Explaining LTR models

Part 1. Blog post on how to use SHAP for interpretability of LTR models.

https://sease.io/2020/07/explaining-learning-to-rank-models-with-tree-shap.html

WHAT IS SHAP.

Tree SHAP allows us to give an explanation to the model behavior, in particular to how each feature impact on the model's output. Here each output/prediction is seen as a sum of the contribution of each individual feature.

- Global interpretability: through summary plots. These reflect the general behavior of the features in the model and allow us to understand which are the features that most impact the final output and of how much.

- Local interpretability: through force/dependence plots. These show the specific behavior of the features in a single model prediction, allowing us to understand all their single impacts on the final output.

- It can be used for several machine learning models: SHAP provides several explainers that cover most of the machine learning methods. These are a collection of classes, where each class represents an explainer for a specific machine learning algorithm. The explainer is the object that allows us to understand the model behavior.

UNDERSTAND THROUGH AN EXAMPLE

Suppose to be in a learning to rank scenario.

We have to manage a book catalog in an e-commerce website. Each book has many different features such as publishing year, target age, genre, author, and so on.
A user can visit the website, make a query through some filters selection on the books' features, and then inspect the obtained search result page.
In order to train our model, we collect all the interactions that users have with the website products (e.g. views, clicks, add to cart, sales..) and create a data set consisting of <query, document> pairs (e.g. the filters selected and the features of the product viewed/clicked/sold/ …).

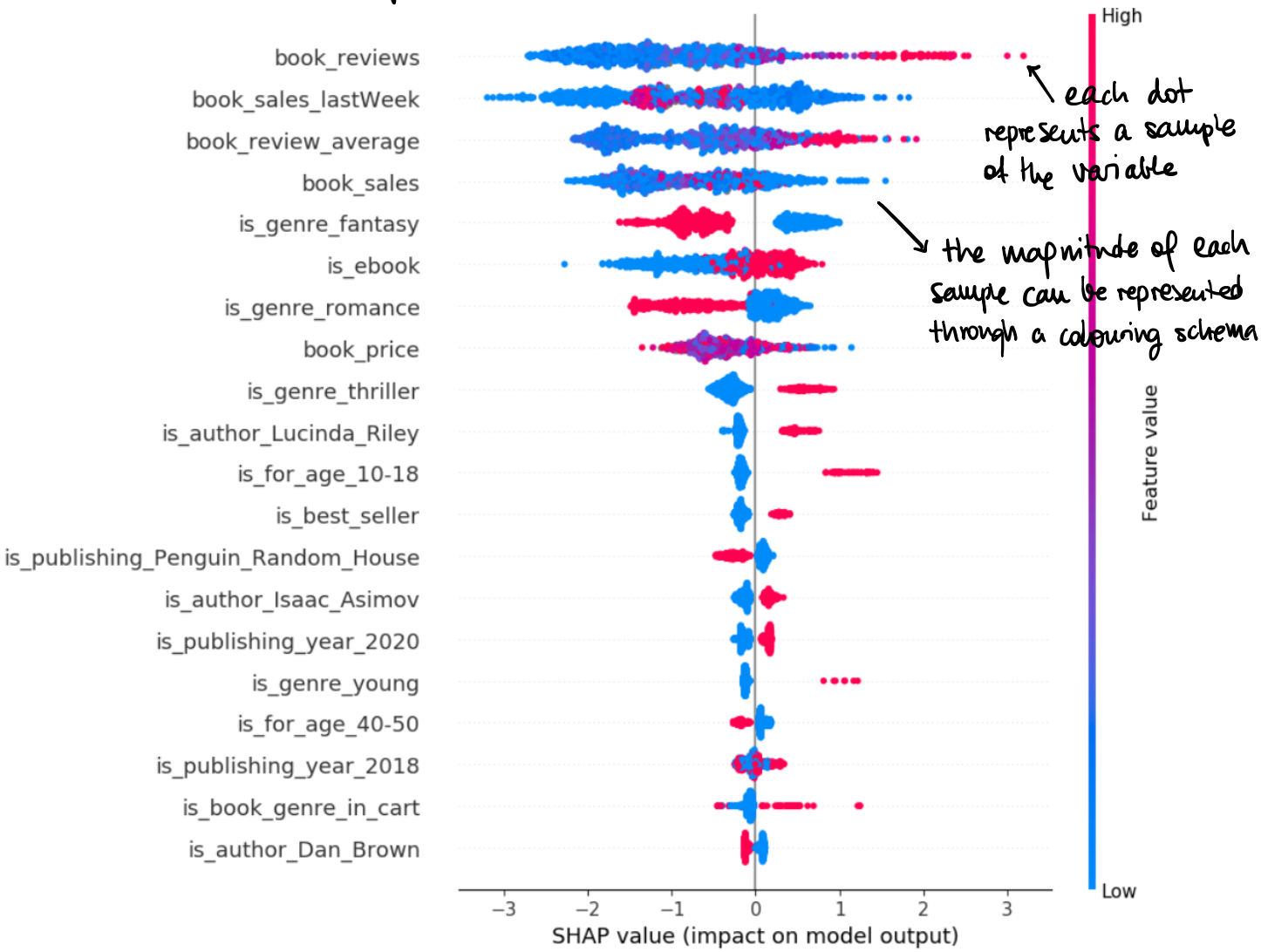
We obtain something like this, where s_feature indicates the selected feature from the website filters and book_feature the feature of the product with which the user interacted

Interaction	filter_genre	filter_price range	book_genre	book_author	book_price
1	thriller	[15-20]	thriller	Brown	18
2	mystery	[5-10]	mystery	Christie	8
3	fantasy	[10-15]	fantasy	Asimov	15

when modelling, these features can be one-hot encoded.

SUMMARY PLOT

features are ordered by most impactful in descending order. This can be seen by the range of SHAP values (the wider the range implies that small changes in the variable value can lead to bigger effects)



each dot represents a sample of the variable

the magnitude of each sample can be represented through a colouring schema

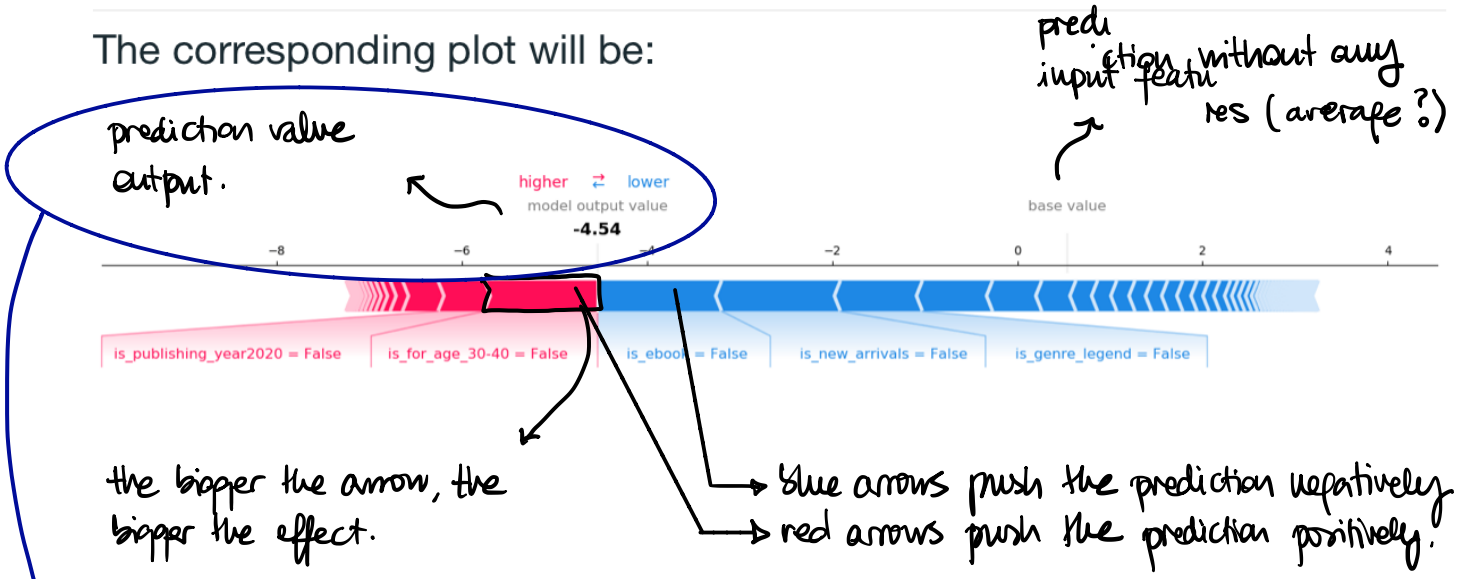
In this example, we could interpret that:

- The higher the number of reviews, the higher positive impact on the relevance.
- The higher the average review, the higher positive impact on relevance.
- Ebooks are more relevant
- Genre fantasy is less relevant.
- There are also features like book sales where it is not clear, or, that it is not impactful (neither + or −) in terms of relevance.

