On the Ranking of Expedia Search Results using LambdaMART

Thomas Bellucci$^{1[2710299]}$, Qingzhi Hu$^{2[13167200]}$, and Chih-Chieh Lin$^{1[2700266]}$

^{1} VU University, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands
{t.bellucci, c2.lin}@student.vu.nl
^{2} University of Amsterdam, Spui 21, 1012 WX Amsterdam, The Netherlands
qingzhi.hu@student.uva.nl

Group 55 DMT-2021

1 Introduction

This report presents the results of our research project on hotel search ranking. Over the last two decades, online travel agencies (OTAs) have increasingly become an important channel for the booking of hotels. Booking websites, such as *Expedia*, *Trivago* and *Booking.com*, which allow users to explore hotel options through online search, have now largely supplanted traditional, brick-and-mortar travel agencies and account for nearly half of travel sales worldwide [16].

For OTAs, service quality is paramount; when a search yields relevant hotel results, an OTA will maximize its chances of securing a booking [6]. In light of this, the *Personalize Expedia Hotel Searches* competition was organized on Kaggle in 2013. In this competition, teams were tasked with predicting which hotels shown in Expedia's search results were most likely to be booked by a user given their search query; this could then help Expedia organize, or sort, their search results more appropriately. Teams were provided with a dataset of search queries, results and purchasing information; from this, a model could then be built to sort hotels in novel search results based on their likelihood of booking.

In this report we present our solution to the Kaggle competition. We frame the problem as a *learning-to-rank* (LTR) problem, where the task is to sort hotels presented in the search results by their predicted likelihood of securing a booking. To achieve this, a set of features were identified from the competition data following a literature review and data analysis. On this data, a LambdaMART model was then trained. In what follows we will elaborate on our process, data analysis, modeling and evaluation, and reflect on the results obtained.

2 Business Understanding

Matching users to hotel listings is critical in the travel industry as inappropriate rankings may cause relevant hotels to be missed. Moreover, as shoppers are shifting between booking sites, rating hotels based on their competitiveness for the customer gives a travel agency the best opportunity of securing a deal.

Before proceeding with model building, a literature review was carried out to identify features predictive of booking, and solutions to the 2013 competition were examined to identify directions for modeling and feature engineering.

2.1 Literature Review

Recent studies have indicated that the information provided by search impressions greatly affects booking behavior. In particular, a meta-analysis by Dolnicar found that booking behavior is largely dictated by characteristics inherent to hotel properties themselves including the quality of service (quantified through reviews), location, reputation, name familiarity and price of the accommodation [4]. However, besides qualities of the hotels themselves influencing the customers' purchasing decisions, it was found that website and search engine characteristics may affect booking behavior as well. According to recent findings, the quality and completeness of search impressions positively contributes to the customer's purchasing decision [11]. Moreover, the position of hotel impressions in the results was found to affect click-through rate, increasing the likelihood of booking for hotels presented early in the results [6].

2.2 Competition Solutions

Although the Kaggle competition ended in 2013, many details of its submissions have been made public since. The unofficial first place, held by team 'Commendo' [17], was an ensemble of models which included the LambdaMART model from RankLib [3], gradient boosting machines and neural networks, and resulted in a score of 0.54071 on the NDCG@38 metric (see section 5.3.). The solution employed all numerical features and described each hotel property by additional various statistics, including the mean, median and standard deviation of several features. These statistics were computed over the training and test set to improve the stability and generalizability of the estimates.

The official winning solution was provided by Owen Zhang [1, 17] who used an ensemble of gradient boosting machines optimizing the NDCG metric. All features provided in the dataset were used along with the mean of the numerical features computed per hotel property, search id and destination. Although a hotel's position on the result page was only available in the training set, it was incorporated as an historical feature allowing it to be used for the test set; precisely, the position was based on the hotel property id, the destination, the month of booking and the hotel position in previous and next search. He also introduced various composite features to better describe price differences [1].

