Assignment 2 - Data Mining Techniques A Real Life Competition

Jingye Wang[1][2702383], Tianshi Chen[1][2727351], and Yiming Xu[1][2696855]

VU Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands
{t5.chen,j18.wang,y13.xu}@student.vu.nl

1 Introduction

This report describes our work on the Kaggle competition[1]. The competition is based on the previous competition[2] about the rank of Expedia's hotels' search list. The task is to rank the given search results in an order that the user who performs the searching is most interested in. A hotel with a higher order of rank means the user is more likely to book the hotel or view the detail of the hotel. To finish this task, we built and trained a predictive model that predicts the probability that the given user will click on the link of the hotels. Also, to fine-tune the model, we used a traversal method to acquire the input parameter that makes the model perform the best and a k-fold method to select the model with the highest accuracy.

This paper will introduce and discuss our approach following the route below. Firstly, we gain insights into the data from the business aspect and the data itself by looking into the given dataset overall and by reading the paper written by people who achieved high scores in Expedia's hotels' project previously. Secondly, we perform the explanatory data analysis to explore the distributions of the values in each feature and which features should be omitted. Thirdly, we use some feature engineering methods to fill in missing values of significant features, and construct new features from the existing ones. Fourthly, we built and trained a predictive model to rank the search results. Finally, we will discuss the advantages and disadvantages of our methods in detail.

2 Business Understanding

In commercial applications, recommendation algorithms are quite vital. A highly effective recommendation system can aid consumers in selecting products that meet their needs and help businesses retain clients. This project will solve Expedia's, a well-known global online travel service, hotel recommendation problem. Specifically, we aim to generate the optimal ranking of hotels for different users from the hotel library in order to enhance the probability of users clicking and booking hotels and decrease the probability of users bouncing to other websites.

[1] https://www.kaggle.com/competitions/2nd-assignment-dmt2022/overview
[2] https://www.kaggle.com/competitions/expedia-personalized-sort/data

This project was motivated by the Expedia Hotel Recommendations competition on Kaggle, and it is also a workshop challenge for ICDM 2013. Before beginning the project, we reviewed many information and papers from this competition.

Xudong Liu et al. [10] use random and balanced sample data to train the models. They introduced several methods in this paper: modified logistic regression, random forest, gradient boosting machine, extreme randomized trees, factorization machine, lambdaMART, and deep learning. They also describe how to ensemble those methods which achieve a considerably higher score which helps them get the fifth place on the leaderboard.

Dolnicar et al. [5] indicated that the hotel property itself, such as location, service quality, reputation, price, etc., have a strong influence on users' decisions when booking a hotel. Also, Liu et al. [9], and Ghose et al. [7] mentioned that The completeness of searching as well as the hotel's position in the searching result, also influence users' behavior significantly.

After the conclusion of the Kaggle competition, Expedia's official team announced the winning solutions [1]. This competition employs NDCG@38 as the sole metric for evaluating model performance. Officially, Owen Zhang won first place. He generated Numerical features averaged over srch id, prop id, and destination id, as well as EXP features and Estimated position using the property id, the destination id, the target month, and the location of the same hotel in the same destination in the prior and subsequent searches. His model's score is 0.53 on NDCG@38 after he finally adopted Gradient Boosting Machines as a model with NDCG as the loss function.

Jahrer et al. [4] earned the unofficial first place; they employed all numeric features, Average, Standard deviation, and Median of numeric characteristics per prop id. They then used a stacking model based on LambdaMART, Gradient Boosted Decision Trees, Neural Networks, and Stochastic Gradient Descent. The cumulative score of stacking models on the NDCG@38 is 0.54.

3 Data Understanding

3.1 Data Overview

This training dataset contains about 54 columns and 4958347 records. In the dataset, 34 fields are the float64 data type, 19 are the int64 data type, and only one field(date_time) is the object data type. The test sets include 50 columns and 4959183 records. The training dataset has more columns than the test dataset because the' position', 'booking bool', 'click bool', and 'gross booking usd' are only provided in the training sets. Based on the knowledge of the data and business understanding, we can classify all the features into six categories shown in Table 1. Moreover, according to our assignment goal, we recognized 'click_bool' and 'booking_bool' as our target values.

