Expedia data

Yuyu Bai—2732696 , Summer Xia—2703936 , Bowen Lyu—2727574

Vrije Universiteit Amsterdam – group 2

1 Introduction

1.1 Background

The development of the Internet over the past decade has led to the establishment of the e-commerce market. In the travel industry, electronic transactions have been gradually transformed. The core demand of Internet companies is "growth", and online tourism in the era of mobile Internet is no exception. With the continuous advancement of technologies such as big data, cloud computing and artificial intelligence, growth through algorithms and models has become a trend.

In recent years, the rapid rise of recommendation systems mainly solves the problem of information overload, helping users to efficiently obtain information of interest, and helping enterprises to improve user conversion rates. Several well-known online travel agencies (OTAs) have started offering personalized services to help customers choose their hotels. In this competitive market matching users to hotel inventory is very important since users easily jump from website to website. As such, having the best ranking of hotels ("sort") for specific users with the best integration of price competitiveness gives an OTA the best chance of winning the sale. Therefore, personalized recommendation services have also become very important for online tourism. Recommendations can liberate users from numerous travel choices, guide users to quickly find items of interest, and greatly simplify users' travel planning and purchases.

1.2 Problem definition

In this project, we use expedia (the world's largest OTA) dataset, which includes hotel data, user behavior data, and price competitiveness information. Our goal is to apply machine learning technology to rank hotels, based on the characteristics of a particular customer, and recommend hotels with a high match score to him/her, that is, to give top priority to "hotels that users are most likely to choose", thereby Increase CTR and booking rates.

In addition, this problem is a subset of the CTR projection scenario. The problem that CTR estimation needs to solve is: given a user and a query, and the doc matching the query, predict the probability of getting clicks after these docs are exposed. This probability value can be used in sorting scripts to improve search results and improve business indicators such as ctr.

This paper details the process of finding better algorithms for building personalized hotel selection and recommendation learning models. Our data, including features such as hotel characteristics, hotel location, aggregation of users' purchase history and information on competing online travel agencies, will be used to design a machine learning algorithm for these highly ranked hotels, hotels that have Higher probability is click/book. Our work will include model selection, optimization, feature selection and model comparation.

The structure of the paper is as follows: We begin with the data pre-processing method , which explains how we handled with our data and used it to generate various features, as well as the several models we'll be using in this section.Some scientific literature supports are also given here. Next is the results and discussion section 5, where we examine and compare the results we obtained from various models. Finally, a summary of what we accomplished in this project are given in conclusion.

2 Related Work

This hotel ranking problem can be classified as learning to Rank problem, which is one of the popular topic in Machine learning field recent years. [9] already give a very comprehensive overview in his work. Learning to rank is often refers to transfer the ranking problem to some supervised or semi-supervised machine learning problems. And three main approach also discussed in his work. The first one is point-wise approach, which transfer the ranking problem into a basic classifier or regression problem. And this is also know as "ctr" prediction, which is aim to predict the "click" or "booking" behaviour. From Yanwu Yang, Panyu Zhai's[17] work we know that State-of-the-art CTR prediction models reported in the advertising literature can be categorized into four groups: (1) Multivariate statistical models such as Logistic Regression; (2) Factorization machines (FMs) based models; (3) Deep learning models; (4) Tree based models. Pair-wise method also aim to turn such a problem into classification problem. But the label is weather a item is more relevent than another. The first important pairwise LTR models was RankNet[3] introduced by Chris Burges who is from MicroSoft Research. And he had done a set of very important theoretical and also empirical works, with established the foundation of the pairwise approach in this relam. Then follows LambdaRank[4] and LambdaMART[16]. A high quality implementation of LambdaMART available in the library lightgbm. We choose to use this method as our main method to implantation in our work. Listwise method adressing this problem in a more straightforward way. The listwise approach includes ListNet , ListMLE, AdaRank, SVM MAP etc[9]. Researchers from Google has generalized the LambdaMART framework to provide a theoretical background of the ranking model of all 3 types of loss function (pointwise, pairwise, listwise) and the direct optimization of all the popular ranking metrics (NDCG, MAP, ...). The framework is called LambdaLoss[15].