FORCE PLOT (SINGLE PREDICTION PLOT)

Interaction	is_publishing_year_2020	is_for_age_40	is_ebook	is_new_arr
interaction_4	False	False	False	False

The corresponding plot will be:



predi ction without any input featu res (average ?)

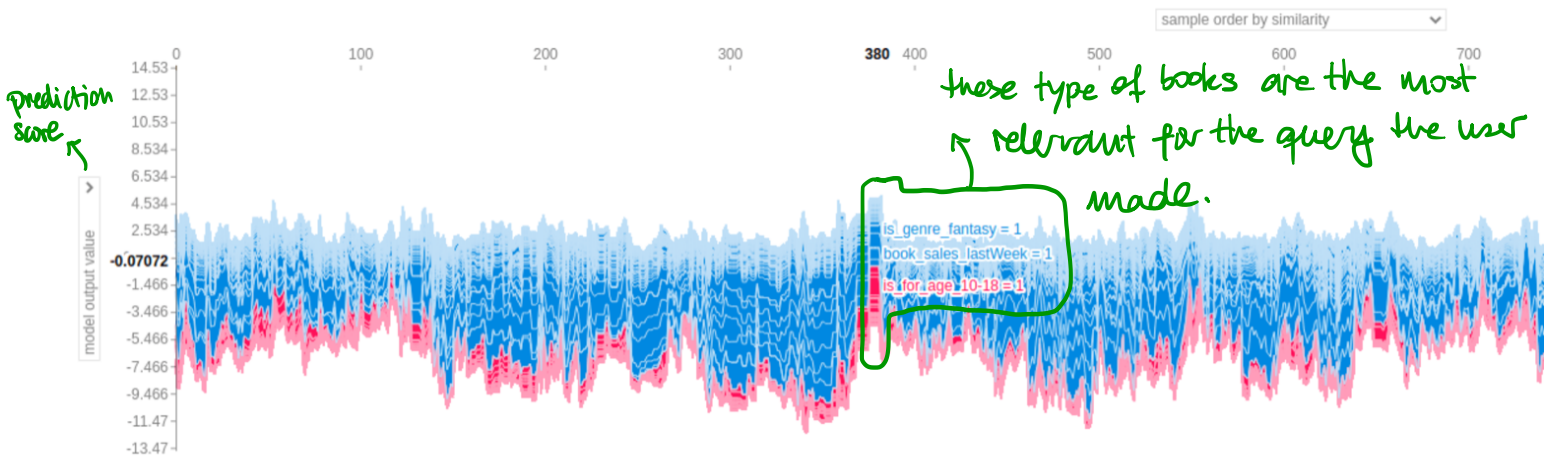
In a learning to rank scenario, this output represents the relevance score of the book.

Interaction	Query	Book	Prediction
1	1	1	-2.91
2	1	2	-4.54
3	1	3	-1.85

→ Presentation to the user

1st → Book 3
2nd → Book 1
3rd → Book 2.

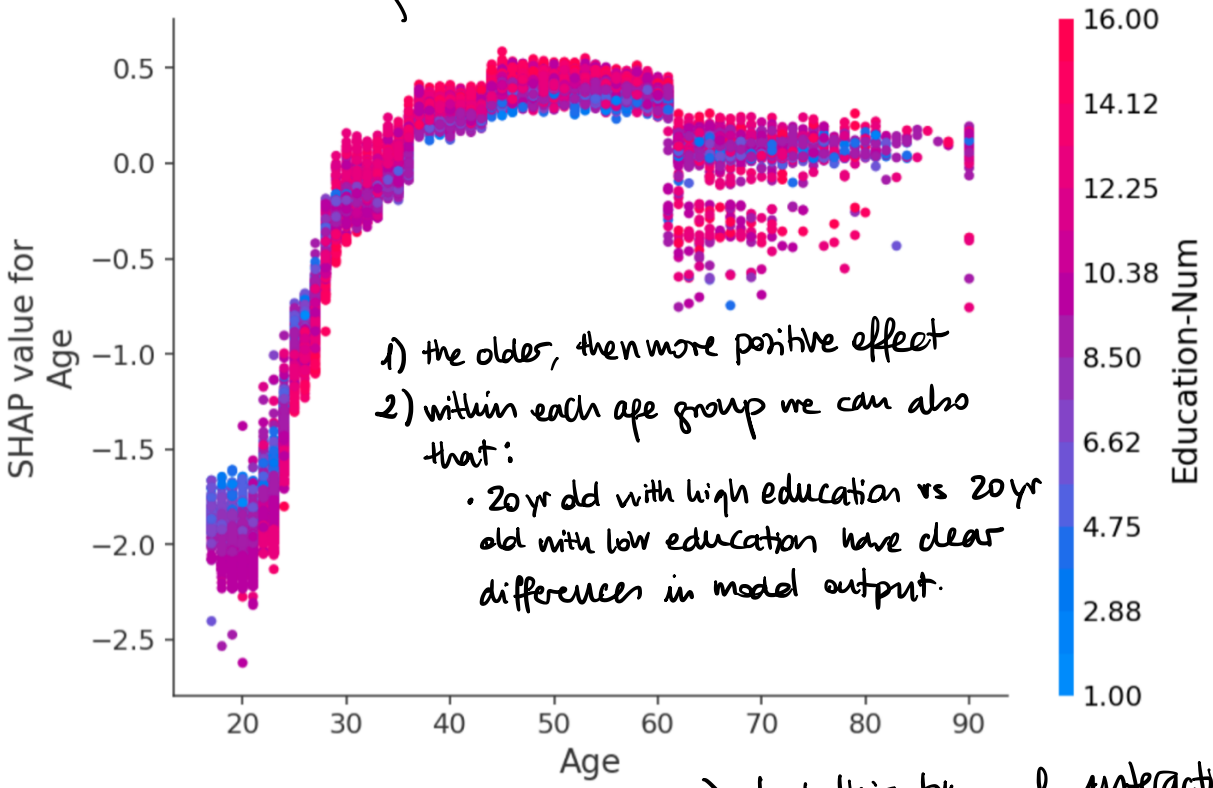
You can plot each individual prediction in a plot like the one below.



DEPENDANCE PLOT : This plot is also very useful for local interpretability, where we can check if there are interactions between 2 features.

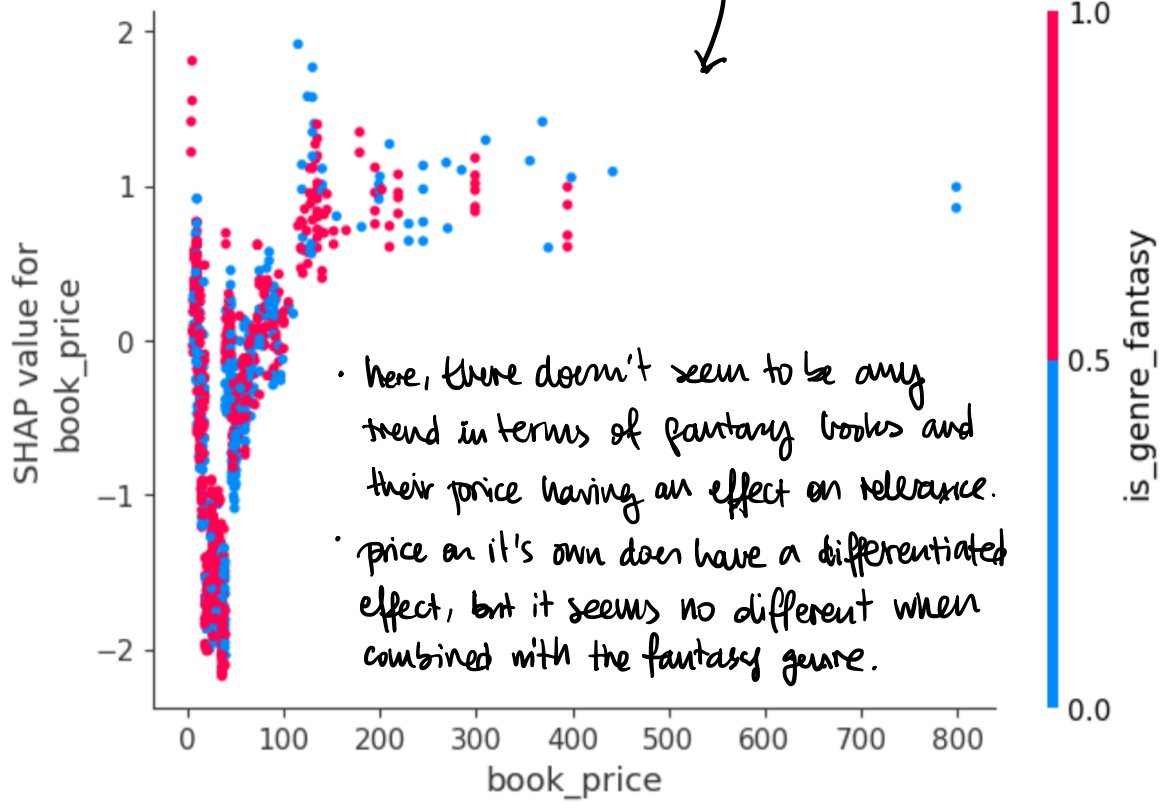
↳ in the plot below, we check if there is interaction between age and education.