The first place solution removed outliers of the numerical features, imputed missing data with negative values and applied down-sampling of no-click records which improved training time and performance [1]. For dealing with categorical features, so-called EXP features were computed to transform them into a numerical space; specifically, computing for each value of the categorical feature the mean of the booking and clicking feature (disregarding the current observation).

The second place solution was built on the insight that users tend not to favor hotels with missing information in their impressions [1, 11]; in light of this, missing values in search impressions were imputed with worst case values. As travel expenses may fluctuate throughout the year and different destinations generally vary in price, features were normalized with respect to the property

Table 1. Excerpt of the training data. For brevity, not all attributes are shown.

Search query					Property characteristics						Price competitiveness			Target	
search id	date time	room count		adults count	prop id	review score	promotion flag		position	price usd	comp1 rate		comp8 inv	click bool	booking bool
81578	2012-11-01 00:08:29	1	...	2	93974	4.5	1	...	12	195.00	NA	...	0.0	0	0
81578	2012-11-01 00:08:29	1	...	2	133689	4.5	0	...	2	179.00	NA	...	0.0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
67044	2013-06-30 23:58:24	1	...	3	39164	4.0	0	...	4	149.92	0.0	...	0.0	1	1
67044	2013-06-30 23:58:24	1	...	3	32887	4.0	1	...	8	179.08	0.0	...	0.0	1	0

Table 2. General statistics of the training data.

Records	
Total Records	4958347
Total Searches	199795
Records Clicked (%)	4.47
Records Booked (%)	2.79
Searches No Clicks (%)	0.00
Searches No Booking (%)	30.73

Searches	
Min. Results	5
Max. Results	38
Avg. Results	24.82
Avg. Clicks	1.11
Avg. Bookings	0.69

Groups	
Properties	129113
Countries	172
Destinations	18127
Visitor Countries	210
Sites	34

and search ids, destination, country and time of year. Wang considered multiple modeling approaches, the best of which was found to be LambdaMART.

The third place solution, developed by the 'Binghsu & MLRush & Brick-Mover' team [12], trained multiple types of models, including support vector machines, random forests, factorization machines and neural networks; models were then combined using a listwise ensemble.

A team from Stanford University used collaborative filtering and several traditional classification algorithms to predict clicking directly [9]. With extensive feature selection, the team obtained limited classification performance ($F_1 = .38$), showing how the classifier-based approach may not be appropriate for the problem at hand.

3 Data Understanding

In order to gain a thorough understanding of the structure and quality of the provided dataset and identify attributes predictive of booking, an exploratory data analysis was conducted.[3] As shown in Table 1, the dataset was provided as a collection of time-stamped search results, each line of which representing a combination of a search query with a specific hotel property that was part of the results. Hotel properties were represented by a set of attributes, including its ID, review score, star rating, price (in US dollars), location score and position within the search results.[4] Information regarding the query included the search ID, date time, destination, length of stay, room count, number of adults/children

[3] A test set, described in section 5.2, was held out as to not inflate evaluation statistics.

[4] The data included two location scores; prop_location_score1 and prop_location_score2