Table 1. The category of features

Prefix	Category	Description
prop_	hotel information	These features are related to information about the searched hotel, such as its customer review score, star rating, Etc.
compX_	Competitor information	These features are about the information of the searched hotel's competitor, like price and if the competitor has a similar and available room.
srch_	search query	These features are queries that the customer uses to search for hotels.
vistor_	visitor information	These features are basic information about the customer, which may help determine search ranking.
	other	These are the remaining features, such as price_usd, promotion_flag, and random_bool.

3.2 Data Statistics

To get insights into basic information about features, we calculate general statistics. Table 2 shows the statistics of a part of the features. From Table 2, we know that prop_location_score1 are not strictly associated with prop_location_score2. Moreover, only 2.7911% of customers will book the hotel after searching, according to the training dataset.

Table 2. The statistic of a part of features

feature name	count	mean	std	min	max
price_usd	9917530	241.7825	14341.81	0	19726330
gross_bookings_usd	138390	386.283316	821.190577	0	159292.380
orig_destination_distance	6701069	1307.018	2030.283	0.01	11692.98
visitor_hist_adr_usd	507612	176.588512	108.434842	0	2768.930
srch_query_affinity_score	635564	-24.302829	15.808004	-326.567500	-2.4941
prop_location_score1	9917530	2.875978	1.532092	0	6.980000
prop_location_score2	7739150	0.1303852	0.1595145	0	1

Aside from that, we analyzed the correlation between features as shown in Fig. 1. From the picture, we know that some features have relatively strong correlations. The feature pairs: visitor_hist_starrating and visitor_hist_adr_usd, prop_location_score1, and prop_location_score2 have positive correlations, which indicates that the first and second hotel desirability location scores are related to each other, and there is positive influence between customers' purchased history of mean price per night and star rating of the hotel. Aside from that, we also observed that there are some positive correlations among competitors. Furthermore, there also are some negative correlations between orig_destination_distance

and prop_country_id, prop_country_id and visitor_hist_starrating, srch_saturday_night_bool and srch_length_of_stay.

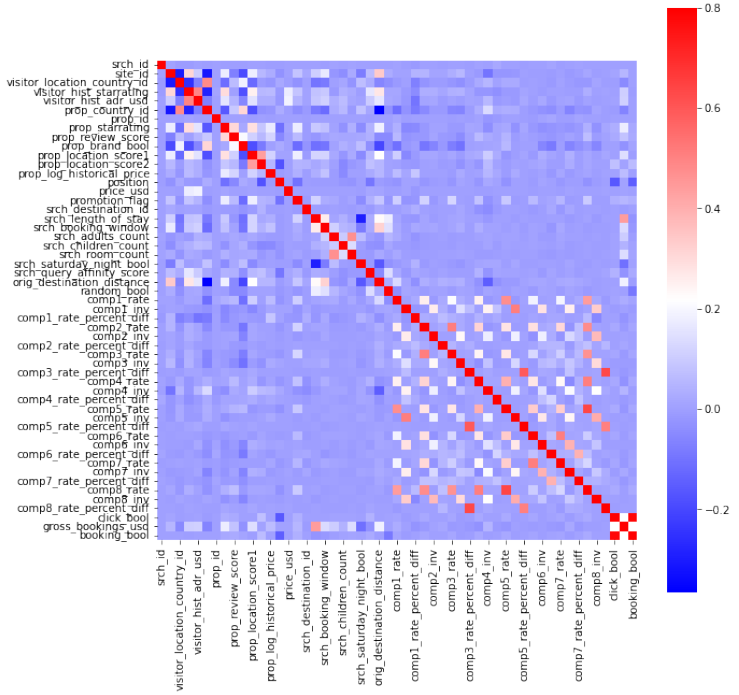

Fig. 1. Correlation of features

3.3 Data Distributions

In the beginning, we output the unique values to find if there are error values. We discovered that the features comp1_inv comp8_inv contain the -1 value, which is not defined in the description. Although it is not defined, we think this value carries special meaning, so we decided to keep these records. After this, we investigated the missing values in the dataset. Fig. 2 shows that from prop_review_score to gross_bookings_usd all contain missing values. Except for booking_bool,gross_bookings_usd,click_bool, position, from comp5_inv to comp1_rate_percent_diff lack more than 50% values. Because this analysis is based on the training and test datasets, the booking_bool, gross_bookings_usd, click_bool, and position features are expected to lack some values.