each point represents a prediction



1) the older, then more positive effect
2) within each age group we can also that:
   - 20 yr old with high education vs 20 yr old with low education have clear differences in model output.

but this type of interactions don't always exist.

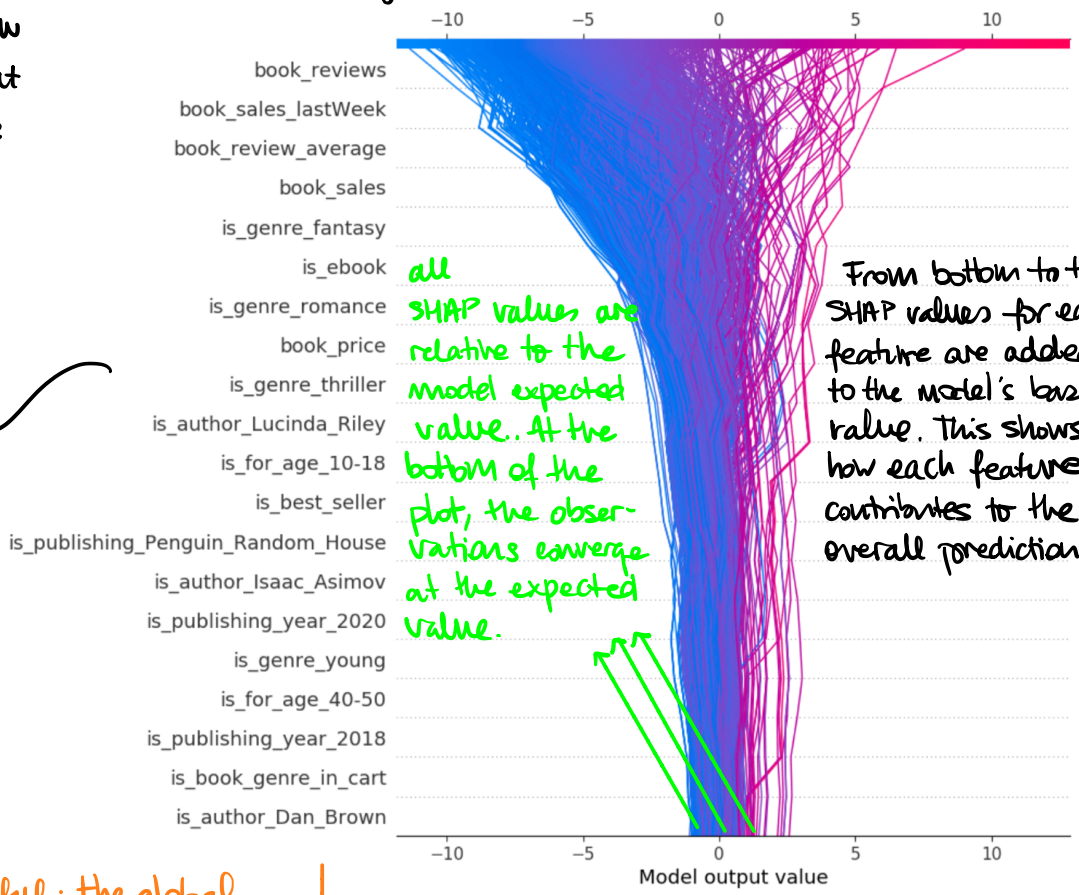


- here, there doesn't seem to be any trend in terms of fantasy books and their price having an effect on relevance.
- price on it's own does have a differentiated effect, but it seems no different when combined with the fantasy genre.

DECISION PLOT

- Decision plots show how complex models arrive at their predictions (make decisions)

Each prediction (line) strikes at the top of the x-axis at it's corresponding predicted value.



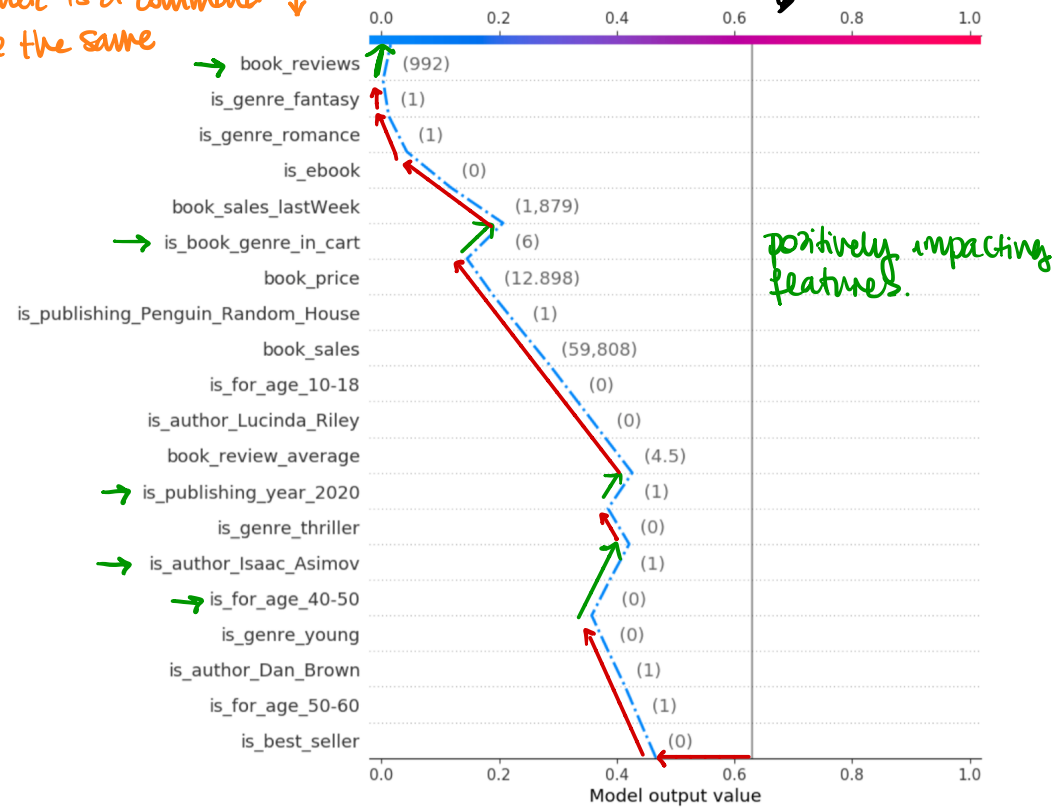
all SHAP values are relative to the model expected value. At the bottom of the plot, the observations converge at the expected value.

From bottom to top, SHAP values for each feature are added to the model's base value. This shows how each feature contributes to the overall prediction

this is the same ordering as shown in the summary plot (features ordered by impact on output)

careful: the global importance order doesn't have to match the order for a single prediction (there is a command to force the same order)

The decision plot supports a link logit function to transform log odds to probabilities



positively impacting features.