3 Method

3.1 data understanding

3.2 Data Preparation

For most of the machine learning models, data preparation and feature engineering is the most vital part for a good model performance. When it comes to feature engineering part. We've got a lot of inspirations from the pubic discussion board form the ICDM challenge 2013[1]. Also Liu et al.(2013)[Combination of Diverse Ranking Models for Personalized Expedia Hotel Searches] who take part in 2013's ICDM competition elaborate a lot about how they generate features upon this kind of data. Eventually, we develop following 4 kinds of features based on both their experience and our data analysis.

1. date related features

The raw data contain the original date stamp features. But such format can't be used to most of the ML models. And based on our business understanding, date info is related to the final target. Travel season and holiday surely will influence the booking behaviours. So it's worth to add date related features. We add month, day of the week and the hours to our features.

2. Features related to single search

In this expedia dataset, data in one search id are located in different rows. The information that there are in the same search will be ignored when using point-wise method and pair-wise method. But this information is non-negligible. When someone are searching a hotel from the website, it's usually the rank-related information that decide which one they will choose. Such information can be if the hotel is the cheapest compared to other options, if the hotel is the highest scored one compared to others and if the hotel got the highest score and most stars. To utilize this kind of information, we aggregated these numeric features such as "propstarring", "proplocationscores" by "searid" and then calculate it's statistical features like mean, standard derivation. The relative value can be obtained by normalizing it.

3. Features related to single hotel

Jankiewicz and Wójcik[7] use item-item similarity in their works. And it's worth trying to retrieve some hotel related features. Such features including, what's the cheapest price for this hotel, what's the difference between the current price and cheapest price, the history ctr(how many people click per exposure) and cvr(how many people book per click) of this hotel. This can also be calculated by aggregated and normalized by "propid". The method is similar to processing features related to single search. So we implemented functions to process aggregated and normalized features.

4. Normalized by destinations

Consumption level differs due to different destinations. We think that normalized by destinationid is also essential especially for priceusd. To producing

[1] Can be found in https://www.kaggle.com/competitions/expedia-personalized-sort/discussion

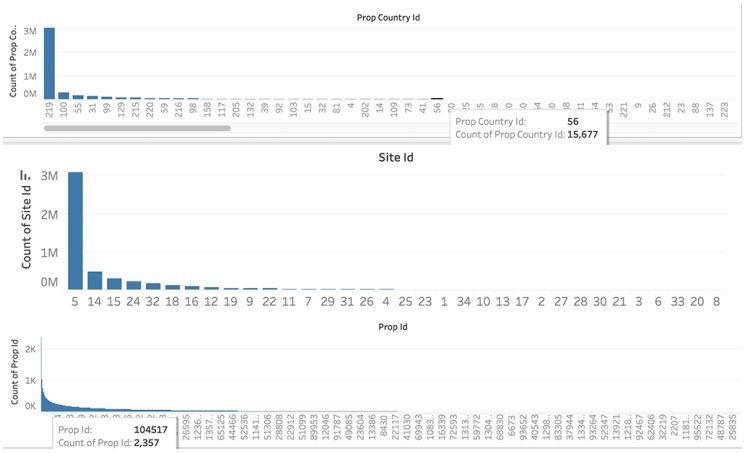

Fig. 1. Distributions

these features, we only need to change "propid" to "destinationid" by using the function mentioned in last paragraph.

5. Position bias

Position bias is one of the intractable problems in this realm. We need to rank it to decide the position on the one hand , different position can leads to different behaviour of users on another hand. In this dataset, since the training data and test data are from the same period of time, we can assume that their are in same distributions. So we purely use the history mean position for one prop to estimated the current position in our data.

