

Additionally, a set of comparison features that explicitly capture the relationship between i_1 and i_2 are derived from the features in F. Let x_1 and x_2 be the values of a feature $f_j \in F$ for i_1 and i_2, respectively. A small constant ε (e.g., 10^{-6}) is introduced to handle potential division by zero. Using x_1, x_2, and ε, a set of comparison features F_{comp} is constructed as follows:

Ratio: The ratio of feature values for items i_1 and i_2 is defined as shown in (1):

$$\frac{x_1}{x_2 + \varepsilon} \tag{1}$$

Mean Percentage Error (MPE): The MPE between feature values, as calculated in (2), is computed as:

$$\frac{x_1 - x_2}{x_1 + x_2 + \varepsilon} \tag{2}$$

Difference: The absolute difference between feature values is simply expressed by (3):

$$x_1 - x_2 \tag{3}$$

Relative Deviation: The relative deviation, given by (4), captures the proportional difference:

$$\frac{x_1 - x_2}{x_1 + \varepsilon} \tag{4}$$

Equality Indicator: For categorical features, an indicator function checks equality, as defined in (5):

$$\mathbf{I}_{x_1 = x_2} = \begin{cases} 1 & \text{if } x_1 = x_2, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The full feature set of RanXplain includes both item features from the original ranking model and features that describe the comparison between item pairs in order for the model to learn more detailed comparison logic. All different combinations of feature sets have been tested by adding and discarding them to optimize the feature set for effective comparative learning.

3) Target Variable: The target variable for RanXplain is a binary indicator, which has the value of 1 if the first item i_1 in the pair (i_1, i_2) is ranked higher by the primary ranking model. If the second item i_2 is ranked higher, the target is 0. This approach turns the primary model's unknown pairwise decisions into a clear, learnable signal for RanXplain.

B. RanXplain Model Selection

Selecting the type of underlying machine learning model is critical for development of RanXplain. It requires balancing robust predictive performance with the need for interpretability when explaining comparative behaviors. Logistic Regression and XGBoost are considered as two prominent models for this purpose.

- Logistic Regression: Initially advanced by [12] and further generalized by [13], Logistic Regression is a linear model which is particularly advantageous for its inherent interpretability in binary classification. RanXplain's aim of interpreting ranking behavior by classifying pairwise preferences, directly aligns with capability of this model type. Within RanXplain, Logistic Regression models the probability that i_1 is prioritized over i_2 by the primary ranking model. Its direct interpretability comes from its learned coefficients:
  - A positive coefficient for a feature $f_j(i_1)$ indicates that an increase in f_j for i_1 directly raises the probability of i_1 being preferred, assuming other features remain constant.
  - Critically, for comparison features, such as $f_j(i_1) - f_j(i_2)$, a positive coefficient directly quantifies that a higher difference in f_j in favor of i_1 contributes proportionally to its higher predicted preference.

  This direct mapping between feature values and their impact on the log-odds of preference provides transparent and comprehensible explanations for the primary ranking model's comparative logic. Its main limitation in this context is its inability to capture complex non-linear relationships or feature interactions that may characterize the primary ranking model's decision-making process.
- XGBoost (Extreme Gradient Boosting): An optimized gradient boosting framework which is introduced by [14], offers superior predictive performance by constructing an ensemble of decision trees. While inherently a black-box model, its utility within RanXplain for generating explanations is realized through the application of SHAP values. SHAP provides a robust, unified framework to attribute the contribution of each feature to a specific prediction.
  - For a RanXplain model trained with XGBoost, SHAP values precisely quantify the impact of each feature on the prediction of whether i_1 is preferred over i_2. This enables local explanations for individual pairs (e.g., attributing i_1's preference to its higher "discount" and "popularity" differential).
  - Furthermore, aggregating SHAP values enables global insights into the most important features effecting comparative preferences across the entire dataset (e.g., identifying "price difference" as a universally strong determinant of higher ranking).