CAREFUL

1) The measure of how important is a feature is calculated based on the interactions on the ENTIRE set.
   - ↳ Interactions AREN'T considered grouped by query.
   - ↳ Therefore, we would have to run these plots for individual queries (or maybe for grouped queries)

2) The output of the model is measure of relevance generated by the SHAP library
   - ↳ It still reflects relative importance:
     - If SHAP outputs a score of 2 for document 1 and score of -1 for document 2, this will still accurately represent that document 1 is more relevant than document 2.
   - ↳ But, to be clear, It is NOT the actual relevance label. You will not get the exact output result.
   - ↳ This is why negative values in SHAP don't represent that a document is not relevant. They represent that a document is LESS relevant, as SHAP will consider it in RELATION with the other products in the query

Part 2: Paper about Valid explanations for learning to Rank models.

Valid Explanations for Learning to Rank Models

Jaspreet Singh[1], Zhenye Wang[2], Megha Khosla[1], and Avishek Anand[1,2]

[1] L3S Research Center, Hannover, Germany.
[2] Leibniz Universität Hannover, Appelstrasse 4, 30167 Hannover, Germany.
{singh,khosla,anand}@l3s.de, bismarckderzweit@gmail.com

Abstract. Learning-to-rank (LTR) is a class of supervised learning techniques that apply to ranking problems dealing with a large number of features.
The popularity and widespread application of LTR models in prioritizing information in a variety of domains makes their scrutability vital in today's landscape of fair and transparent learning systems. However, limited work exists that deals with interpreting the decisions of learning systems that output rankings. In this paper we propose a model agnostic local explanation method that seeks to identify a small subset of input features as explanation to a ranking decision. We introduce new notions of validity and completeness of explanations specifically for rankings, based on the presence or absence of selected features, as a way of measuring goodness. We devise a novel optimization problem to maximize validity directly and propose greedy algorithms as solutions. In extensive quantitative experiments we show that our approach outperforms other model agnostic explanation approaches across pointwise, pairwise and listwise LTR models in validity while not compromising on completeness.

1 Introduction

Interpreting the reasons behind predictions made by modern machine learning models is an important yet challenging problem. As a result, many approaches have been proposed to tackle the problem of interpretability for several model families – decision trees, sequential models, convolutional networks – and problem settings – text classification, image recognition, and regression tasks. However, limited work has been done for interpreting the ranking task and specifically learning to rank (LtR) models that are widely utilized in search and recommendation systems. Initial work on ranking models has been limited to the ad-hoc retrieval scenario where the complex ranking models under consideration focused purely on better text representation [20,7,19,21]. However, in many practical scenarios like web search, recommender systems, vertical search, etc. ranking is typically performed using LtR models operating on a large set of features of which text-based features are but a small subset.

Typical LtR models are trained using standard ML algorithms adjusted to a ranking objective. While explanation approaches exist for standard ML models like neural networks [15] and decision trees [1], they are *model introspective*. Such

explanation methods are inextricably tailored towards specific model families. Consequently *model agnostic* approaches, like SHAP [14] and LIME [17] have been popularised that treat the trained model as a functional black box. However, most of these approaches have been devised specifically for classification and regression problems and to the best of our knowledge, no approach exists for explaining ranked outputs of LTR models.

Rankings make for an inherently challenging scenario unlike classical classification and regression tasks in that rankings can be viewed as aggregations of multiple predictions – sorting regression scores (`PointWise` LTR), aggregating preferences (`PairWise` LTR) or choosing permutations (`ListWise` LTR). Furthermore, explanations for rankings should not only explain single item relevance but more importantly be able to accurately describe rank differences between items post aggregation.

In this paper we propose interpretability methods for LTR models that enable us to explain entire ranked outputs rather than explaining a single item score or a classification result. Secondly, we propose a model agnostic feature attribution/explanation method that can help us cross-examine various ranking algorithms coherently without having to worry about the training regimen (pointwise, pairwise, listwise [13]), access to training data and access to the model parameters. Such approaches are not only helpful when debugging for model developers but especially so for audits and legislative procedure where a balance needs to be struck between exposing business secrets and transparency.

Our Contributions We propose the problem setting of explaining LTR models as: given a trained ranking model and a ranking produced for a query, our aim is to select a *subset of features* as an explanation that is most responsible for the prediction.

Our first contribution is to define feature explanations of a given pre-trained LTR model in terms of their *validity* and *completeness*. *Validity* encodes the amount of predictive capacity contained in the explanatory features measured in terms of rank correlation with the original ranking. *Completeness* on the other hand measures the lack of information in the features that are not explanation features. Note that a feature subset might be valid but not complete. Consider an example where the two variables encode the predictive capacity for a ranking decision and are perfectly correlated. In such a scenario, selecting any one of these variables would result in high validity. However, if only of these features is chosen in the explanation set, the non-explanation set also retains enough predictive capacity. Validity and completeness measures have been proposed earlier [5] but we are the first in our knowledge to propose an algorithms to explicitly optimize it for LTR.

Explained in detail later.

As our second contribution, we pose the problem of explaining rankings in terms of a size-bounded feature subset that has maximum validity or rank correlation with the original ranking. We propose and empirically analyse greedy heuristics to find such valid explanation sets.

As a final contribution, we carry out extensive experimental evaluation where we show that our proposed approach is significantly better than SHAP (between

10% and 100%) for finding explanations that are valid yet complete for LtR models. Specifically, we devise 3 carefully considered greedy heuristics and empirically compare them against the popular Shap on a variety of LtR models and 2 commonly used LtR datasets – `MQ2008` and `MSLR` [16].

2 Related Work

A large family of methods for posthoc interpretability of ML models have been proposed in recent literature [10]. Broadly speaking, we classify the methods of posthoc interpretability into model introspective methods, where we have access to the model parameters, and model agnostic methods where we use the trained model as a functional blackbox. Model introspective methods typically attribute importance to input features by model specific optimizations like gradient flows [15], attention values [23] etc.

Model agnostic methods, popularized by [17,2,9], on the other hand are more general in that they are agnostic to how the underlying model was trained or how the scores are computed. Most model agnostic methods formulate simple proxy models attempting to learn a locally faithful approximation around the prediction, for example through linear models or sets of rules, representing sufficient conditions on the prediction [12] or by using linear models over local perturbations [17,9]. In this work, we operate in the **model agnostic** regime where we do not assume any access to the ranking model's parameters.

Beyond these approaches, there are game-theoretic approaches like Shap [14] that use model agnostic or introspective approaches as kernels while preserving important properties for explanation like accuracy, missingness and consistency. *Shapley Additive Explanations* computes feature importance or contribution values to a prediction (for a specific input) as compared to its expected value. In Section 3.4 we describe the adaptation and limitations of Shap for explaining ranking models. In our experiments, we extend Kernel Shap – a model agnostic variant to estimate shapley values [14], for ranking models and use it as a baseline. Note that Kernel Shap is similar to the popular Lime approach [17] as stated in [14].

Ranking models, like LtR, are different from the classical prediction tasks in that their output is an ordering of input items. This ordering can be obtained directly as model output or as an aggregation over multiple individual ranking decisions. Consequently, none of the existing work on posthoc interpretability can be directly applied to our setting. Model agnostic approaches for interpretability in rankings include [18] which constructed surrogate models to imitate decisions of black-box LtR models. Other works [19,21] adapt an existing local linear explanation model, Lime[17] to the problem of explaining textual document relevance with respect to a query. Both approaches deliver term based explanations for pointwise models and their applicability beyond text features is limited. We, on the other hand, focus on explaining LtR models with arbitrary features and can be applied additionally to pairwise and listwise approaches. Another line of work [20] proposes an approach for locally explaining the ranked list output of

pure text based neural rankers (that learn latent feature representations) via a set of intent terms. Our problem setting however is the more commonly prevailing setting in LTR where we have a known set of input features such as TF-IDF scores, page rank, etc. from which to choose an explanation.

3 Explaining Rankings

In this work, we focus on the ranking task in a document retrieval setting where given a query $\mathbf{q}$ the trained LTR model outputs a ranking of documents that is most relevant to $\mathbf{q}$. We are then interested in explaining the decision of the trained LTR model for a given output ranking with respect to the query $\mathbf{q}$.

LTR models are trained either in a `PointWise`, `PairWise`, or `ListWise` manner [13]. In a `PointWise` approach, the LTR problem is cast into a regression problem where the goal is to predict a ranking score for a query-document pair. The `PairWise` approach, on the other hand, trains a binary classifier model that classifies the document preference pair as concordant or discordant. In particular, the model outputs the likelihood or probability of the the input pair being a concordant pair. We utilize these output probabilities for the input ordered document pairs as scores in our approach. Finally, `ListWise` approaches directly optimize the value of ranking evaluation measures, like MAP, NDCG, averaged over all queries in the training data. Typical features used in LTR include the frequencies of the query terms in the document, the BM25 and PageRank scores. In our setting we are independent of the training regimen since we treat the LTR model as a blackbox.

