Data Mining Techniques 2 - Group 139

Tianhao Xu[1,2[2740825]], Ruijia Lei[1,2[2733510]], and Peter Pan[1,2[2717917]]

Vrije Universiteit[1] and Universiteit Van Amsterdam[2]

1 Introduction

The objective of this assignment is to explore a dataset provided by Expedia—an online travel shopping company, which contains 5 million search hotel queries by real users. We are expected to leverage all possible data mining techniques to train a model that can be adapted by the search engine to predict users' choices and rank the hotels based on the user's history searching records and hotel information. Finally, to see the performance of our model, we will upload our prediction results to Kaggle to compete with other groups.

2 Related work

Learning to rank(L2R) [9] is the core method of recommendation, search and advertising. The ranking result largely affects user experience, advertising revenue, etc. It can be well used to predict the search results of hotels. L2R is a supervised machine learning process that extracts features for each given query-to-document pair, and obtains real data annotations by log mining or manual annotation methods. L2R algorithms consist of three main categories: PointWise Approach, PairWise Approach, and ListWise Approach.

PointWise approach deals with a single document. After converting the document into a feature vector, the machine learning system scores the document based on the classification or regression function learned from the training data, and the scoring result is the search result. Its method is well understood, that is, using traditional machine learning methods to learn the relevance of documents under a given query, for example, LR [13], FM [5], DeepFM [6], DIN [15]and other algorithms can be learned using PointWise method, but sometimes the order of sorting is important, and PointWise method learns the global relevance, and does not penalize the merit of the order.

For search systems, the system receives a user query and returns a list of relevant documents, so the key to the problem is to determine the sequential relationship between documents. The single-document approach calculates the classification score of a single document without considering the order relationship between documents. The document pair approach transforms the sorting problem into a multiple pair sorting problem by comparing samples two-by-two, constructing biased-order document pairs, and learning to sort from the comparison. It compares the order of different documents. Commonly used methods are SVM Rank [4], RankNet [1], RankBoost [10].

The single-document method treats each document in the training set as a training instance, the document-pair method treats any two document pairs in the search results of the same query as a training instance, and the document-list method differs from both of these methods in that the ListWise method directly considers the overall sequence and optimizes it for the Ranking evaluation metrics. Commonly used ListWise methods are LambdaRank [2], AdaRank [14], SoftRank [11], LambdaMART [1].

Liu et al. [7] discuss and evaluate various ranking models, including random forest, gradient boosting machine, lambdaMART, factorization machine, logistic regression and extreme randomized trees. They consider price_usd, prop_starrating and prop_location_score2 to be the three most important list ranking functions.

Liu et al. used the evaluation metric that Normalized Discounted Cumulative Gain (NDCG) [12] to evaluate the model. In search and recommendation tasks, the system often returns a list of items. NDCG is used to evaluate the sorting results. It adds up all the results and the higher the overall quality of the list the greater the NDCG value. Because the search results are different for various queries, there is no way to compare the DCG values of queries. So NDGC is expressed using DCG/IDCG, in which case NDCG is a relative value, and then different queries can be evaluated against each other by NDCG values.

Model The highest NDCG score among the different models is the Gradient Boosting Machine (GBM) with a score of 0.52477. It is a machine learning technique for regression problems that generates predictive models in the form of an ensemble of weak predictive models. It constructs models in stages and generalises them by allowing the optimisation of arbitrary differentiable loss functions. Gradient boosting methods can also be used for classification problems by reducing them to regression using a suitable loss function. Liu et al. use GBM in R language with the balanced data.

Mavalankar et al. [8] investigate a similar problem that provide personalized hotel recommendations for Expedia. They found that price and rating were useful along with location features, which was consistent with the findings of Liu et al. For ranking, inspired by them, Mavalankar et al. studied ranking models of Random Forest, SGD Classifier, and Naive Bayes.

3 Exploring Data

The dataset consists of training and testing sets, we only explore the training set because it has extra 3 important features—`click_bool, gross_bookings_usd and booking_bool`. The dataset has 4,958,347 instances and 53 features, each instance stands for an user's search query. Due to the large number of features, we concludes them into 5 classes—User information, Hotel information, Competitor information, Search query property and Evaluation property.