3.3 Data analysis

As we mentioned, the training sample data has 5 million examples. It contains user clicks as well as bookings data including 18 continuous attributes and 31 categorical attributes.

Distribution investigation We started by looking at the distribution of some categorical columns using a histogram, as shown in Fig1.

As we seen, the distribution of *site_id, prop_id and country_id* are with some extreme skewness. These attributes are used to distinguish independent individuals which cannot be numerically normalized. Because of the skewness, we are going to treat them as sparse features by one-hot labeling or embedding methods.

Data Preprocess Normalization on continuous variables is essential, since it can speed up the gradient decent approach and improve the accuracy of the

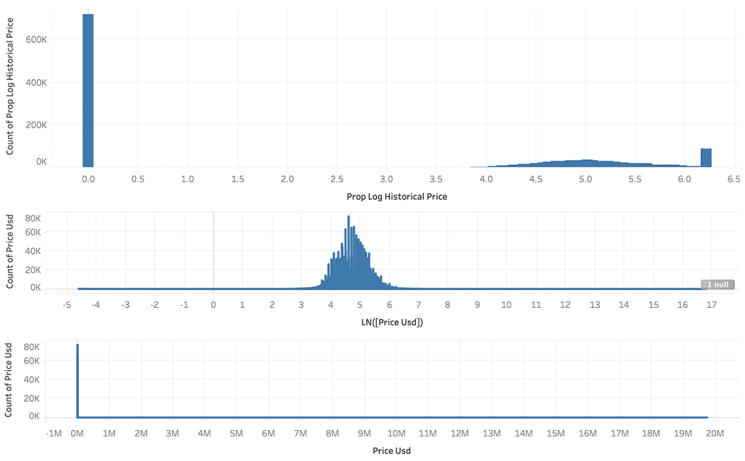

Fig. 2. Price distribution

classifier, to a certain extent. Due to the skewness, we applied log transformation on *price_usd* column before the min-max normalization. All the continuous variables are scaled to (-1,1).

In addition, we also found that several extreme values of price which are unrealistic or with huge difference compared with historical log price. Therefore, we chose to use 99.7% of data without these extreme values to fit our models.

Effects on clicks and booking We regarded this hotel recommendation challenge as a multi-label classification task, since we want to find both the hotels that users are most likely to click and the clicked hotels which are most likely to be booked at the same time. Consequently, we also tried to investigate the factors that influenced the booking intention of users after clicking.

According to the category of variables, we mainly focused on the variables with four different properties: Boolean variables, user search parameters, hotel quality measurements, and independent individual information (such as *site_id, prop_id*, etc).

With an average click-booking probability of 0.623, the first potential influencing factor that comes to mind is the price or the quality of hotel.

Through the result showed in Fig3 cannot conclude any significant effects exist within these attributes, though we did observe that hotels with cheaper price and better quality will be a little bit more likely to be booked, relatively.

Next, we turned our attention to the parameters of the user search (number of people, booking windows, length of stay) and some Boolean features (promotion, prop brand, randomly displayed..). We found from users' search results that the probability of a click converting into a booking became higher with fewer days and fewer people. This might be explained by "people are more likely to make decisions with fewer uncertainties".