3.1 Explanation Definition

Given a trained LTR model $\mathcal{R}$ and an input feature space $\mathcal{F}$, we are interested in explaining the output ranking $\mathcal{R}(\mathbf{q}, \mathcal{F}) \longrightarrow \pi$, for $\mathbf{q}$ in terms of a subset of features $\mathcal{F}' \subseteq \mathcal{F}$ that are most *impacting* for π. $\mathbf{q}$ here is shorthand for all retrieved documents for a given query.

We define two measures, namely *validity* and *completeness*, to quantify the importance of a feature subset (explanation) for π. We use Kendall's tau rank correlation coefficient to quantify the validity and the completeness of our explanations which is defined over a pair of rankings π, π' as

rank correlation ↝

$$\tau(\pi, \pi') = \frac{\text{no. of concordant pairs} - \text{no. of discordant pairs}}{\text{total no. of pairs}}.$$

Definition 1 (Validity). *An explanation $\mathcal{F}' \subset \mathcal{F}$ for π is said to be valid if $\mathcal{R}(\mathbf{q}, \mathcal{F}') \longrightarrow \pi' \approx \pi$, i.e., the model's output is predictable from the explanation alone. In our setting a valid explanation implies that the returned smaller set of features are sufficient to reconstruct the original ranking output by the model (when it used all the features). In particular we measure validity score ($\mathcal{V}$) of*

↳ create 2 rankers → 1 with $\mathcal{F}'$, another with $\mathcal{F}$ and compare τ

our explanation by computing the rank correlation between the ranking returned by the model when using the explanation and the original ranking.

$$\mathcal{V} = \tau(\mathcal{R}(\mathbf{q}, \mathcal{F}'), \mathcal{R}(\mathbf{q}, \mathcal{F})).$$

Definition 2 (Completeness). *An explanation $\mathcal{F}' \subset \mathcal{F}$ is said to be complete with respect to a trained model $\mathcal{R}$ and $\mathbf{q}$ if removing or altering the explanation features from the input will change the output function or ranking considerably. In other words, an explanation $\mathcal{F}'$ is complete if $\mathcal{R}(\mathbf{q}, \mathcal{F} \setminus \mathcal{F}')$ cannot approximate π. We define the completeness score of an explanation as the negative of the rank correlation between the original ranking and the ranking obtained by using features which are not included in the explanation, i.e.,*

$$\mathcal{C} = -\tau(\mathcal{R}(\mathbf{q}, \mathcal{F} \setminus \mathcal{F}'), \mathcal{R}(\mathbf{q}, \mathcal{F})).$$

Intuitively, a feature subset $\mathcal{F}'$ is *valid* if it contains enough information to re-create π when $\mathcal{F} \setminus \mathcal{F}'$ is not considered for ranking. Alternately $\mathcal{F}'$ is *complete* if $\mathcal{F} \setminus \mathcal{F}'$ does not contain enough information capacity to re-create the ranking π. Note that a valid feature subset $\mathcal{F}'$ might not always be complete. This case arises, for example, when two or more predictive features are correlated – only one of them suffices to be in the valid set but both are required for the complete set. In the following section we formulate the optimization problem for finding explanations and describe our approach.

3.2 Finding Fixed Size Valid Explanations

this make it complicated as you need to test ALL possible combinations

even solving for k can be infeasible

If number of variables to try is 100 and you want to select the best performing 10, you would have to compute equation 1, $\binom{100}{10} = 1.7e^{13}$ times...

Validity and completeness give us two ways of measuring the impact of an explanation for rankings. By defining them using τ, we have direct measures that we can optimize for. To find the ideal valid or complete feature attribution, one must exhaustively search all possible feature subsets $\mathcal{F}'$ of all possible sizes that maximize the respective scores. Trivially, when $\mathcal{F}' = \mathcal{F}$, then $\mathcal{F}'$ has maximum validity and completeness, i.e., the entire feature set $\mathcal{F}$ is a valid and complete explanation.

We emphasize that any human understandable explanation should be small in size. Therefore we restrict the size of explanations to a small constant $k << |\mathcal{F}|$. Mathematically, if we are interested in finding a *valid* explanation $\mathcal{F}'$ such that $|\mathcal{F}'| = k$, we solve the following optimization problem:

$$\arg\max_{\mathcal{F}'} \tau(\mathcal{R}(\mathbf{q}, \mathcal{F}'), \mathcal{R}(\mathbf{q}, \mathcal{F})) \quad s.t. \ |\mathcal{F}'| = k \tag{1}$$

Note that one can optimize for either validity or completeness to find an explanation but we choose to focus on validity in this paper. While the measures and optimization formulations may seem relatively similar there is a subtle point of contention. Valid explanations explain the construction of a ranked list whereas completeness in essence is a measure of sensitivity. Considering that we want a fixed size explanation, valid explanations inherently mean small succinct

sets whereas complete explanations imply larger feature sets since they must take all feature correlations into account.

Theoretical Properties and Limitations. Before we describe our approach, let us examine some theoretical constraints of the optimization objective. The naive approach to solve equation 1 is to compute τ for all possible k-sized subsets. This is an exact solution but is computationally infeasible when $n >> k$ where $n = |\mathcal{F}|$. In practice $n \geq 100$ and $k \leq 10$ usually which means that we have $\binom{n}{k} = \binom{100}{10} \approx 1.7e^{13}$ possibilities. Hence we must find a way to prune the $\mathcal{F}'$ search space while still trying to find the optimal solution. A commonly used technique in such a setting is greedy incremental search as shown in Algorithm 1. In case the optimization function is submodular or even weakly submodular [4], greedy selection gives us a theoretical bound on the error of our selection. The (weak) submodularity condition can be checked using the following definition from [4].

Definition 3. *(Submodularity Ratio) Let g be a nonnegative set function. The submodularity ratio of g with respect to a set U and a parameter $k \geq 1$ is*

$$\gamma_{U,k}(g) = \min_{L \subseteq U, S:|S| \leq k, S \cap L = \emptyset} \frac{\sum_{x \in S} g(L \cup \{x\}) - g(L)}{g(L \cup S) - g(L)},$$

where the ratio is considered to be equal to 1 when its numerator and denominator are both 0. Then g is submodular if and only if $\gamma_{U,k}(f) \geq 1$ for all U and k. If $\gamma = \min_{U,k} \gamma_{U,k}(f) \in (0, 1)$, the function is said to be weakly submodular.

In our case U is the set of all input features, L and S are non-intersecting subsets of U and the function g (for a particular ranking model) corresponds to the rank correlation between the original ranking and a new ranking obtained by adding new feature to an existing explanation set. Because of the black box nature of our ranking models, neither submodularity nor weak submodularity of the objective function can be guaranteed which hinders us from achieving any theoretical guarantees for the solution. Nevertheless, in practice, greedy solutions have been shown to provide convincing and better results (for example see [6]). In this work, we propose variants of a greedy approach to find valid explanations and empirically analyze the effectiveness of our proposed methods.

3.3 Our Solution Framework

In greedy incremental search, we begin with an empty explanation set and add features incrementally as long as the objective criteria is non-decreasing. Rather than optimizing directly for τ we optimize for a proxy utility measure based on the relevance score difference instead of rank difference between documents. The justification for which will become clearer in the remainder of this section. In the following we elaborate on our proxy measure and our greedy selection procedure.

Step 1: Preference Matrix. We first construct a preference matrix where the rows represent features or subsets of features from $\mathcal{F}$ and the columns are

concordant pairs of documents as determined by the original ranking π. For short ranked lists we use all pairs but for longer lists we sample uniformly at random from all concordant pairs as determined by π. Please note that as the top ranked documents appear in a larger number of concordant pairs, they will be more likely to appear in the sampled columns. Let P denote the set of sampled concordant pairs from π.

Step 2: Propensity calculation. Each row in our matrix is meant to denote the impact of adding f to $\mathcal{F}'$ for every $p \in P$. By observing which pairs are concordant when adding f we can directly optimize for τ. A simple binary indicator of concordance may at first glance seem to be ideal but in the greedy setting purely looking at ranks can be volatile. Adding a new feature to $\mathcal{F}'$ that does not increase pairwise concordance but changes the relevance scores is also important to consider. **Firstly** (and trivially), a positive score difference directly implies concordance. **Secondly**, by observing the magnitude of score difference we can start to estimate the likelihood of the pair staying concordant (or becoming concordant) as more features are added to $\mathcal{F}'$ thereby introducing a level of robustness lacking when only looking at rank differences incrementally when directly optimizing for τ.