We first counted the dataset's pivotal values, there are 199,795 unique search queries, and 129,113 hotels, but the click-through rate(CTR) is only 4%. We then surprised noticed that the missing values widespread in many features, the results is shown in Figure 2. From the figure, we can observe that the the features

of competitor class contribute more than 90% missing values, followed by user information class. To get the insight characteristics from search queries, we grouped the data by `srch_id` and get the plots Figure 3 of features srch_length_of_stay, srch_room_count, srch_adults_count, srch_children_count, visitor_hist_starrating and visitor_hist_adr_usd, we observe that most users prefer stay 1 to 2 nights, and most users will take a adult partner without child and book 1 room, and most users prefer to search 3 to 5 starring hotels, and the budget for hotel booking falls within the range of 100 to 250 dollars.

For more efficient and deeper observation, we randomly chose 10,000 unique users from the dataset, which coincident with 248,977 search queries. We then filtered the instances with `click_bool` equals to 1 and plot related countplots to get an overview of chosen hotels Figure 3, we notice that most booked hotels have 4 to 4.5 review starring, which average are 3 or 4 stars hotel. Finally, we adopted gradient boosting regressor to discover the most important features that could impact user's final decision. Before that, we created a target feature 'rating', which is defined as *rating = clicked_bool + 3 * booked_bool*, it has 3 results, 0, 1 and 4 (3 is impossible due to the dependency that user must clicks before booking hotels), which stands for none, clicked and booked. From Figure 3, we observed that `prop_location_score2, prop_location_score1, price_usd` have the greatest importance than the others.

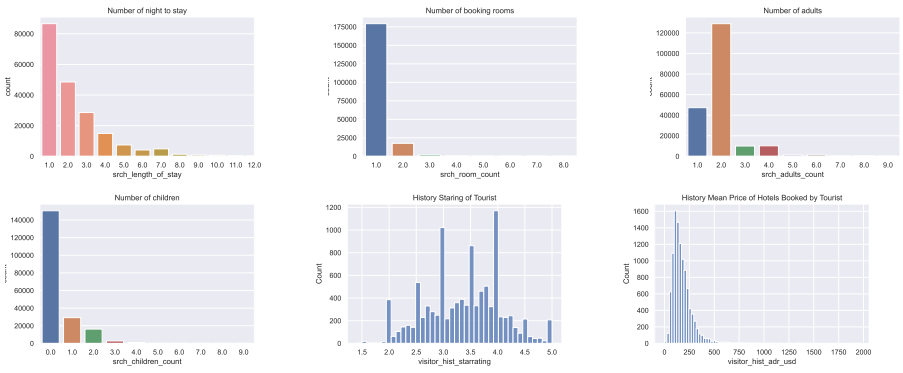

Fig. 1. Distribution of Features of Search Query and Hotel. Up Left: Distribution of Number of Night to Stay, Up Middle: Distribution of Number of Booking Rooms, Up Right: Distribution of Number of Adults, Down Left: Distribution of Number of Children, Down Middle: Distribution of History Starring by Visitor, Down Right: Distribution of History Mean Price of Hotels Booked By Visitors