Besides analyzing missing values, we also analyzed all the features' outliers. Fig. 3 demonstrates some of the features which we think their boxplots are more important based on our business understanding. From Fig. 3, we discovered that

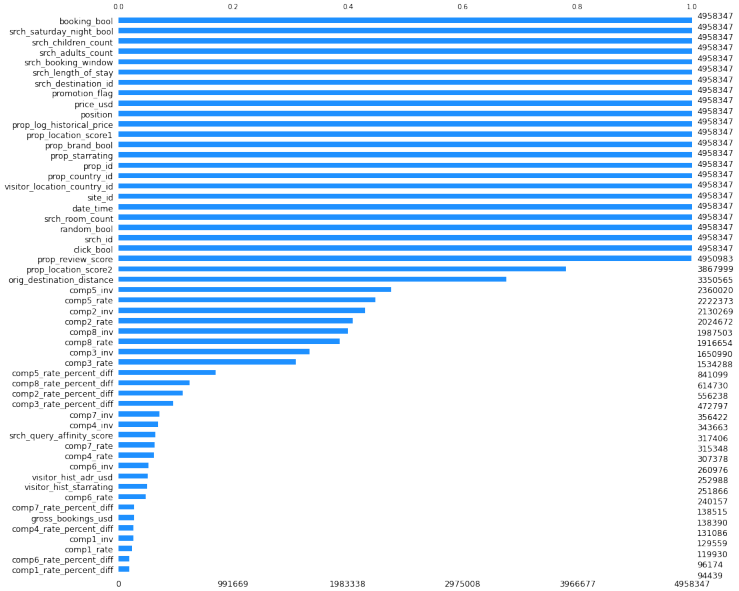

Fig. 2. Missing values of dataset

the value of price_usd contains some really high values, which causes a long-tail distribution. Based on section and Fig. 3, we know that its 75% quantifier is only 185, but the max value is 19726330, which means there are some outliers. Furthermore, Fig. 3 indicates that other features except for prop_location_score also contain outliers.

Additionally, we dig a bit deeper to determine if there is a relationship between some feature combinations. Considering the position feature is essential based on our business knowledge. However, this value is not strongly correlated with clicking or booking. We think that maybe there are other features combined with position to influence customer clicking and booking positively. Several features can be conjoined with position, such as random_bool, prop_brand_bool, and prop_review_score. Fig. 4 and Fig. 5 display the click and booking distribution under different condition. We can observe from the figures that the higher position number and the number of bookings and clicks are lower. Normal order, brand hotel, and 4.0 or 4.5 review score seem to have a higher chance of making customers click and book. As shown in Fig. 5, the 5.0 review score has a lower booking number. We think that is mainly because not many hotels have a 5.0 review score which is only around 8.7% in the dataset.

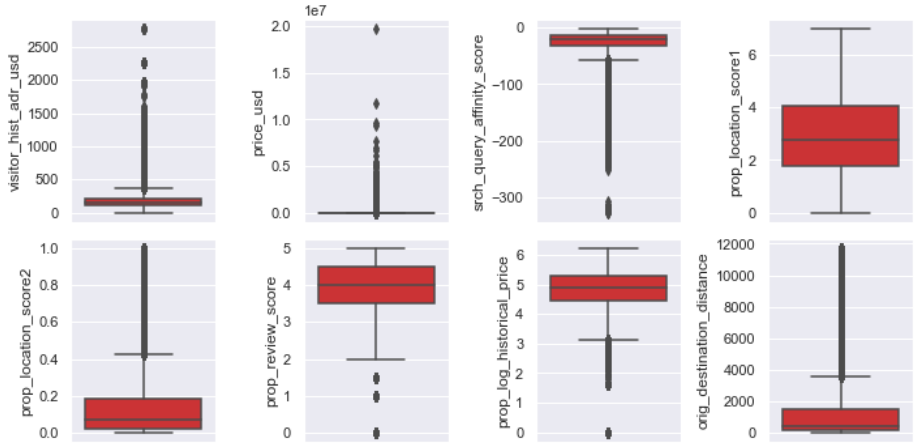

Fig. 3. The boxplot of selected features

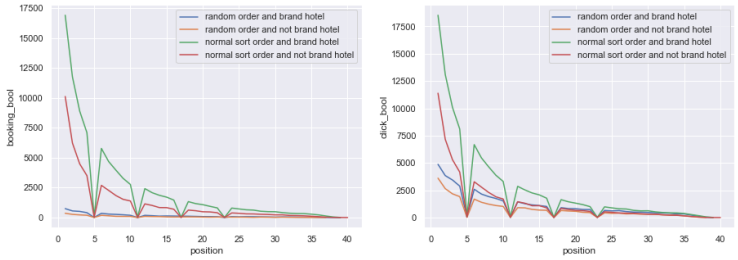

Fig. 4. Click and book distribution with position, random order, and brand hotel features combination