Hence we compute each cell value z_p^f of the preference matrix as the likelihood of the pair of being concordant when feature f is added to the explanation set instead of a simple binary indicator. If $z_p^f > 0$, p is concordant. Formally, if $\mathcal{R}(\boldsymbol{x}_i, \mathcal{F}')$ denotes the score for document $\boldsymbol{x}_i$ when only features in $\mathcal{F}'$ were used to represent the input to an already trained LtR model while masking all features in $\mathcal{F} \setminus \mathcal{F}'$. Then we compute

$$z_p^f = (\mathcal{R}(\boldsymbol{x}_i, \mathcal{F}' \cup f) - \mathcal{R}(\boldsymbol{x}_j, \mathcal{F}' \cup f)) \cdot w_p \tag{2}$$

where w_p is a weighting factor for each pair (column) which we set to the difference in ranks of the documents represented by $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Intuitively, pairs with larger rank difference are more important to preserve than smaller ones in order to maximize τ and hence are given higher weights. Initially when $\mathcal{F}'$ is empty, z_p^f is impact of a single feature f on the pairwise concordance of $p \in P$.

Utility Computation in Greedy Incremental Search We utilize a greedy iterative procedure to select features. In the first iteration $\mathcal{F}'$ is empty and $\mathcal{F}' \cup f = f$. We compute the preference matrix as described above for all features $f \in F$ and all pairs $p \in P$. In order to select k features to populate $\mathcal{F}'$, we estimate the utility of adding a feature to $\mathcal{F}'$ by aggregating the likelihood values for every pair in P, i.e. $u_i(f, P) = \sum_{p \in P} z_p^f$, where $u_i(f, P)$ denotes the utility of adding feature f in the ith iteration. We select the feature f that has the maximum utility in this iteration to be added to $\mathcal{F}'$. Once the feature f is selected ($\mathcal{F}' = \mathcal{F}' \cup f$), we remove the row corresponding to f from the preference matrix. We then recompute the preference matrix with the new $\mathcal{F}'$. In step $i+i$, we once again estimate the utility of all features if added to $\mathcal{F}'$. As in traditional greedy approaches the stopping criteria is to halt when adding a new feature

does not increase utility. Simply put, if

$$\arg\max_{f \in \mathcal{F} \setminus \mathcal{F}'} u_i(f, P) > \arg\max_{f^* \in \mathcal{F} \setminus (\mathcal{F}' \cup f)} u_{i+1}(f^*, P)$$

then do not add f^*. This means that an explanation can also be smaller than k if a new feature does not improve utility. We denote this approach in our experiments as GREEDY . The pseudocode is given in Algorithm 1. We shortened $\mathcal{R}(\mathbf{q}, \mathcal{F})$ to $\mathcal{R}(\mathcal{F})$ since $\mathbf{q}$ is implicit in our setting.

Algorithm 1 GREEDY

Input: Trained LTR model $\mathcal{R}$, input feature set $\mathcal{F}$ corresponding to query-document pairs, size of explanation k
Output: Valid Explanation $\mathcal{F}'$ of size k

function GREEDYSEARCH($\mathcal{R}, \mathcal{F}$)
  Initialize $\mathcal{F}' = \emptyset$
  Initialize P with sampled concordant(w.r.t. original ranking) pairs
  for $(i = 0, 1, \ldots k-1)$ **do**
    Choose $f = \arg\max u_i(f, P)$
    Let f^* is the feature selected in the $i-1$th iteration
    if $u_i(f, P) > u_{i-1}(f^*, P)$ **then**
      $\mathcal{F}' \leftarrow \mathcal{F}' \cup f$

Algorithms greedy-cover and greedy-cover-ϵ . We note that in each iteration, the GREEDY algorithm does not always guarantee an increase in the number of concordant pairs when adding f to the explanation set. Especially when adding a correlated feature, utility might increase when the score difference increases for a pair which was already concordant in the previous step for instance. However, we know that increasing the number of concordant pairs directly leads to an increase in τ. To account for such cases, we devise an improved heuristic that instead of maximizing utility over all concordant pairs P at step i only maximizes utility for all pairs $\mathcal{P} \subseteq P$. $\mathcal{P}$ is the set of all pairs that were not already considered concordant by the explanation set at step $i-1$, i.e. we want to improve the concordance propensity of pairs previously considered to have low z_p^f. $P \setminus \mathcal{P}$ is then the set of pairs already "covered" by $\mathcal{F}'$ until step i.

Just like $\mathcal{F}'$, $\mathcal{P}$ is also updated incrementally. After selecting the feature f with maximum utility at step i, all pairs p for which $z_p^f < \epsilon$ are added to $\mathcal{P}$. ϵ is a threshold that allows us to control which pairs are considered still unaccounted for. When $\epsilon = 0$ we strictly consider only pairs that were still discordant in the previous iteration. Raising the value of the coverage factor ϵ lets us also retain pairs for which $z_p^f > 0$ in $i+1$. The intuition here is that a small z_p^f is a weak indicator of concordance.

Note that our approach is predicated on the assumption that adding a feature to $\mathcal{F}$ does not change the concordance of a previously covered pair. While this assumption can seem limiting, our experiments show promising results where

the improved strategy maximizes validity by focusing on different parts of the ranked list incrementally. Intuitively, the stopping criteria now becomes coverage oriented, i.e. when $\mathcal{P} = P$.

We call this greedy heuristic variation GREEDY-COVER when $\epsilon = 0$ and GREEDY-COVER-ϵ when $\epsilon > 0$ in our experiments. The pseudocode of our algorithm GREEDY-COVER-ϵ is provided in Algorithm 2.

Algorithm 2 GREEDY-COVER-ϵ

Input: Trained LTR model $\mathcal{R}$, input feature set $\mathcal{F}$ corresponding to query-document pairs of $\mathbf{q}$, size of explanation k, coverage factor ϵ
Output: Valid Explanation $\mathcal{F}'$ of size k

function GREEDY-COVER-ϵ $(\mathcal{R}, \mathcal{F})$
  Initialize $\mathcal{F}' = \emptyset$
  Initialize P with sampled concordant(w.r.t. original ranking) pairs
  for $(i = 0, 1, \ldots k-1)$ **do**
    if $P \neq \emptyset$ **then**
      Choose $f = \arg\ \max u_i(f, P)$
      Set $P = P \setminus \{p : z_p^f > \epsilon\}$
      $\mathcal{F}' \leftarrow \mathcal{F}' \cup f$

3.4 Comparison with SHAP

Kernel SHAP is the model agnostic version of SHAP [14] that we compare our approach against in the experiments. Kernel SHAP approximates the original model, $f(x)$ on input feature vector x (of size M) by a linear explanation model $g(x')$ on binarized input x' such that $g(x') = \phi_0 + \sum_{i=0}^{M} \phi_i x'_i$, where $x'_i \in \{0, 1\}$ depending on whether ith feature is used or not. ϕ_i corresponds to the importance/attribution of feature i for model prediction. This version of SHAP is equivalent to LIME [17] with the new kernel mentioned in [14] to better estimate the shapley feature attributions.

Recall that the LTR model takes as input a query-document instance and outputs a relevance score for the document. That means we can use SHAP for a given query-document input to obtain the corresponding feature importance coefficients for the predictive score of a particular document. One can in principle add the corresponding feature coefficients by running SHAP for all documents. An explanation set is then computed using the top k features with highest importance coefficients. In our experiments, we observed that it is best to use the importance values for the top ranked document.

Limitations of SHAP. As opposed to our approach, SHAP does not directly output an explanation set. Instead it must be inferred from the feature attributions towards the predicted score of each document. **First**, there is no principled way in which these attributions should be aggregated for explaining scores of a

list of documents at the same time. **Second**, SHAP has a strong independence assumption between features while computing attributions and disregards the feature dependence structure. Our solution on the other hand actively considers dependence between features in the selection of feature sets. So while SHAP might be accurate in determining feature-wise importance, it fails to accurately determine the importance of sets of features.

4 Experimental Setup

Datasets. We chose two benchmark learning-to-rank datasets from LETOR4.0 [16] to evaluate our approaches. `MQ2008` consists of 800 queries with labeled query-document feature vectors. There are 46 features ranging from text based features like BM25 and network based features like PageRank. `MSLR` is a bigger dataset consisting of 10k queries and 136 features per query-document vector. For all experiments we use fold1 of the train-test split out of the 5 provided.