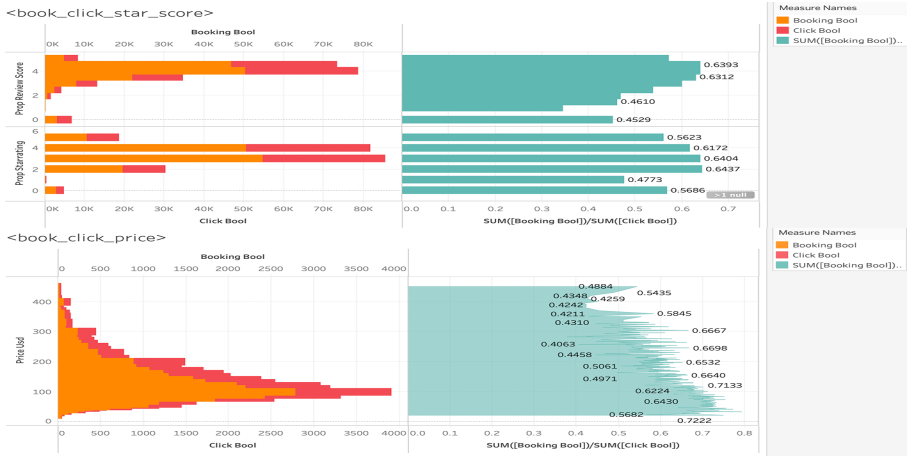

Fig. 3. click-booking under price and starrating

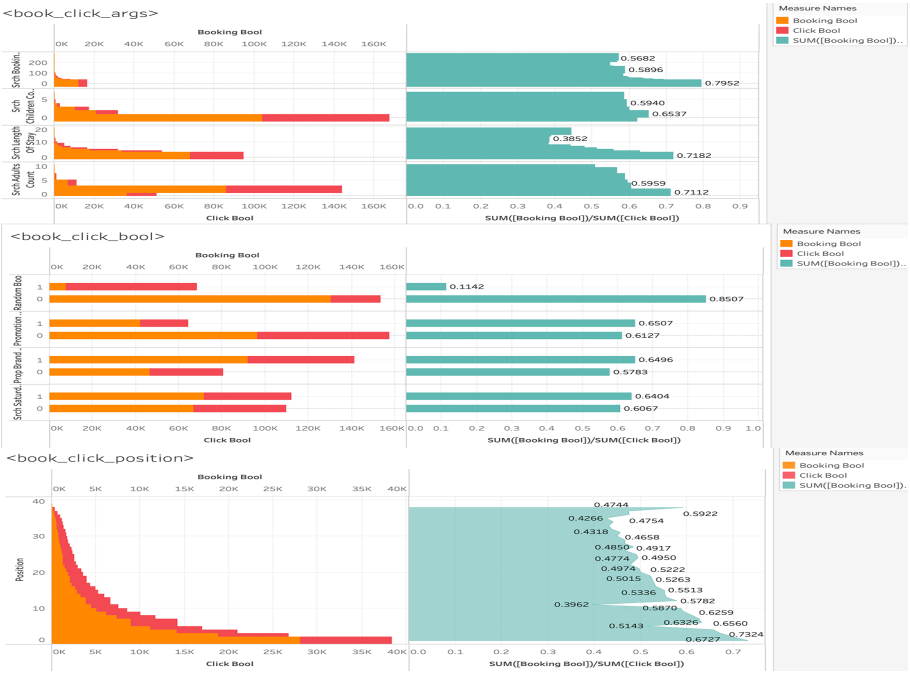

Fig. 4. click-booking under args and boolean attrs

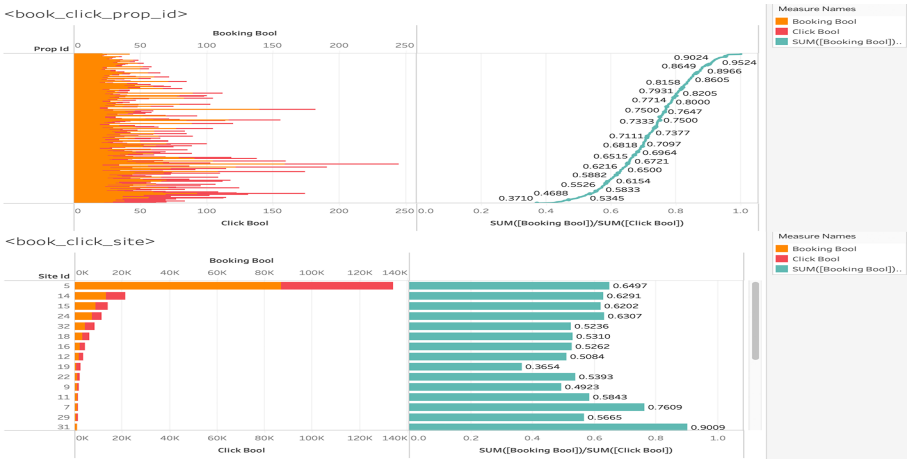

Fig. 5. click-booking under site$_i$*dandprop*$_i$*d*

Another interesting discovery was that without inserting promotional ads, meaning displayed in a normal order, the user's click-to-book conversion rate was surprisingly high with a probability of 0.85, compared to 0.11 in the artificial order. That made us interested in the effect of the hotel's presentation position on the booking result, so we then tested the *position* variable.

Unfortunately, the effect of the *position* variable was not as intuitive as we thought. It was only slightly greater than the average booking rate for the top three hotels, but the trend was that the later the hotel was positioned, the less likely it was to be booked. In the light of this result, we got the idea to characterize the variable *position*. Since it would not be available in the test set, we added the historical average position as a new feature for each hotel.

Finally, we performed the same examination for the two identity features *site_id and prop_id*. The result was that the id of sites and hotels had a significant effect on the booking. That is, for some particular hotels and websites, customers would have a stronger intention for booking. This finding likewise supported the idea that we did feature-columns operation for such variables, which was also a reason why we used deep learning models later.

3.4 Model selection

Logistic Regression Logistic regression is a robust machine learning techique which is capable of interpreting linear and nonlinear patterns and strong generalization ability[6]. It is commonly employed in binary classification problems. With linearly separable classes, it is fairly simple to implement and gives excellent results.