XGBoost's advantage lies in its competency to model complex non-linear relationships and high-order feature interactions, potentially offering a more accurate representation of the primary ranking model's intricate decision boundaries. The need for post-hoc explanation methods like SHAP is the disadvantage of using XGBoost for RanXplain. Although SHAP is a powerful method to produce explanations, it is more complex and computationally intensive than using direct coefficients from Logistic Regression.

C. Explanation Generation and Presentation

The practical applicability of the RanXplain methodology extends beyond its predictive capacity, addressing the non-trivial step of translating its output into useful, understandable explanations for end-users and system designers. This process is fundamentally guided by the ability to use the model's internal feature weights and contributions to pinpoint the most influential factors in a ranking decision.

Consider a real-world e-commerce scenario in which a recommendation system presents a user with a ranked list of products. Within this list, two items are of particular interest: Item A, an expensive shoe from a well-known brand with an applied discount, and Item B, a medium-priced shoe from a common brand without a discount. The primary ranking model prioritizes Item A over Item B, and RanXplain successfully predicts this outcome.

When RanXplain correctly predicts the prioritization of one item over another, its model coefficients (for Logistic Regression) or feature importance values (for XGBoost) reveal which comparison features were most influential. For instance, the model can identify that the difference in discount ratio, relative brand popularity, or the user's affinity for a specific brand were the key drivers behind the ranking. These features, which quantify the relative properties of the two items, allow for the generation of clear and concise explanations.

This capability enables the extraction of concrete insights, such as: "Item A was ranked higher than Item B because, while Item B is cheaper, the model gave more weight to the discount available on Item A and the user's affinity for Item A's brand." This ability to generate detailed, feature-based explanations serves several primary purposes in real-world applications:

- User Trust and Understanding: Providing explanations for why a specific item was prioritized helps users understand the system's logic, leading to increased trust and confidence in the recommendations.
- System Debugging and Improvement: Explanations act as a critical tool for developers to diagnose the primary ranking model's behavior. By analyzing why certain items are ranked in a particular order, developers can identify potential biases, correct model errors, and gain insights for future feature engineering.
- Cross-functional Insights: Explanations can be shared with other teams (e.g., merchandising, marketing) to provide a deeper understanding of customer behavior and content performance. For example, by analyzing explanations, a merchandising team could determine that a 10% price decrease on a specific product would cause it to be ranked higher than a competitor's product for a particular segment of users.

IV. RESULTS | EVALUATION

Experimental evaluation of RanXplain involves a rigorous process, beginning with the detailed construction of three distinct datasets: (i) training set, (ii) test set and (iii) simulation set. The training dataset was formed using top 50 recommendations per user generated by the ranking model within a specified historical period. It consists of 4.5 million rows by over 30,000 unique users and more than 130,000 distinct items while maintaining a balanced 50% target ratio. As the test set has been obtained by splitting the initial training set according to 80%-20% parity, it contains 1.1 million rows while exhibiting comparable unique user and item counts and the same 50% target ratio. Crucially, simulation dataset, consisting of 50 million rows, was generated by incorporating all 50 top-ranked items for each user from a later temporal period than the training set, including approximately 45,000 users and over 175,000 distinct items, also with a 50% target ratio.

The choice of sampling strategy is crucial for both the practical impact and computational efficiency of RanXplain. By transforming the complex task of explaining a ranked list into a series of pairwise classification problems, RanXplain becomes computationally tractable for large-scale recommendation systems, which is a significant advantage over other methods. Two different sampling strategies were explored for training RanXplain: (i) content-based sampling and (ii) random sampling. Performance metrics of the models trained on both datasets were observed as highly similar. However, content-based sampling yielded slightly superior performance and provided a more representative distribution across diverse users. This intentional sampling approach makes the training process more efficient and ensures that the resulting explanations are representative and of high quality, which is vital for real-world application. Consequently, content-based sampling method was adopted by randomly selecting 20 items per user from their top 50 recommendations and generating all possible combinations for the selected 20 items.

Experiments of RanXplain proceeded to exploring two critical dimensions in more detail: feature set composition and model architecture, concluding with a detailed simulation-based performance evaluation.