Ranking models. We selected one model from each of the three main learning-to-rank categories to explain – `PointWise`, `PairWise` and `ListWise` models. We chose the best performing ranking models in terms of their ranking performance (@NDCG scores) for each of the datasets. However, the ranking performance of our trained models has no bearing on our explanation method.

We chose LINEAR REGRESSION as the `PointWise` and LAMBDAMART [22] as the `ListWise` for both datasets. Note that LAMBDAMART is a type of GBDT (Gradient Boosted Decision Tree) that is optimized for NDCG and is a popular choice in most commercial search engines. For `MQ2008`, we chose RANKNET [3] as `PairWise`. However for `MSLR` we used RANKBOOST [8] instead of RANKNET since the ranking performance in NDCG was significantly better (0.33 vs 0.20). All models were trained using the RankLib [3] implementation of the aforementioned algorithms with default parameter settings.

	MQ2008	MSLR
PointWise	0.471	0.229
PairWise	0.468	0.336
ListWise	0.655	0.468

Table 1: **Ranking performance of trained models (NDCG@10).**

Baseline and competitors. We use a random selection of k features as the baseline for all our experiments called RANDOM. We use SHAP (see Section 3.4) as our baseline in two different variations. The approach named SHAP-1 uses SHAP's output explanation corresponding to the score of the top ranked document. We also devised an approach called SHAP-5 that aggregates the feature attribution from SHAP for the top 5 ranked documents. The aggregation function we chose is a simple sum of the feature attribution values. To obtain an explanation of size k, we select the top-k features as per their aggregated attribution values. Kernel SHAP has two hyperparameters

[3] https://sourceforge.net/p/lemur/wiki/RankLib/

`n-samples` and `background-size` that we set to 200 and 500 respectively after tuning.

Our approach variants. In the previous section we presented 3 variations of our approach – GREEDY , GREEDY-COVER and GREEDY-COVER-ϵ . For GREEDY-COVER-ϵ we set the threshold to be the mean of the concordant pair propensity scores for a given feature (i.e. mean of the values in the corresponding row of the preference matrix). Formally, for a selected f

$$\epsilon_f = \frac{\sum_{z \in Z} z}{|Z|}$$

where $Z = \{z : z_p^f > 0 \text{ for all } p \in \mathcal{P}\}$. For all greedy approaches we set the number of pairs sampled from π to 50 for `MQ2008` and 100 for `MSLR` since `MQ2008` has smaller ranked lists.

Seed selection. Greedy approaches tend to be sensitive to the first item selected. In order to account for suboptimal first feature selection, we choose the top 3 candidate/seed features in the first iteration and switch to choosing the best from the second iteration there on. Intuitively this can be seen as running our greedy procedure 3 times with a different first feature added to the solution set in each run. This simple strategy greatly improved the selections as compared to starting from a single choice.

unless you sample all variables

basically, running it 3 times with different seeds

Metrics. We measure the goodness of an approach in terms of validity and completeness as described in Section 3. For `MQ2008` we take all 156 queries in the test set whereas for `MSLR` we sample 500 queries from the test set. Validity and completeness is computed for each query's explanation and then averaged.

5 Results

With our experiments we answer the following research questions:

- **RQ I:** How effective is our approach in choosing valid explanations compared to the baselines? Section 5.1)
- **RQ II:** To what degree are the valid explanations also *complete* ?(Section 5.2)
- **RQ III:** How does the obtained valid explanations compare with the optimal explanation? (Section 5.4)

5.1 Validity of Explanations

In Table 2 we compare the validity of the baseline approaches to our method for $k = 5$. In the `MQ2008` dataset, we observe that both our approach and the SHAP-based feature attribution baselines significantly outperform the RANDOM baseline. Secondly, there is a no significant difference between SHAP-1 and SHAP-5

	Validity ↑			Completeness ↑		
	PointWise	PairWise	ListWise	PointWise	PairWise	ListWise
			MQ2008			
Random	0.062	0.237	0.041	-0.155	-0.718	-0.305
SHAP-1	0.131	0.381	0.124	0.039	-0.517	-0.068
SHAP-5	0.133	0.269	0.135	0.041	-0.391	**-0.031**
GREEDY	0.202	0.287	0.259	0.041	**-0.369**	-0.091
GREEDY-COVER	0.197	0.268	0.280	**0.072**	-0.371	-0.096
GREEDY-COVER-ϵ	**0.299**	**0.616**	**0.361**	-0.013	-0.623	-0.162
			MSLR			
Random	0.002	0.009	0.000	-0.172	-0.634	-0.303
SHAP-1	0.006	0.050	0.023	-0.007	-0.071	-0.052
SHAP-5	0.009	0.048	0.001	0.000	-0.067	-0.051
GREEDY	0.007	0.060	0.045	0.001	**-0.057**	**-0.034**
GREEDY-COVER	**0.071**	0.094	**0.081**	-0.025	-0.130	-0.070
GREEDY-COVER-ϵ	0.059	**0.110**	0.074	**0.020**	-0.074	-0.041

Table 2: Results for the MQ2008,MSLR when k=5. All greedy approaches are statistically significantly better than the baselines for validity measures. Results in bold are the best for each model and significantly better ($p \leq 0.05$) that the closest competitor.

for PairWise and ListWise models. This suggests that aggregating over multiple documents does not improve validity for feature attribution methods like SHAP.

Greedy vs Shap. Our simple GREEDY approach is already significantly better than both SHAP-1 and SHAP-5. Furthermore, GREEDY-COVER-ϵ is almost 2× as good as SHAP-1 for all models in MQ2008. The MSLR dataset has nearly four times as many features as MQ2008 which translates to lower values overall for the same explanation size although a Kendall's τ value above 0 already indicates that at least 50% of item pairs are concordant. Nevertheless, we find similar trends. All our approaches significantly outperform SHAP with GREEDY-COVER-ϵ often being 2× better than SHAP-1. This shows that directly optimizing for our validity objective is more desirable than choosing features using feature attribution techniques like SHAP. Note that all greedy variants that we propose significantly outperform SHAP.

Effect of Pair Coverage. For MSLR, GREEDY-COVER consistently outperforms GREEDY whereas in MQ2008, it is better than GREEDY for ListWise while remaining competitive in the rest. This shows that augmenting a simple greedy heuristic with a coverage criteria generally leads to good results. We incorporated the notion of coverage in our approach to account for the redundancy of corre-

lated features in the explanation. If we choose to cover pairs based on a carefully selected threshold like in GREEDY-COVER-ϵ we can yield significant gains. By essentially ignoring pairs (removed from the preference matrix iteratively) only where the score difference is positive and large, our method can start to select features that are important for other parts of the ranking π. Redundant features that contribute to the concordance of more "confident" covered pairs will now have less utility. In `MQ2008`, for the `PairWise` model in particular, using only 5 features we are able to produce a π' that is very strongly correlated (0.616) with π. In our experiments we chose a simple heuristic to find the the threshold. This heuristic is not always the right choice (`PointWise` for `MSLR`) however. For improved performance, one can also manually tune the threshold parameter.

ok, but how do we explain the effect of variables in the model?

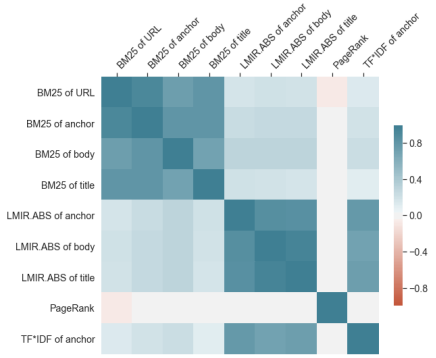


Fig. 1: Feature Correlation of `MQ2008`

	SHAP-1	GREEDY-COVER-ϵ
	BM25 Body	BM25 Body
	BM25 Anchor	TF-IDF Anchor
	LM Anchor	BM25 Title
	LM Title	LM Title
	BM25 URL	Page Rank
$\mathcal{V}$	0.27	1.00

Table 3: Explanations for the query *18328* from `MQ2008` and the `PairWise` model

Anecdotal example from results. For query *18328* (from `MQ2008`) and the `PairWise` model, the 5 feature explanation of SHAP-1 and GREEDY-COVER-ϵ is shown in Table 3. The validity score of GREEDY-COVER-ϵ for this query is 1 while SHAP-1 is 0.27. While the features SHAP-1 selects are good for predicting the relevance of the top ranked document, it is clear that these features alone do not influence the whole ranking as earlier pointed out in Section 3.4. Our approach on the other hand is able to explain the entire ranked list by selecting non-redundant holistically (for the list) important features (e.g. PageRank, see Figure 1).