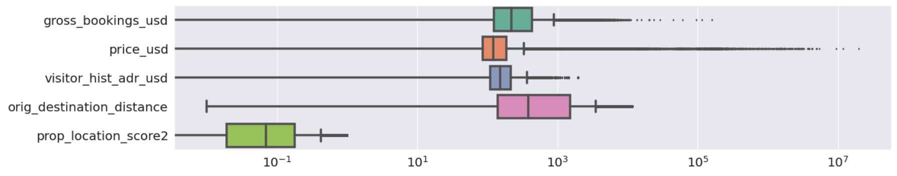

Fig. 1. Box-plots of a selection of raw attributes

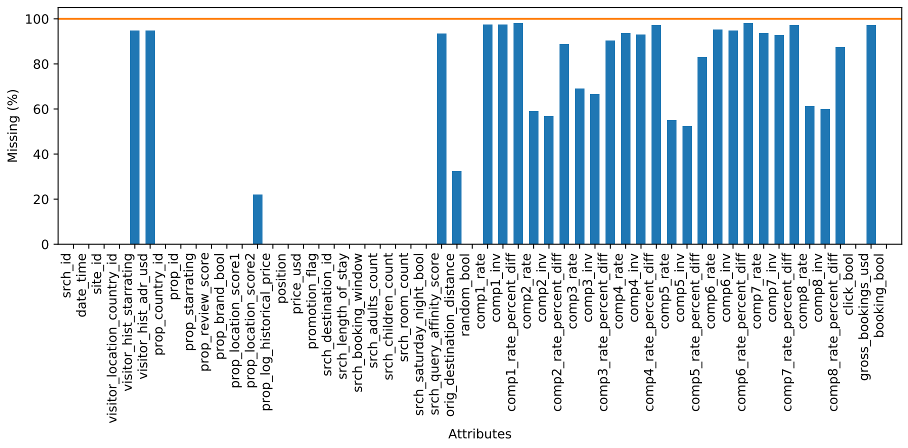

Fig. 2. Proportion of missing values of each attribute in the data

and booking window.[5] When available, competitor rates and historical customer information were included as well. In total, the data set included 8 months of searches, from November 1st, 2012, to June 30th, 2013.

In Table 2, various general statistics are shown for the data. Here, it can be seen that the number of search results varied considerably between queries (from 5 to 38). Moreover, it shows that, on average, more than one property is clicked on by a user (with a minimum of 1), yet frequently no bookings were made as a result of a search (30.73%). Fig. 1 and Fig. 2 show the distributions of a subset of attributes and their percentage of missing values; you can see that many of the attributes (e.g. price_usd) exceed their expected value ranges and often show a long-tailed distribution of high values and outliers. As shown in Fig. 2, certain attributes also contain missing data in the form on NaNs.

From the figures above, it can be concluded that the data in its raw form presents some challenges. The first issue to address is the imbalance in the data. As shown in Table 2, most search results were not booked or clicked, resulting in an imbalance of negative (no-click) samples relative to positive (click) samples in the data; however as we are dealing with ranking instead of classification and

[5] Time between the search and the time of reservation.

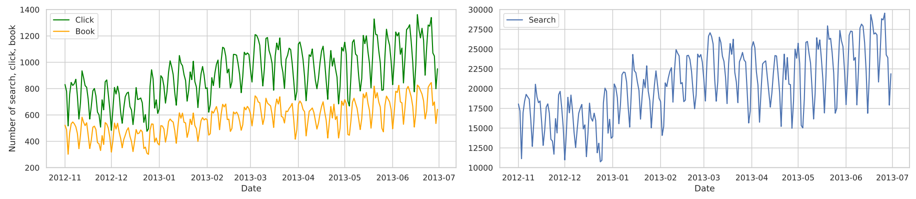

Fig. 3. Number of search queries, clicks and booking per day.

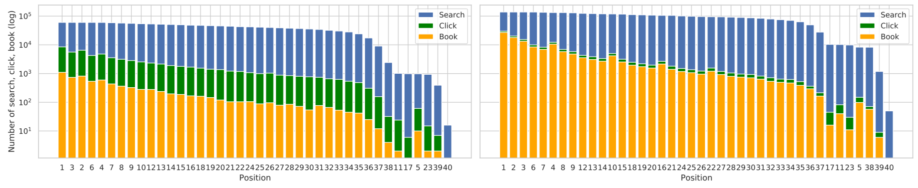

Fig. 4. Search, click and booking distribution over positions when (left) displayed sort was random and (right) sort was made according to Expedia's internal ranking.

only 4.47% of results were clicked, the common practice of down-sampling before training would likely result in data sparsity and hence over-fitting.

Another issue to address is how to encode categorical attributes as applying simple one-hot encoding would be very memory-consuming, especially for attributes with many possible values such as the destination id. Furthermore, since the data contains reservations that have been made over the year, temporal information may need to be accounted for (see Fig. 3)

In the data, a special attribute named *random_bool* was present indicating whether the returned results lists were sorted by Expedia's internal ranking algorithm or randomized. Since users are most likely to click on hotels early in the results, this attribute may influence indirectly (through interaction) the likelihood that hotels are clicked. Thus, one may need to pay attention to the position bias even if the hotels are sorted in random order (see Fig. 4).