A. Experimentation of Feature Sets

To investigate the impact of features on RanXplain model, a set of experiments were conducted. Table I depicts performance metrics across train, test and simulation datasets of the Logistic Regression models trained with different feature sets in BigQuery ML [15]. Performance of the models were assessed using the Receiver Operating Characteristic Area Under the Curve (ROC-AUC), which quantifies the ability of a classifier to discriminate between positive and negative classes across various thresholds [16]. Similar behavior was observed across other performance metrics, such as accuracy and recall.

Initially, Model 1 was trained using only item features which is an approach that mirrored the original ranking model. By incorporating comparison features alongside these item features in Model 2, a significant improvement in model performance was observed across all ROC-AUC metrics for the training, test, and simulation datasets.

TABLE I. RANXPLAIN MODEL PERFORMANCE WITH DIFFERENT FEATURE SETS

Metric	Model 1	Model 2	Model 3
Item Features	Included	Included	Excluded
Comparison Features	Excluded	Included	Included
Train ROC-AUC	0.62	0.73	0.74
Test ROC-AUC	0.61	0.74	0.74
Simulation ROC-AUC	0.61	0.69	0.70

Interestingly, Model 3 which is trained exclusively with comparison features achieved slightly better predictive performance than the models with item features. While the predictive gains were marginal, using only comparison features significantly enhanced the qualitative aspect of explanations compared to Model 2. The increase in qualitative aspect is due to the directness of interpretability that comparison features provide when comparing two items.

Slightly improved performance along with stronger interpretability indicates the vital role of comparison features in accurately capturing the relative ranking of items. Therefore, the comparison features are adopted as the feature set of RanXplain.

B. Experimentation of Model Types

For model selection, Table II shows performances of models that differ by model type and maximum tree depth. Although Model 3 was the best performer in the experiments of feature sets, Model 2 was chosen as a baseline model to be compared with XGBoost models (which are trained using BigQuery ML [17]) so that both item and comparison features are included in experimentation of model types.

TABLE II. RANXPLAIN PERFORMANCE FOR DIFFERENT MODELS

Metric	Model 2	Model 4	Model 5
Model Type	Log Reg	XGBoost	XGBoost
Max Tree Depth	-	15	5
Train ROC-AUC	0.73	0.92	0.79
Test ROC-AUC	0.74	0.92	0.79
Simulation ROC-AUC	0.69	0.78	0.69

Reducing the maximum tree depth in the XGBoost model causes significant decrease in model performance across all train, test and simulation sets. This decrease is evident in Table II, as shown by the performance difference between Model 4 and Model 5. This observation motivates the use of a more sophisticated XGBoost model within RanXplain. Additional complexity is required to effectively approximate the behavior of primary ranking model, which is a highly complex model. However, while higher complexity increases the prediction performance, it also makes the interpretation of the explanation model more challenging.

It can be inferred from Table II that Model 2 underperformed Model 4 with respect to ROC-AUC metrics. However, Logistic Regression possesses inherent interpretability as opposed to XGBoost which requires additional methods like SHAP for explanations. Although more complex models like XGBoost might be a better fit depending on the specific application's requirements, Logistic Regression is found more suitable for RanXplain of the primary ranking model that is used in this study.

C. Simulation Based Performance Evaluation

Simulation dataset was used to conduct various offline evaluations on the preferred model (Model 3). Consistent ROC-AUC performance was observed across the training, test, and simulation datasets, with only a slight performance decrease on the simulation set. Further analyses were conducted on the simulation data to understand the behavior of RanXplain model more comprehensively. These analyses are based on two key factors: (i) proximity of item ranks in the original ranking model and (ii) whether item pairs belonged to the same high level item category (e.g., electronics category).

Table III depicts train, test and simulation performances of Model 3. Simulation performance was analysed with respect to three additional breakdowns: subset of the simulation data (i) where the difference between rankings of two items are greater than 20 ($r_{diff} > 20$) (ii) where the difference between rankings of two items are less than or equal to 3 ($r_{diff} \leq 3$) and (iii) where the two items belong to the same category (Same Category).