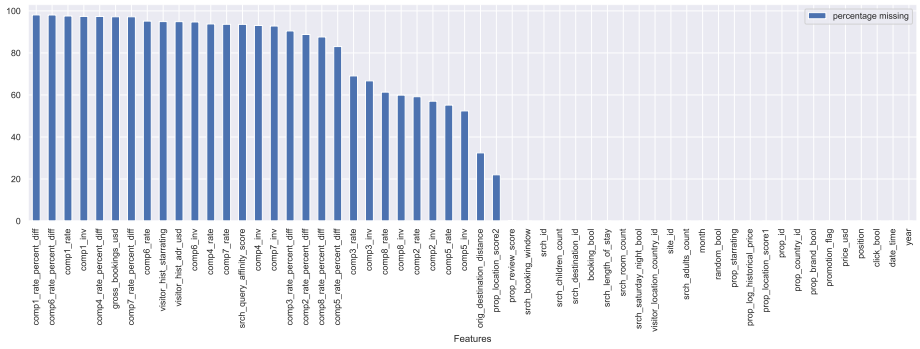

Fig. 2. Missing Values of 53 Features

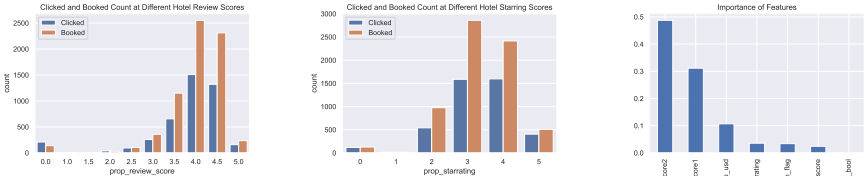

Fig. 3. Distribution of Characteristics of Search Query with click_bool = 1. Left: Distribution and Comparison of Clicked and Booked Count at Different Hotel Review Scores, Middle: Distribution and Comparison of Clicked and Booked Count at Different Hotel Starring Scores, Right: Importance of Features

4 Pre-processing

As far as we are concerned, feature selection is the process of automatically or manually selecting those features that contribute the most to the predictor variable or output of interest. The inclusion of irrelevant features in the data can reduce the accuracy of the model and cause the model to learn based on irrelevant features. Macroscopically, this dataset includes more than 50 features, and how to filter and utilize them is a very critical step in data preprocessing.

First, after roughly processing the data and looking at the distribution, the first thing that emerged was that many of the features contained missing values, so we looked at Figure 2, which reflects the corresponding percentage of missing values for each feature. We find that 31 features contain missing values, the most is as high as 98%. Based on the fact that the algorithm and model have not been clearly determined, we could have taken those approachs into consideration:

1) Deleting the missing values directly and abandoning those features. (For this project, the features which contain over 70% missing value are dropped.)

2) Replacing null value with the average value. However this approach is less controllable, it could have a more significant impact(mostly negative) on the subsequent processing of the values and was therefore abandoned by us.

3) Keeping the null values or the data of the numeric type if they can all be filled with 0.

4) Replacing them with negative value.

5) Replacing the missing by the regression result by the regression method using the sampled data.

As for feature engineering, the idea is to rank and find the potential connections of the features. Since we have removed some mass missing value carriers, the left we can conduct is a normalization process. We focus on `orig_destination_distance` and tranfsfer them to standard normal distribution by following the formula 1, which limits the range to between 0 and 1.

$$\mathrm{X}norm = \frac{\mathrm{X} - \mathrm{X}mean}{\mathrm{StandardDeviation}} \tag{1}$$

By comparison, we observe that price_usd is more accurate than gross_booking_usd, however, there are many outliers. We considered crawling the current Expedia hotel prices to generate data for mapping to compare outliers, then filtering by price difference, and those with too large a price difference we would either discard or replace with existing prices. However, we encountered a lot of problems with this, so we abandon this idea. Therefore, we also adopted the Z-score normalization for price_usd. Next, we created the prediction column called the "target" and gave weight 1 or 5 when user clicked or booked respectively. Meanwhile, the duplicates in click_bool, booking_bool and random_bool will be dropped because it is not exploitable in our method. Less redundant data means less opportunity to make decisions based on noise which will reduce overfitting. We also checked the distribution of different features to make a further decision.

Next, after removing null values, we check the correlation between features. To summarize the results in general, (we consider correlation as to be strong when its value is bigger than 0.7) there is positive correlation between location and country, and the results also confirm that we are accurate to do normalization on price_usd because it is autocorrelated. We also find that the correlation between click_bool and booking_bool is pretty evident.