Lastly, some attributes contained a big portion of missing values and outliers. Missing values were especially noticeable in price competitiveness and user history attributes (see Fig. 2); outliers in price-related attributes (see Fig. 1).

3.1 Univariate Attribute Analysis

In order to gain insight into the predictive value of the raw attributes, a ranking-specific feature selection procedure was employed [5]; rankings were created with respect to each attribute in the data which were in turn scored using the NDCG@5 metric.[6] Fig. 5 shows the result of this procedure relative to a random baseline. In line with previous research, the position of the property within the

[6] As some attributes may negatively correlate with booking, a single-attribute regression model was fit to allow for arbitrarily signed relations.

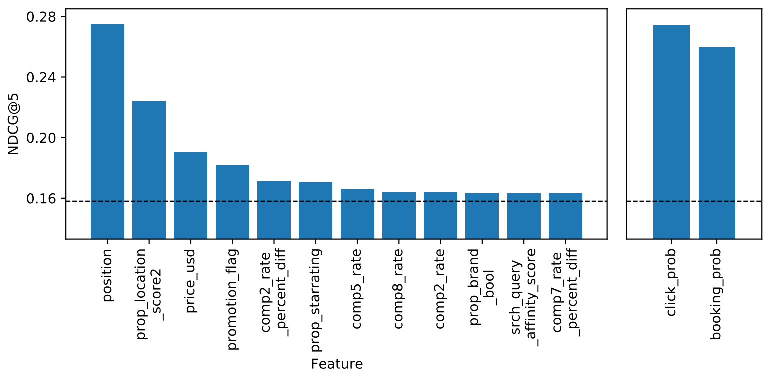

Fig. 5. NDCG@5 scores for raw attributes (left) and derived historical attributes (right) relative to the random baseline (NDCG@5=.159). For brevity, attributes scoring below baseline were left out. The NDCG metric is described in section 5.3.

results, its second location score and the price show to be most predictive of booking. However note that, in isolation, most attributes are limited, outscoring the random baseline by only a few percentage points.

Prior probabilities of booking and clicking for hotel properties were also examined and showed to provide powerful priors for ranking (see Fig. 5).

4 Data Preparation

4.1 Feature Engineering

From the literature review and exploratory data analysis a number of relevant features were identified and incorporated into the feature set using the pandas Python library. For one, in light of the feature analysis, a subset of the raw numerical features were incorporated, which included the room price, review score, star rating, its location scores, the brand flag, the promotion flag and all competitor rates and search attributes. Categorical ID attributes were not utilized as they were not found to provide discriminative information.

In addition to the raw numerical features, several composite features were found to be predictive from the exploratory data analysis and literature review. For one, as the hotel's position within the search results was shown to be highly predictive of whether it would be clicked or booked, the expected position of a property was added to the feature set.[7] For each hotel property in the training set, the mean of the position in the training data was computed according to Eq. 1 and used as a proxy for the hotel's position in the search results:

$$h^{(*)}_{mean\_position} = \frac{1}{|I_h|} \sum_{i \in I_h} h^{(i)}_{position} \tag{1}$$

[7] As positions of the properties within the search results were not provided for queries in the competition test set, positions needed to be estimated from the training data.

Here, $h_f^{(i)}$ represents the value of feature f for hotel h in results list i and I_h denotes the set of all result lists containing h. In addition to the mean, the standard deviation of the position in the training set was computed for each property as to represent the uncertainty in position. As a fraction of the search results were ordered randomly and so would add noise to the estimates, care was taken to exclude searches where the ordering was random (random_bool = 1).