4 Data Preparation

4.1 Removal of outliers

From section 3.3, we discovered some outliers in the raw data, including a large number of extremely large outliers in hotel booking prices. As a result, for this feature, we decided to eliminate all training samples with prices exceeding $10,000 USD.

4.2 Implication of missing data

As described in section 3.3, there are numerous missing values in the raw data, with up to 90% of missing values in some variables. We believe these variables have little value for model training, so we eliminate them.

For *prop_location_score_2*, this variable is intuitively one of the most important factors that users consider when selecting a hotel. We discovered a strong relationship between this variable and *prop_location_score_1*. Since *prop_location_sc*

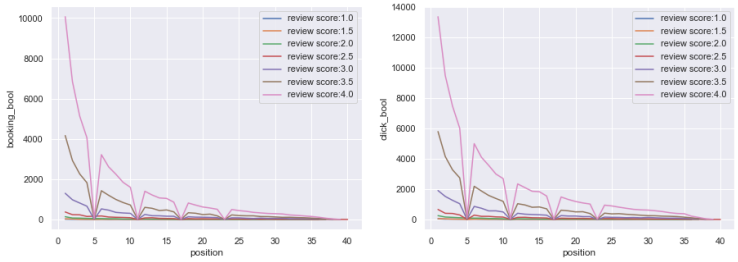

Fig. 5. Click and book distribution with position and review score features combination

ore_1 has no missing values, we developed a linear regression model to predict *prop_location_score_2* using *prop_location_score_1*.

For the features *prop_review_score, orig_destination_distance, srch_query_affinity_score, visitor_hist_adr_usd* and *visitor_hist_starrating*, we chose the median rather than the mean to fill in missing values in order to minimize the influence of outliers on the estimates. For the missing values of other features, we intuitively use 0 to fill the missing values.

4.3 Construction of new features

Informed by related work, we attempt to improve the performance of the model by adding certain features.

Time features We first separated the date-time variable into the year, month, day of the week, and hour. These features were developed on the premise that certain hotel bookings are cyclical and that the time of day a user searches for a hotel (such as day and night) may influence their decision-making.

Position Estimation The position of the hotel is one of the most important factors when users select a hotel; however, only the training set of raw data contains position information. To apply this information to the test set, we propose the following method for predicting hotel positions based on the search destination and prop_id:

$$x^j_{Position\_estimation} = \frac{\sum_j^N x^j_{position}}{N}$$

N is the number of occurrences of hotel j in the dataset. $x^j_{position}$ is the position score of a record.

Statistic features According to the ICML2013 official document, statistical features significantly improve the model's accuracy. Per Prop id and srch id, we calculated the mean, median, and standard deviation of numerical features.

$$x^i_{Statistic} = Statistic(i), \quad i \in Prop_j$$

CTR and CVR Click-Through Rate (CTR) and Click Value Rate (CVR) are crucial metrics for measuring the effectiveness of a recommendation system. In this task, CTR is defined as the probability that a user will see a hotel and then click to view it. CVR is the booking rate, as well as the probability that a user will book a hotel out of all the hotels they click on. The algorithms for CTR and CVR are as follows:

$$x^j_{cvr} = \frac{COUNT(x^j_{booking\_bool})}{COUNT(x^j_{click\_bool})}$$

$$x^i_{ctr} = \frac{COUNT(x^j_{click\_bool})}{COUNT(x^j_{Presentation})}$$

Other features Difference between the hotel's historical average price and its current list price:

$$x^i_{h\_diff} = exp(x^i_{prop\_log\_historical\_price}) - x^i_{price\_usd}$$

The difference between the user's historical average purchase price and the current list price:

$$x^i_{u\_diff} = x^i_{visitor\_hist\_adr\_usd} - x^i_{price\_usd}$$

The difference between the user's previous hotel star rating and the current hotel star rating:

$$x^i_{starrating\_diff} = x^i_{visitor\_hist\_starrating} - x^i_{prop\_starrating}$$