[13]. It models a binary output variable using a sigmoid function. Compared with linear regression, logistic regression does not require a linear relationship between inputs and output variables[2]. The parameter of Logistic Regression is chosen by maximizing the conditional data likelihood[1] and can

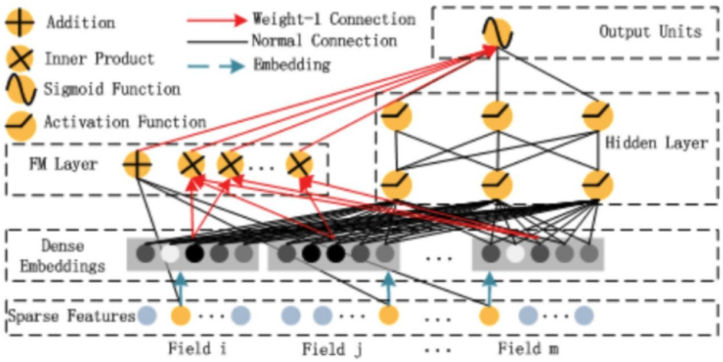

Fig. 6. Wide deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

be modified to handle categorical explanatory variables through definition of dummy variables[12].

The classic pointwise classification approach is used to tackle this problem using Logistic Regression[10]. To forecast the value of booking bool and clicking bool, two binary classification models based on logistic regression are created in this project. The scores of each property for each search is then computed using the formula $5 * bookingbool + 1 * clickbool$ and the search results are then sorted in reverse order based on the calculated scores.

DeepFM CTR estimation focuses on learning combined features. Note that the combined features include second-order, third-order, and even higher-order ones. The higher the order, the more complex it is and the harder it is to learn. Google's paper research concludes that both high- and low-order combined features are important, and that learning both combined features at the same time performs better than considering only one or the other.

In order to solve the problem of "how to efficiently extract these combined features", the deepfm model came out. It was proposed in 2017 and is an upgraded version of Google's widedeep recommendation model. It contains two parts: FM and DNN. The FM model can extract low-order features, and DNN can extract high-order features, so no manual feature engineering is required. In addition, since the input is only raw features, and the FM and DNN share the input, the training speed of the DeepFM model is relatively fast.