5.2 Completeness

On the utility of completeness. We now compare the effectiveness of different explanation methods with respect to completeness that measures the predictive capacity of the features not present in the selected explanation. Firstly, we note

that while a small number of features may prove to be sufficient for good valid explanations, the same does not hold for completeness. To ensure high completeness all the correlated predictive features need to be selected into the explanation set. However, longer explanations have negative implications on human decision making as previous works have already established [11]. Neverthless it is indeed important to measure completeness of explanations as a complementary measure to validity in measuring explanation performance.

Greedy vs rest. From Table 2, we see that all approaches are considerably better than a random baseline. That is, removing a small random set of features has relatively low impact on the rank order of π' (especially for the pairwise model where completeness is -0.634). Removing 5 important features attributed by SHAP results in significant gains (-0.071 as compared to -0.634). Our greedy approaches are better than SHAP in nearly all cases (except `ListWise` for `MQ2008`). Although, we do not directly optimize for completeness the GREEDY approach is able to uncover more complete explanations especially for the `PairWise` model. For `ListWise` models there is however no clear consensus and we observe that the dataset influences whether SHAP or our approach is a better choice.

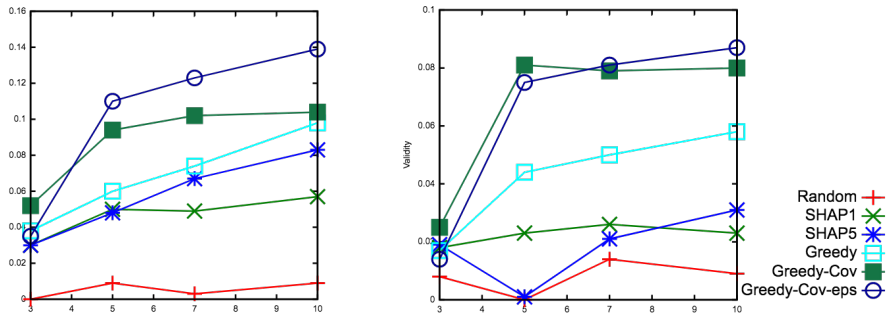
Why is greedy good? Out of our approaches, GREEDY outputs more complete explanations except for the `PointWise` model in `MQ2008` where it is second best overall. GREEDY does not take redundancy of feature contributions into account since coverage of pairs in not modelled here. Hence it is more likely to select correlated features as compared to GREEDY-COVER or GREEDY-COVER-ϵ reducing its informativeness and validity.

Anecdotal example. GREEDY selects the following features as an explanation for query 823 (from `MSLR`) and the `PairWise` ranker – (i) BM25 score of the body, (ii) LM score of the body, (iii) covered query term ratio of the title, (iv) sum of stream length normalized term frequency for the URL and (v) covered query term number for the title whereas GREEDY-COVER-ϵ selects (a) LM of the whole doc, (b) sum of stream length normalized term frequency for the URL and (c) covered query term number for the body. The LM and BM25 score of the body are highly correlated but only the LM score is selected by GREEDY-COVER-ϵ . This results in a completeness score of -0.1 for GREEDY and -0.6 for GREEDY-COVER-ϵ .

A key takeaway from measuring completeness is that maximizing validity does not always ensure minimizing completeness. This points to the presence of small feature subsets that are both valid and complete. Finding an explanation that is both valid and complete is a challenging problem due to the dual nature of the objectives (one minimizes rank correlation while the other maximizes it) and we leave this to future work.

5.3 Comparison to optimal explanations

Now we compare our explanation sets to the one with highest validity score obtained by brute force search over all possible feature subsets of size $k = 3$. We



(a) Pairwise Ranker (b) Listwise Ranker

Fig. 2: Effect of k. The graphs show the validity of increasing explanation sizes $(3, 5, 7, 10)$ for MSLR.

find that for the `PairWise` model for MQ2008 the validity of the ideal feature set is 0.860 on average. Our best approach produces an explanation $\mathcal{F}'$ (k=5) that achieves an average validity of 0.616 which shows that even if we make sub-optimal choices in the beginning we are able to exploit pairwise preferences effectively to eventually construct an explanation that is closer to the ideal than all the other approaches. For `PointWise` and `ListWise` models, the ideal validity is 0.80 and 0.76 respectively. Here we see a larger scope for improvement. For MSLR, the ideal validity for all models was approximately 0.3 which once again shows that we are not too far from the ideal explanation especially for pairwise models (0.110 for GREEDY-COVER-ϵ).

5.4 Effect of k

In section 3, we outlined that our optimization problem was not exactly submodular. Inspite of this, our greedy approaches are still able to find increasingly valid explanations as the size of the explanation is increased. Figure 2 illustrates this finding for 2 models for MSLR. We found similar trends for the other dataset and models. An interesting finding here is that increasing k benefits GREEDY-COVER-ϵ more than GREEDY-COVER . GREEDY-COVER is more likely to be affected by smaller ranked lists (fewer concordant pairs sampled) since pairs will be covered at a faster rate. GREEDY-COVER-ϵ on the other hand is able to iteratively cover only pairs where it is most confident. By having more pairs in each iteration, the utility of the feature added to the explanation can be better estimated.

6 Conclusion and Future Work

In this paper we introduce the novel problem of finding explanations for LTR models in a local posthoc manner. We defined notions of validity and completeness specifically for rankings. We proposed a flexible framework for valid

explanations that effectively explores the search space by optimizing for pairwise preference (extracted from the target ranked list) coverage. In our experiments on several model families and datasets we show that our approach is significantly better for LTR models in terms of validity. By comparing against the ideal feature attribution we see that our approach is the first step in this novel problem domain. With the promise of GREEDY-COVER-ϵ , we envisage a more adaptive and optimization specific threshold leading to improved performance. Finally, we will explore how our approach can be adapted in a semi-blackbox setting where we have access to the learning algorithm but not the model parameters.

References

1. Alvarez, I.: Explaining the result of a decision tree to the end-user. In: ECAI (2004)
2. Alvarez-Melis, D., Jaakkola, T.: A causal framework for explaining the predictions of black-box sequence-to-sequence models. In: EMNLP. ACL (2017)
3. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: ICML (2005)
4. Das, A., Kempe, D.: Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In: ICML (2011)
5. DeYoung, J., Jain, S., Rajani, N.F., Lehman, E., Xiong, C., Socher, R., Wallace, B.C.: Eraser: A benchmark to evaluate rationalized nlp models. arXiv preprint arXiv:1911.03429 (2019)
6. Elenberg, E.R., Dimakis, A.G., Feldman, M., Karbasi, A.: Streaming weak submodularity: Interpreting neural networks on the fly. In: NeurIPS (2017)
7. Fernando, Z.T., Singh, J., Anand, A.: A study on the interpretability of neural retrieval models using deepshap. In: ACM SIGIR (2019)
8. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of machine learning research (2003)
9. Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., Turini, F.: Factual and counterfactual explanations for black box decision making. IEEE Intelligent Systems (2019)
10. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM CSUR (2018)
11. Lage, I., Chen, E., He, J., Narayanan, M., Kim, B., Gershman, S.J., Doshi-Velez, F.: Human evaluation of models built for interpretability. In: AAAI (2019)
12. Letham, B., Rudin, C., McCormick, T.H., Madigan, D., et al.: Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics (2015)
13. Liu, T.Y., et al.: Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval (2009)
14. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in NeurIPS (2017)
15. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition (2017)
16. Qin, T., Liu, T.: Introducing LETOR 4.0 datasets. CoRR (2013)
17. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: ACM SIGKDD (2016)

18. Singh, J., Anand, A.: Posthoc interpretability of learning to rank models using secondary training data. arXiv preprint arXiv:1806.11330 (2018)
19. Singh, J., Anand, A.: Exs: Explainable search using local model agnostic interpretability. In: ACM WSDM (2019)
20. Singh, J., Anand, A.: Model agnostic interpretability of rankers via intent modelling. In: Conference on Fairness, Accountability, and Transparency (2020)
21. Verma, M., Ganguly, D.: Lirme: Locally interpretable ranking model explanation. In: ACM SIGIR (2019)
22. Wu, Q., Burges, C.J., Svore, K.M., Gao, J.: Adapting boosting for information retrieval measures. Information Retrieval (2010)
23. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: ICML (2015)