The total amount paid by the user to book a hotel:

$$x^i_{total\_fee} = x^i_{price\_usd} - x^i_{srch\_room\_count}$$

5 Modeling

5.1 LambdaMART

This project requires us to match a suitable hotel sequence for users, which is fundamentally a ranking problem. To solve the ranking problem, we employ LambdaMART [3]. LambdaMART is the combining model of LambdaRank and MART. LambdaRank optimizes RankNet by employing the symmetry of pair loss. Additionally, a heuristic weight (shown in Equation 1) is added to measure the impact of the different order of pairs in the sorting result on the performance of the model, allowing the model that uses NDCG as the evaluator to train with

improved results. Gradient Tree Boosting is an alternative name for Multiple Additive Regression Trees (MART) [6].

$$\lambda_{ij} = -\sigma(\frac{1}{1 + e^{\sigma(s_i - s_j)}}) * |\Delta NDCG|$$

We train the ranking model with LGBMRanker from the LightGBM [8] package, which implements LambdaMART and offers multiple boost methods whose performance has been validated in previous competitions.

5.2 Parameter Optimization

Optuna [2] is a framework for automatic hyperparameter optimization designed for machine learning. It provides a module called LightGBM Tuner for LightGBM models exclusively. Performance is evaluated experimentally.

We mainly adjust the parameters of max_depth, num_trees, learning_rate, num_leaves, feature_fraction, bagging_fraction, bagging_freq, and the final parameter adjustment result is shown in Table 3.

Table 3. Parameter tuning results using Optuna

Parameter	Initial value	Final value
num_leaves	192	31
boosting_type	gbdt	gbdt
max_depth	17	7
learning_rate	0.4	0.12
n_estimators	3500	5000
feature_fraction	1.0	0.52
bagging_fraction	1.0	0.45

5.3 Evaluation

Our Rank model is evaluated using the NDCG (Normalized Discounted Cumulative Gain) metric. NDCG is a common metric for assessing the precision of ranking outcomes. Typically, the recommendation system returns a list of items to the user. NDCG@K can be used to measure the difference between the top K ranks in the ranking list and the top K rankings in the user's real interaction list. In this project, we have set K to 5, focusing on the difference between the top 5 results and the actual results in the list of hotels that our ranking model recommends for each user.

The Normalized Discounted Cumulative Gain (NDCG) is a normalized variant of the Discounted Cumulative Gain (DCG) (DCG). A query's DCG@k is defined as follow:

$$DGG@K = \sum_{i}^{K} \frac{2^{r(i)} - 1}{log_2(i+1)}$$

Where k represents the truncation level and r(i) represents the relevance of the item ranked as number i. In our project, if the user purchased a room at the hotel, $r(i) = 5$, if the user clicked through to see more information on the hotel, $r(i) = 1$, if the user neither clicked on this hotel nor purchased a room at the hotel, $r(i) = 0$.

Then NDCG@k is defined as:

$$NDCG@k = \frac{DCG_k}{IDCG_k}$$

Where IDCG@K is the maximum possible DCG for this query, therefore, NDCG is a value between 0 and 1.

6 Results

On Kaggle's testing dataset, our model achieved 39.160% accuracy, which is not a high score that matches our effort to build and train the model. However, adding features to the dataset in feature engineering phases improved the predictive model's performance. Table 4 shows that we made four versions of the model with different numbers of features. When we only use the feature given, the accuracy on Kaggle's testing dataset is only 0.37672, which is the lowest among the four versions. When adding composite features, the accuracy raised to just less than 37.9%. Also, if we only compute and add the mean, median, and standard deviation of the numeric features to the dataset, the accuracy is slightly higher to 39.0%. Hence, individually adding composite features and numeric features could improve the model's accuracy. When we use both of them, the accuracy is further higher, at 0.39160.

Table 4. Accuracy comparison between different additional features

Additional features	Kaggle accuracy
No additional features	0.37672
Composite	0.37896
Numeric	0.39038
Both composite and numeric	0.39160

To figure out the significance of feature construction, we visualized the 20 most important features that we used to train the model, as Fig. 6 shows. The feature CTR (Click-Through Rate) is the most significant feature among all of the features, which means this feature contributed to the prediction largely.