Furthermore, in light of the feature analysis, the prior probability of clicking or booking a property was incorporated as well. These functioned as an indirect quality estimate of the hotel impression and were derived as follows:

$$h^{(j)}_{click\_prob} = P(click|h) = \frac{\left(\sum_{i \in I_h} h^{(i)}_{click\_bool}\right) - h^{(j)}_{click\_bool}}{|I_h| - 1} \tag{2}$$

$$h^{(j)}_{booking\_prob} = P(booking|h) = \frac{\left(\sum_{i \in I_h} h^{(i)}_{booking\_bool}\right) - h^{(j)}_{booking\_bool}}{|I_h| - 1} \tag{3}$$

The current impression j is subtracted from the estimate as to not leak the value of the target $h^{(j)}_{click\_bool}$ into the feature set. As the obtained position and probability estimates may be noisy or unstable, the number of previous search results containing the hotel, $|I_h| - 1$, was added as a feature set as well.[8]

As mentioned in section 2.2, statistical features were key to achieving performance for the top runners of the 2013 competition [3]. These features were implemented for each property h as follows:

$$h^{(*)}_{statistic\_f} = statistic(\{h_f^{(i)} \mid i \in I_h \text{ and } h_f^{(i)} \neq NaN\}) \tag{4}$$

After experimenting with various statistics, the following were included: the mean, median, standard deviation, minimum, maximum, quantile, sum and normalized rank within the search. We ultimately computed feature statistics over the following attributes:

visitor_hist_starrating	prop_log_historical_price	srch_children_count
visitor_hist_adr_usd	price_usd	srch_room_count
prop_starrating	orig_destination_distance	srch_destination_id
prop_review_score	srch_saturday_night_bool	srch_booking_window
prop_brand_bool	srch_length_of_stay	promotion_flag
prop_location_score1	srch_query_affinity_score	random_bool
prop_location_score2	srch_adults_count	

4.2 Outlier Removal and Imputation

As several feature types were implemented from continuous numerical features to bounded probability estimates, different imputation strategies were necessary. The base features and probability estimates were imputed with a default value

[8] To indicate that some properties occurred only a small number of times, making the position and probability estimates unstable.

of zero as this value was most intuitive. Statistical features were imputed with -1 when undefined (e.g. mean of NaNs). The remaining features, including the estimated position, were imputed with the average position in the training set.[9]

As the data contained considerable outliers in price (see Fig. 1), rows where $price\_usd > 1e^4$ were discarded. Contrary to previous solutions, down-sampling of negative (no-click) records was found to cause overfitting; hence, no reduction in the number of samples was performed beyond outlier removal.

5 Modeling and Evaluation

5.1 LambdaMART

As the problem of sorting hotel search results based on booking prospects is inherently tied to the problem of ranking, a LambdaMART model was implemented [18]. On a high level, LambdaMART is a boosted ensemble of regression trees which can be used for optimising rankings with respect to LTR-specific evaluation metrics such as NDCG. LambdaMART works in a pairwise fashion; that is, it selects a pair of items (i, j) from the results list and computes for it a gradient λ_{ij}. This gradient, the so-called *lambda*, acts as a 'force' which moves the items in the pair up or down the results in opposing directions. By computing gradients λ_{ij} proportional to their change in NDCG for every pair of items in the result list and having the regression trees learn to model the lambdas, we can then learn how to perform listwise ranking.[10]

The rationale for choosing LambdaMART over other ranking approaches is its ability to take into account non-smooth loss functions in its optimization, allowing it to directly optimize for our NDCG evaluation metric (see section 5.3). Moreover, an efficient implementation of the model was available through the LightGBM package [10] and similar ensemble methods had shown in the past to be successful in the competition (see Section 2.2).

The training data for LambdaMART was also easy to obtain as it only required the available data matrix of results to be grouped by search ID and a corresponding list of relevance scores to be constructed (see section 5.3).

5.2 Stochastic Hill-Climbing Hyper-Parameter Optimization

The boosted trees underlying LambdaMART are highly parameterized allowing the model to be further optimized by hyper-parameter tuning. To tune the parameters, the data was randomly split up into mutually-exclusive training, validation and test sets; here, 5% of search IDs were used for validation, 5% for testing and the remaining 90% for training.[11] On the training and validation

[9] As features are assumed to be monotonic with the target, imputing with zero in the *position* variable would mistakenly render all missing hotels to be ranked highly.