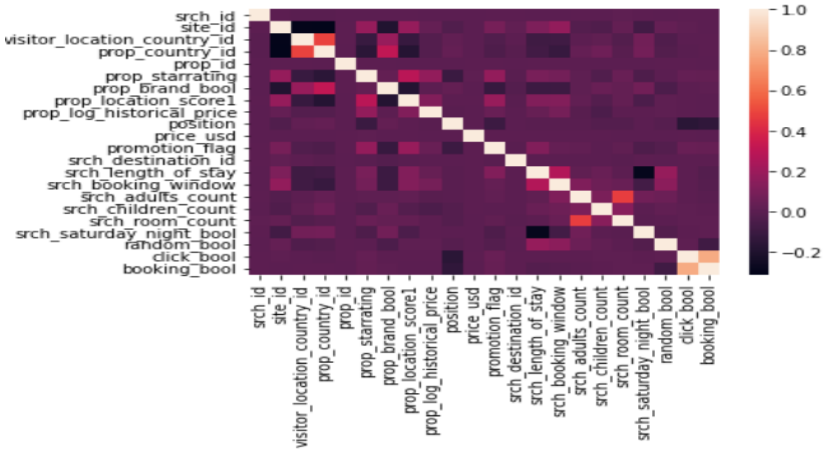

Fig. 4. Correlation Matrix of Features

To figure out the importance of different features(as we have already reduced features from 53 to 22), it is wise to conduct a selection process. We adopt several methods like the random forest model to check the importance of features and select the top 20 features.

5 Modeling

For the dataset, we can see that there are features biased towards user behavior, features of the product (here is the hotel) itself attributes, contextual features(such as the location and Saturday night) and some cross features. We try to consider logistic regression model, but it is not suitable in the case of large data volume, simple linear model, lacking the ability to learn high-order features, it is difficult to learn important features from training samples that never or rarely appear, the model itself only considers the first-order features, so it needs to do the high-order feature crossover manually. Linear regression methods can be unstable in the case of feature redundancy (i.e., the presence of multicollinearity). We then considered random forest models, as well as building GBDT models based on XGBoost or LightGBM and DeepFM models. Such nonlinear models are used to fit the data to gain better results.

Random Forest Since the random forest is the most popular general-purpose model, which is an ensemble model, we choose it as a baseline to compare with more advanced listwise L2R methods. Compared to the decision tree, the random forest is composed of multiple relatively uncorrelated models that help to outperform individual decision tree, the reason behind this is that a large number of trees eliminate individual errors and avoid over-fitting, which in turn improves the model accuracy. However, due to the essence of random forest is pointwise approach, which only takes a single doc in loss function at a time, and

the final ranking list is based on sorted docs scores, so it's inevitable to come across inversions in ranking.

DeepFM We firstly are also interested in DeepFM[6], a model contains deep learning and it is jointly proposed by Huawei and HIT, which improves on Google's Wide&Deep method to achieve improved results.

It is a combination of the FM and DNN model, by Factorization Machine using low-order feature crosses and DNN model using high-order feature crosses. However, most of the crossover information in the data is implicit. FM will consider a two-by-two crossover between all features, which is equivalent to artificially doing crossover for left and right features. But since the combination of n feature crossings is of the order of n^2, FM designs a new scheme to train a vector V_i for each feature i. When two features i and j cross, the weights of the two features after crossings are calculated by $V_i * V_j$. This greatly reduces the complexity of the computation. It includes the Addition unit and the inner product unit, which reflect the first-order and second-order combined features, respectively, eliminating a lot of manual feature engineering work.

Moreover, the feature vector of embedding is shared between the Wide side and Deep side, and the Deep part will be followed by the higher-order feature combination of features using a fully connected network. After obtaining the outputs of the FM and Deep units separately, the final predicted values are obtained by concatenating the outputs of the two parts, passing through a fully connected layer and then an activation function.

XGBoost [3] is a machine learning library focusing on gradient boosting algorithm born in February 2014, which has gained wide attention for its excellent learning effect and efficient training speed. Compared to the implementation of the gradient boosting algorithm in another popular machine learning library, scikit-learn, XGBoost often has a tenfold or more performance improvement.

The similarity between XGBoost and Random Forest is that they both use a cluster of trees as a model. The main difference between them is the way they are trained. Random forest handles all trees uniformly, while boosted trees use different levels of learners to classify progressively more difficult samples. [8]

We try to split a leaf into two leaves to calculate the splitting gain for each feature, the score it obtains is given by the equation 2:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma \tag{2}$$

LightGBM Similar to XGBoost, LightGBM is still an improved implementation of the GBDT algorithm framework, a fast, distributed, high-performance GBDT framework based on the decision tree algorithm. The main pain point is to improve the efficiency and scalability of the GBDT framework algorithm in the face of high-dimensional big data.