Among the top 20 important features, there were 11 of them constructed manually, which proves that feature construction plays an essential role in improving the accuracy of the predictive model.

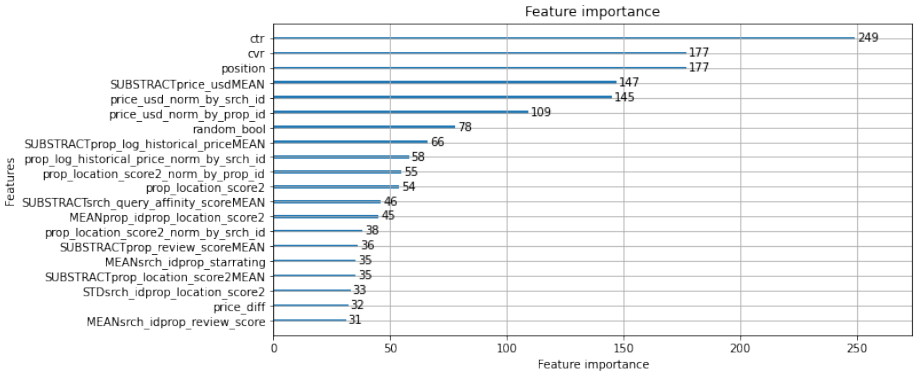

Fig. 6. Significance of features

To compare the performance of our model, we build a baseline model, a simple deep learning model containing three fully connected layers, as shown in Table 5, to regression with five epochs. Also, to compare the performance of our lambdaMART model, we built another model using XGBRanker. As Table 6 shows that the baseline acquired an accuracy of around 15.5% on Kaggle's testing set, which is the lowest in all three models, and the XGBRanker achieved approximately 38.3%, slightly less accurate than lambdaMART.

Table 5. Summary of baseline model

Layers	Output shape	Number of parameters
Input	192	0
Dense(relu)	1024	197632
Dropout(40%)	1024	0
Dense(relu)	1024	1049600
Dense	1	1025

Moreover, fine-tuning the model's parameters with Optuna also contributed to the accuracy of our model. Before we implemented Optuna functions, the accuracy on Kaggle was 0.38050. Because of the optimization provided by Optuna, we calculated the parameter for our model so that it could achieve higher accuracy. Although the parameter with the highest precision on the validation dataset might not mean it could gain the highest accuracy on the testing set, using it still improves our model.

Table 6. Accuracy comparison between different methods

Methods	Kaggle accuracy
Deep learning (baseline)	0.15498
XGBRanker	0.38301
lambdaMART	0.39160

7 Conclusion

Overall, our approach was able to finish the task with relatively high accuracy, for it used and combined several cutting-edge technologies such as lambdaMART. Adding features that did not exist in the training set improves the accuracy significantly. Using Optuna to traverse all possible combinations of parameters was reasonable and proved helpful.

However, there were still multiple disadvantages. First of all, the precision is not as good as we expect. This problem might be because we took some features that significantly influenced the users clicking and booking behavior into account despite too many empty values inside. Filling them in with a fixed value of inferring them from other features may reduce the precision of the whole model. Besides, the models we attempt to use easily get overfitted. Sometimes models with high accuracy on the validation test would get a low score on Kaggle. Also, using Optuna need to train the model too many times with different combinations of parameters, this process takes hours to finish, but its improvements are not worth the time it costs.

To conclude, we finished the task with a pipeline that contains explanatory data analysis, feature engineering, feature construction, and train a lambdaMART model. We also compared it with a deep learning model and an XGBRank model, which shows that our model improved compared to the other two models.

References

1. Presentation of Competition at ICDM 2013. `https://www.dropbox.com/sh/5kedakjizgrog0y/_LE_DFCA7J/ICDM_2013`
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 2623–2631 (2019)
3. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
4. David, Kofoed Wind, O.W.: Concepts in predictive machine learning (2014)
5. Dolnicar, S., Otter, T.: Which hotel attributes matter? a review of previous and a framework for future research (2003)
6. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
7. 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)

8. 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** (2017)
9. 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)
10. Liu, X., Xu, B., Zhang, Y., Yan, Q., Pang, L., Li, Q., Sun, H., Wang, B.: Combination of diverse ranking models for personalized expedia hotel searches. arXiv preprint arXiv:1311.7679 (2013)