[10] For a thorough description of LambdaMART, the reader is deferred to [2].

[11] A fixed split was used as sufficient data was available and cross-validation was found to be prohibitively expensive.

Table 3. Hyper-parameters before and after optimization using SHC.

Parameter name	Initial value	Final value
Number of leaves	31	28
Number of estimators	100	866
Maximum tree depth	10	9
Feature fraction (for each tree)	1	0.927
Bagging fraction	1	0.958
Bagging frequency	1	18

sets, a stochastic hill-climbing (SHC) algorithm was then applied [15]; with SHC, an initial guess is made for each hyper-parameter (i.e. the parameter defaults) which are then incrementally improved upon by making small adjustments and validating these adjustments on the validation set. As features varied in magnitude, adjustments were uniformly sampled for each feature between zero and 10% of the current feature value. 50 iterations of SHC were performed.

This semi-directed optimization approach was used as the training time increased exponentially with the number of estimators, prohibiting the use of an exhaustive grid search. Also, non-directed alternatives such as random search were deemed impractical as many hyper-parameters needed to be optimized.

For the gradient boosted trees, several parameter configurations were examined including variations in the number of leaves, number of estimators, bagging fraction, bagging frequency, feature fraction and maximum tree depth (see Table 3). In the end, a model using 800 estimators, a bagging fraction of 0.95, a feature fraction of 0.95 and a maximum tree depth of 23 was found to be optimal.

5.3 Evaluation Setup

In order to obtain a final evaluation score for the ranker, a final evaluation was performed. After optimizing for hyperparameters using SHC, LambdaMART was retrained on all available training and validation data and evaluated on the 5% held-out test set, containing approximately 240.000 records.

To evaluate the performance of our ranker the Normalized Discounted Cumulative Gain (NDCG) metric was used. NDCG is a frequently used metric for determining the consistency of rank for a set of search results [8]. It makes the following assertions: results that are extremely important are more useful than results that are relatively relevant, which are more useful than results that are insignificant (cumulative gain).

The evaluation metric used in this competition was NDCG@5, that is the NDCG restricted to the top 5 search results, averaged over all queries. For a single query, NDCG@k is defined as follows:

$$NDCG_k = \frac{DCG_k}{IDCG_k} \in [0,1], \text{ where } DCG_k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}$$

Here, rel_i is the relevance score assigned to the item with rank i and IDCG_k is the DCG_k of the optimal ranking. The following relevance labels were used:

$$rel_i = \begin{cases} 5 & booking\_bool_i = 1 \\ 1 & click\_bool_i = 1 \\ 0 & otherwise \end{cases}$$

To quantify the importance of individual input feature according to the models, we compute two feature importance metrics; the number of splits made on each feature and SHAP values [13, 14].

5.4 Final Model Description

In this section we briefly recap the final model used. The model was LambdaMART implemented using LightGBM in Python. The model consisted of 866 boosting trees, each of which has a maximum depth of 9 and were trained on 92.7% of the features. In addition to boosting, the model employed bagging with a bagging fraction of 0.958.

In line with the final evaluation, the optimization criterion was NDCG. To obtain a final evaluation score, a model with the optimized parameter configuration was trained on all the available training data (95% of total data) and tested on a held-out test set (see Evaluation).[12] In addition, the final model was evaluated on a public test set provided by the competition.

6 Results

6.1 Evaluation

Table 4 shows the NDCG@5 score of our final LambdaMART model. On the public competition leaderboard the model obtained a score of 0.41726, equivalent to second place in the competition.[13] As can be seen from the obtained validation scores, we were able to obtain a score around 0.38 using a subset of the raw input features (described in section 4.1); however, only when additional composite features (i.e. mean position, click/booking probabilities) and statistic features were included were we able to obtain a score approaching 0.41.