TABLE III. SIMULATION PERFORMANCE

Metric	Model 3
Train ROC-AUC	0.74
Test ROC-AUC	0.74
Simulation ROC-AUC	0.70
Simulation ROC-AUC ($r_{diff} > 20$)	0.80
Simulation ROC-AUC ($r_{diff} \leq 3$)	0.54
Simulation ROC-AUC (Same Category)	0.70

The results revealed a clear trend, predictive capability of the model significantly improves as the rank difference between item pairs increases. For example, the model performed substantially better when the rank difference exceeded 20, achieving an ROC-AUC of 0.80. On the contrary, performance dropped considerably for closely ranked items (($r_{diff} \leq 3$)), with an ROC-AUC of approximately 0.54. This finding indicates that RanXplain has difficulty in predicting (and thus explaining) prioritization when the primary ranking model assigns similar scores to items, which is expected.

This behavior is a key advantage of the RanXplain approach, as it allows us to know in advance when its outputs can be used to confidently interpret the ranking model's decisions, thereby preventing misleading or false insights. The correctness of RanXplain's predictions (and therefore their reliability for generating insights) is known in advance, since the real rankings are already known. This enables the clear identification of when it is safe to use RanXplain's outputs to interpret the behavior of the underlying ranking model for specific item pairs, thereby avoiding misleading or false insights.

Regarding category influence, RanXplain's predictions for pairs within the same item category were very similar to its performance on pairs from the whole simulation set, indicating no significant performance differential. Consequently, for the application of this study, improving RanXplain's performance on closely ranked pairs is of minor importance, although such improvements are feasible by adjusting sampling strategies or incorporating additional comparison features.

V. Conclusion and Future Work

This paper introduced RanXplain, a methodology designed to address a significant gap in recommendation systems, which is the need to explain ranking decisions rather than individual item predictions. As outlined in the previous sections, RanXplain functions as a seperate machine learning model trained on item pairs, employs features derived from the original ranking model. Both the effectiveness and operational behavior of RanXplain is illustrated through a systematic investigation of various feature sets and model architectures, along with simulation-based performance evaluation.

The main contribution of RanXplain lies in shifting the focus of explainability from pointwise predictions to the comparative logic behind ranked outputs. RanXplain enables a more intuitive and actionable understanding of why one item is ranked above another by reframing the task of explaining a ranked list as a series of pairwise classification problems. The aim is to provide interpretable insights into the decision-making process of black-box recommendation models, supporting user trust and contributing to system debugging.

The evaluation based on ROC-AUC across various datasets highlighted the strong influence of comparison features. Models trained exclusively on these features not only achieved better predictive performance but also yielded more interpretable explanations as a result of the direct relevance of the input features. While more complex models, such as XGBoost, offered better predictive performance, Logistic Regression proved to be more suitable for applications that require interpretability, even at a modest cost to accuracy.

The simulation based evaluation further revealed that RanXplain's predictive performance improves significantly as the rank difference between items increases. On the other hand, its performance naturally decreased when items were very closely ranked, which is expected given that the ranking model assigns similar scores in such cases. It is important to note that one of RanXplain's primary advantages is that the correctness of its predictions is known in advance, since the ground truth rankings are available. This capability allows for the identification of cases where RanXplain's outputs can be confidently used to interpret the ranking model's decisions, thus avoiding potential misinterpretations. This observation also draws a parallel to the concept of rank reversal in pairwise

comparison methods like Saaty's AHP, suggesting that the underlying ranking decisions for closely-ranked items are inherently more ambiguous and less stable, making them difficult to explain with high confidence.