The "Light" is mainly reflected in three aspects, i.e. fewer samples, fewer features and less memory, through Gradient-based One-Side Sampling, Exclusive Feature Bundling, and Histogram. In addition, LightGBM has been engineered to support both feature and data parallelism, and to reduce the amount of communication by optimising the respective parallel methods.

6 Evaluation

6.1 Evaluation Metric

Based on the requirement our group used NDCG@5 as an indicator for cross-validation.

NDCG is the Normalised Discounted Cumulative Gain. It is a common metric for ranking and recommendation assessment. In this case, NDCG@n indicates that only the first n rankings are of interest to be correct and the later ones are not considered. So we are only concerned with the first 5 rankings.

The calculation of the NDCG is shown below:

1. Calculate DCG with the formula 3:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2 i} \tag{3}$$

   where rel is a rank score for this sorted list, result i. i is the current position number of result i. In search engines, the rel rank score, is a manual sampling of data, and according to certain rules to type out the rank score.
2. Calculate the IDCG (Ideal DCG). It is also calculated in the same way as step 1, except that the sequencing sequence is not derived by the algorithm, but the best sequence is ranked by hand for the sequence according to certain evaluation criteria.
3. Based on the results from the previous 2 steps, calculate the NDCG with the formula 4:

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{4}$$

6.2 Model

Random Forest We used top 20 features in the training set to keep as much information as possible, and used the combination feature 'rating' as the target feature for the testing set. The model hyperparameters are shown in Table 1, which is gained from grid search for the best parameters, and we set n_jobs=2 since the model is trained on a dual-core CPU platform. We then adopted a 10-fold cross-validation and the mean accuracy score is 0.862.

DeepFM We used the original top 22 features with 2 or more self-established crossover features such as prop_country_id and price_usd (there are more outliers because some prices include the whole trip), promotion_flag with visitor_hist_starrating. We set embedding_size(embedding size of features) to be 256, and deep_layers(full connected layer of Deep part) = [512, 256, 128].

Unfortunately, it takes a lot of time to test the performance with different feature crosses, but the results of the attempt were bad. Moreover, this model will take up overmuch computing resources, so the time does not allow us to fine-tune the parameters and conduct multiple feature crossovers cyclically. Thus, we finally gave up this method.

XGBoost We used the top 20 features for training and used grid search to find the best parameters and tuned the parameters with cross-validation to find some important parameters: n_estimators = 3000 (number of gradient boosting trees built), learning_rate = 0.15 (learning rate), max_depth = 3 (maximum tree depth, to to control overfitting). With these parameters, we obtain a mean accuracy score of 0.905.

LightGBM It can train large datasets much faster and is more accurate than other boosting models. We also selected the top 20 features, used a grid search to find the parameters, and tuned them by cross-validation.

To ensure peak accuracy while avoiding overfitting, we adjusted several important parameters: max_depth = 6, learning_rate = 0.1, num_leaves = 31 (number of leaves to be formed in a tree), boosting = dart.

Setting num_leaves to a value higher than 30 ensures the accuracy of the model. It should also be less than 2^max_depth. We also tried three gradient enhancement methods, GBDT, DART and GOSS, and we found that DART worked best. It is a method that uses dropout to improve model regularisation and to deal with some other less obvious problems. The mean accuracy score of LightGBM is 0.926.

Final model Due to the essence of Random forest model is point-wise approach that only deals with 1 document. From the Table 2, we can observed that random forest gets worse ranking performance than the other pair-wise and list-wise approaches. We hence do not consider it.

For DeepFM, it needs a lot of validation for modifying the feature crosses generation method and the tuning of neural network parameters to gain better performance. However, time is a huge constraint, and we do not have enough powerful computational resources to execute the model repeatedly.

XGBoosting uses pre-sorting, where the features of the nodes are pre-sorted before iteration and traversed to select the optimal segmentation point; the greedy method is time-consuming when the amount of data is large. It is less time consuming than LightGBM, which uses the histogram algorithm and takes up less memory and has a lower complexity of data segmentation. The mean accuracy score of LightGBM is the highest among four models. So we choose LightGBM as our final model.