Nonetheless, from table 4 it can be seen how the model overfits on the training data; relative to the training NDCG score obtained, the performance of our model drops off considerably on the validation and test data. However, when training on all available training data, overfitting is reduced as reflected by the improved performance on the test set relative to the validation set.[14]

Feature importances for our model are shown in Fig. 6 and 7. Although the two importance methods shows slightly different results, they agree on many of the important features. According to Fig. 6 the model stresses the importance of including the price of the accommodation and its prior clicking probability along with, to a lesser extent, the *booking_prob*, *price_dispersion* and

[12] Care was taken to evaluate the model throughout only on the validation set.
[13] Model predictions were submitted to an in-class competition leaderboard.
[14] The model was trained on all available training data before the final evaluation.

Table 4. Evaluation scores of the model for several feature subsets. The last row shows the scores obtained for the final model with all features.

Model	Training	Validation	Test	Leaderboard (public test set)
Base features	0.42237	0.38287	-	-
Base features + Position	0.43472	0.38893	-	-
Base features + Position + Click/Booking probs	0.44964	0.39746	-	-
Base features + Position + Click/Booking probs + Statistical features	0.48509	0.40946	0.41285	0.41726

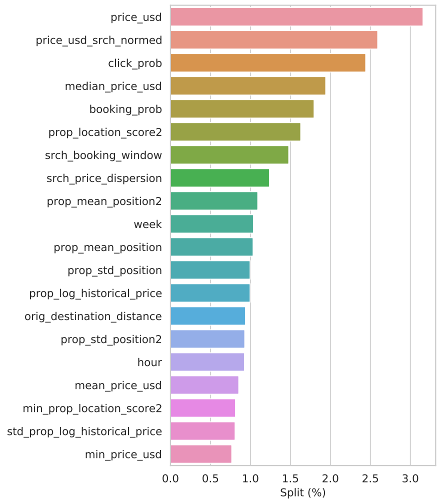

Fig. 6. Feature split importance plot

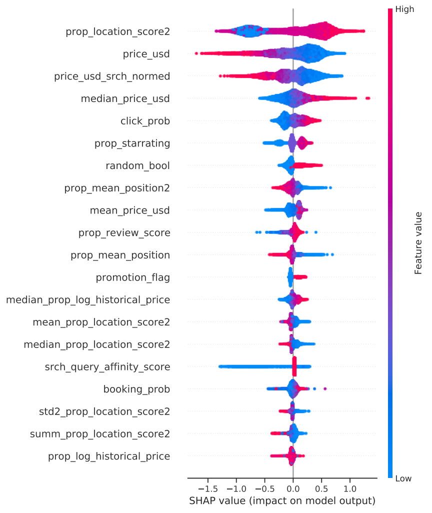

Fig. 7. SHAP summary plot

prop_location_score2. This is in line with previous findings showing these variables to either affect or reflect customer booking behavior (section 2.1 and 3.1).

In terms of the statistic features, the median and mean were found to be most important according to Fig. 7; this was to be expected as previous approaches utilized these features as well with great success. Also, the room price normalized within search results (*price_usd_srch_normed*), showed to bear importance.

6.2 Discussion

Feature engineering was found to be an important part of our solution (see Fig. 4). We manually created many varieties of features covering an expectation of the position, price features and click and booking probabilities. We also grouped numerical features by property ID and calculated various summary statistics from it to be used as features. Previous solutions from the competition solely considered the mean, median and standard deviation of the numerical features grouped by property ID; we expanded upon this by grouping the data by more ID variables (i.e. country_id) and considering quantile, rank, minimum and maximum. However, there remains space for improvement in terms of encoding categorical variables; in the current work, these were not considered.

As for imputing missing values, we followed the approach suggested by the winning contestants of the original competition by filling these values with either zero, a negative value or the mean, depending on the variable type, which yielded reasonable results. What we could have improved on is to try to impute missing values using a modeling approach, as opposed to defaulting to predefined values.

In addition, it was of critical importance to determine what target we wanted to predict. In our case, we had two options, *booking_bool* and *click_bool*, both of which were relevant to the task at hand. By combining these variables into a single *relevance* variable and optimizing for relevance directly using LambdaMART we mitigated this issue. However, as LambdaMART has been around for a while, one can imagine how results could improve when using more contemporary models. Researchers have recently developed new strategies dubbed "unbiased learning-to-rank" that use click data to remove location bias and train a relatively high-performance ranker. For instance, Unbiased LambdaMART [7] is an example of a "unbiased learning-to-rank" algorithm that effectively "debiases" click data and improves relevance rankings.