The approach shows promise in interpreting pairwise relative rankings; however, RanXplain is not designed to provide a single, holistic explanation for an entire ranked list. While a high-level explanation might be desirable, it can often be too simplistic to capture the nuanced decision-making process of a complex ranking model. Instead, RanXplain provides a series of granular, actionable insights. An explanation for an entire ranked list can be composed by chaining together a series of pairwise comparisons, such as explaining why Item 1 was ranked above Item 2, why Item 2 was ranked above Item 3, and so on. This approach offers a more detailed and accurate understanding of the ranking process, as it clearly articulates the specific feature-level trade-offs that led to the final ordering. This modular nature allows RanXplain to provide highly specific insights on demand, supporting both user understanding and system debugging by clarifying the reasons behind individual ranking decisions.

For future work, several promising directions can be explored to further enhance RanXplain. The AUC performance of the model, particularly on closely ranked pairs, can be enhanced through various methods. This could involve incorporating additional non-linear comparison features, such as the power of the difference of feature values, to better capture the primary model's complex decision boundaries. Furthermore, exploring alternative and more advanced sampling techniques or using a wider range of training data could lead to significant improvements in model performance and a more robust understanding of the ranking model's behavior. An extension of RanXplain to support counterfactual explanations could offer more actionable insights for users and system designers by indicating how changes in specific features would affect the relative ranking of items. The trade-off between user-based and random sampling, and how different sampling strategies impact the quality of explanations, presents a key area for further research. RanXplain can also be used in a reverse engineering context to guide feature design in the original ranking model. When important comparison features are identified but prove insufficient on their own, new supporting features can be introduced to the original ranking model. This can improve both the model's performance and its explainability.

ACKNOWLEDGMENT

This study was supported by Trendyol's R&D Center through infrastructure and organizational contributions. The work was carried out within the scope of an R&D project approved by the Republic of Türkiye Ministry of Industry and Technology.

REFERENCES

[1] V. Hassija et al., "Interpreting black-box models: A review on explainable artificial intelligence", *Cognitive Computation*, vol. 16, 2023. DOI: 10.1007/s12559-023-10179-8

[2] Y. Zhang and X. Chen, "Explainable recommendation: A survey and new perspectives", *Foundations and Trends in Information Retrieval*, vol. 14, no. 1, pp. 1–101, Mar. 2020. DOI: 10.1561/1500000066

[3] M. T. Ribeiro, S. Singh, and C. Guestrin, ""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 (KDD '16)*, New York, NY, USA: Association for Computing Machinery, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778

[4] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions", in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*, Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 4768–4777.

[5] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods", in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ACM, 2020, pp. 180–186. DOI: 10.1145/3375627.3375830

[6] S. Bordt, M. Finck, E. Raidl, and U. von Luxburg, "Post-hoc explanations fail to achieve their purpose in adversarial contexts", in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*, New York, NY, USA: Association for Computing Machinery, 2022, pp. 891–905. DOI: 10.1145/3531146.3533153

[7] H. Lakkaraju and O. Bastani, ""how do i fool you?": Manipulating user trust via misleading black box explanations", in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES '20)*, New York, NY, USA: Association for Computing Machinery, 2020, pp. 79–85. DOI: 10.1145/3375627.3375833

[8] J. H. Friedman, "Greedy function approximation: A gradient boosting machine", *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[9] L. Breiman, "Random forests", *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] T. L. Saaty, *The Analytic Hierarchy Process*. McGraw-Hill, 1980.

[11] T. L. Saaty, *Decision Making with Dependence and Feedback: The Analytic Network Process*. RWS Publications, 1996.

[12] J. Berkson, "Application of the logistic function to bio-assay", *Journal of the American Statistical Association*, vol. 39, no. 227, pp. 357–365, 1944.

[13] D. R. Cox, "The regression analysis of binary sequences", *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, 1958.

[14] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, San Francisco, CA, USA: ACM, Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785

[15] Google Cloud, "Create model statement for logistic regression models (glm)", Accessed: 2025.07.18, 2024. [Online]. Available: https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-create-glm

[16] T. Fawcett, "An introduction to roc analysis", *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. DOI: 10.1016/j.patrec.2005.10.010

[17] Google Cloud, "Create model statement for boosted tree models (boosted_tree)", Accessed: 2025.07.18, 2024. [Online]. Available: https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-create-boosted-tree