Table 1. Hyperparameters of Models

Model	Parameters
Random Forest Classifier	n_estimators = 50, min_samples_split = 10, max_features = 6, max_depth = 6, random_state = 1, min_samples_leaf = 2, n_jobs = 2
DeepFM	feature_sizes = 22-24, embedding_size = 256, deep_layers = [512, 256, 128], epoch = 3, batch_size = 64, learning_rate = 1.0, l2_reg_rate = 0.01, deep_layer_activation = sigmoid
XGBoost	objective= 'rank:ndcg', n_estimators= 3000, max_depth = 3, learning_rate = 0.1, subsample = 0.5, early_stopping_rounds = 100, seed = 7
LightGBM	objective = "lambdarank", metric = "ndcg", n_estimators = 4000, learning_rate = 0.1, max_position = 5, max_depth = 6, label_gain = [0, 1, 2], random_state = 7, seed = 7, num_leaves = 31, boosting = "dart"

7 Result

We used the scores of NDCG@5 from both the training and test sets for evaluation. From Table 2, it is easily to find that the lightGBM model worked best with the training score 0.5244792 and test score 0.416901. And the score on Kaggle is 0.40851.

Table 2. NDCG@5 Scores for Training and Test Sets

Model	Training	Test
Random Forest	0.33953	0.33997
XGBoost	0.466792	0.415757
LightGBM	0.5244792	0.416901

Future work We could begin by combining some relevant features into a tuple unit and using this as a more accurate input to the model. Then we can continue to improve the model, cf. Liu et al. [7], by using the ensemble learning approach to combine the models. And spend more time fine-tuning the hyperparameters to improve the accuracy of the model.

8 Lessons

Finishing the second assignment is not only a great challenge, where we make full use of the techniques that we learn from the course to explore the huge dataset but also a good chance to consolidate our data mining knowledge and let us solve a real ranking optimization problem. It was a fun exploration and adventure. We learned a lot about using Python libraries such as sklearn, and pandas for those purposes, like when doing EDA to visualize the distribution condition of the dataset and explore the relationship and correlation between various features, normalizing different features or discretizing features to optimize model training, utilizing operations for dataframe to preprocess multifarious features to clean the dataset and find the most influential features, choosing the most suitable and corresponding algorithms for recommendation task, and tuning parameters that have the best performance for models.

We believe that feature engineering of datasets in machine learning tasks is extremely important. Methods such as deleting unimportant features to save computational resources, reducing their negative impact on model performance, filling missing values suitably, normalizing some features with high outliers, and creating crossover features are inspiring. Not only that, but it is also necessary to consider the models used subsequently and different feature engineering may behave completely differently on diverse models. Furthermore, we get a deeper understanding of the most advanced L2R algorithms such as LightGBM, XGBoost and the deep learning model DeepFM, we also learned how to adapt the traditional ML model Random forest to solve the ranking problem as well.

References

1. Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.
2. Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

3. Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.
4. Vladimir Cherkassky and Yunqian Ma. Practical selection of svm parameters and noise estimation for svm regression. *Neural networks*, 17(1):113–126, 2004.
5. Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.
6. Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.
7. Xudong Liu, Bing Xu, Yuyu Zhang, Qiang Yan, Liang Pang, Qiang Li, Hanxiao Sun, and Bin Wang. Combination of diverse ranking models for personalized expedia hotel searches. *arXiv preprint arXiv:1311.7679*, 2013.
8. Aditi A Mavalankar, Ajitesh Gupta, Chetan Gandotra, and Rishabh Misra. Hotel recommendation system. *arXiv preprint arXiv:1908.07498*, 2019.
9. Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. Controlling fairness and bias in dynamic learning-to-rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 429–438, 2020.
10. Bryan James Prosser, Wei-Shi Zheng, Shaogang Gong, Tao Xiang, Q Mary, et al. Person re-identification by support vector ranking. In *BMVC*, volume 2, page 6. Citeseer, 2010.
11. Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 77–86, 2008.
12. Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, volume 8, page 6. Citeseer, 2013.
13. Raymond E Wright. Logistic regression. 1995.
14. Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398, 2007.
15. Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1059–1068, 2018.