Lastly, in the initial stages of development we attempted down-sampling of negative, no-click instances as to improve training times; however, we abandoned this idea due to the fact that it resulted in severe overfitting issues. In the future, it could be worthwhile to further validate whether down-sampling may possibly improve performance or how to build a model that gives faster training times without negatively affecting performance. Furthermore, techniques such as model ensembling may be implemented to improve results.

6.3 What We Learned

In this competition, we were faced with a problem and evaluation metric that were related to the problem of ranking; a problem which some of us were inexperienced with. However, we found it very rewarding to have our score on the leaderboard at the end of the day to motivate us to improve further.

The most important lesson learnt from the competition is that "great oaks from little acorns grow"; that is, it is important to start small and build up more complex systems step-by-step. When you keep track of the changes made, it enables you to spot small errors along the way and so diagnose what goes wrong more quickly. Other lessons learnt from this competition were that it is often beneficial to investigate prior solutions to (similar) problems as to not reinvent the wheel, and that it can be useful to examine multiple approaches concurrently in order to identify which is most suitable to the problem at hand.

Lastly, we realized the importance of making use of the online community. We have benefited a lot from using the Kaggle community where people discussed new algorithms, modeling approaches and their pitfalls. Overall, we enjoyed the challenges posed by the competition and feel like we have gained more data mining experience, because of it.

References

1. Presentations of the icdm '13 workshop. "https://storage.googleapis.com/kaggle-competitions/kaggle/3504/media/ICDM2013_Presentations_2013-12-08.zip" (2013)
2. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
3. Dang, V.: The lemur project-wiki-ranklib. Lemur Project (2012)
4. Dolnicar, S., Otter, T.: Which hotel attributes matter? a review of previous and a framework for future research (2003)
5. Geng, X., Liu, T.Y., Qin, T., Li, H.: Feature selection for ranking. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 407–414 (2007)
6. Ghose, A., Ipeirotis, P.G., Li, B.: Examining the impact of ranking on consumer behavior and search engine revenue. Management Science **60**(7), 1632–1654 (2014)
7. Hu, Z., Wang, Y., Peng, Q., Li, H.: Unbiased lambdamart: An unbiased pairwise learning-to-rank algorithm (2019)
8. Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: ACM SIGIR Forum. vol. 51, pp. 243–250. ACM New York, NY, USA (2017)
9. Jiang, X., Xiao, Y., Li, S.: Personalized expedia hotel searches (2013)
10. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30**, 3146–3154 (2017)
11. Liu, J.N., Zhang, E.Y.: An investigation of factors affecting customer selection of online hotel booking channels. International Journal of Hospitality Management **39**, 71–83 (2014)
12. Liu, X., Xu, B., Yuyu, Z., Yan, Q., Pang, L., Li, Q., Sun, H., Wang, B.: Combination of diverse ranking models for personalized expedia hotel searches (11 2013)
13. Lundberg, S., Lee, S.I.: A unified approach to interpreting model predictions. arXiv preprint arXiv:1705.07874 (2017)
14. Lundberg, S.M., Erion, G.G., Lee, S.I.: Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888 (2018)
15. Panichella, A.: A systematic comparison of search-based approaches for lda hyperparameter tuning. Information and Software Technology **130**, 106411 (2021)
16. Talwar, S., Dhir, A., Kaur, P., Mäntymäki, M.: Why do people purchase from online travel agencies (otas)? a consumption values perspective. International Journal of Hospitality Management **88**, 102534 (2020)
17. Wind, D., Winther, O.: Concepts in predictive machine learning (2014)
18. Wu, Q., Burges, C.J., Svore, K.M., Gao, J.: Ranking, boosting, and model adaptation. Tech. rep., Technical report, Microsoft Research (2008)