In our experiment, we set booling_bool as target. We used all the original features directly and normalized min/ Max only for all continuous variables. For those sparse variable features, our dense embedding is implemented, and then a FM layer is added. On the other side, all continuous variables and sparse variables after embedding are integrated into the full link model of three layers. Each layer is attached with RELU activation function, which becomes the Deep

layer. Finally, the results of FM layer and deep layer are output by a sigmoid softmax function as the prediction results

Lightgbm Lightgbm is a type of gradient boosting machine is an ensemble model of decision trees, which are trained in sequence. In each iteration, the algorithm learns the decision trees by looking at the residuals errors. Compared with XGboost, the trees of LightGBM grows differently: traditional framework trees grow per level, while the growth of LightGBM is focused on the leafs[8]. In addition, unlike typical tree models, LightGBM can directly address ranking problems, which is ideal for our needs in this project.

In this project, LambdaRank is selected as objective function, which has proved to be very effective on optimizing ranking functions such as nDCG. Compared with RankNet, LambdaRank writes down the desired gradients directly rather than deriving them from a cost. It makes LambdaRank able to bypass the difficulties introduced by the sort in most information retrieval measures[5].

4 Experiment

4.1 Sample generation

The original training set in the project was 1.3GB in size, with 4,958,347 records and 54 features. Using the entire dataset for model training would take longer based on the computational capacity we have. At the same time, the experimental comparison shows that there is no significant degradation in the training performance of the model on a smaller dataset. Furthermore, the dataset is significantly skewed when utilizing a pointwise approach to solve the problem. That is, the number of records with the booking bool or click bool of 1 is significantly less than those with booking bool or click bool of 0.

As a result, in the logistic regression data pre-processing step, we choose all positive samples and a random number of negative samples equal to the number of positive samples. Irrelevant features and outliers from are also eliminated from the data sample. The tree model, on the other hand, can better handle imbalance data[11], and we utilize the entire dataset for lightgbm to maximize the training impact.

4.2 Missing Values

Some features in the dataset contain a large proportion of null values, which are of little reference value. Therefore, for lightgbm we removed the features such as comp1_rate, gross_bookings_usd since they contain more than 90% null values. Because the tree model is able to handle null values[14], no further processing for missing values is required.

The logistic regression model, on the other hand, demands that the data set be free of null values. The logistic regression model is better trained when there are fewer features. Therefore, features containing more than 60% null values

are removed. Then we impute review score as the mean score for that hotel. For hotels with no rating info available, we impute them by the mean rating for hotels in that country. The prop_location_score2's missing values are replaced with $prop\_location\_score1/100$ since prop_location_score2 only has correlation with prop_location_score1 and location score is different for each hotel. Similarly, the orig_destination_distance is populated with the mean orig_destination_distance for that country. The remaining null values are removed.

4.3 Train,validation,test data set split

Because the test set is already present in the dataset, we select 10% of the data using for validation set and applied it to the lightgbm model. We take 30% of the original unbalanced data and utilize it as a validation set for Logistic Regression. To construct a training set, the remaining data is processed following the procedure described in section 4.1.

4.4 Parameter tuning

For lightgbm, the grid search is applied to obtain the most suitable parameter. The parameter chosen is illustrated in table 1. With eval_at set to 5, verbose set to 20, early stopping rounds set to 200, and category feature highlighted, the model is trained.

Parameters	value
objective	lambdarank
metric	ndcg
$n_estimators$	100
learning rate	0.12
max position	5
label gain	[0, 1, 2]
random state	69
seed	69
boosting	dart

Table 1. parameters of the lightgbm models

5 Result and discussion

5.1 Evaluation

The Normalized Discounted Cumulative Gain(NDCG)@5 is applied to evaluate the performance of the model. NDCG is often used as an evaluation metric

for ranking results to assess the accuracy of the ranking. Assume the list of goods recommended by the system for the user is K items long. The difference between this sorted list and the user's actual interaction list is evaluated using NDCG@K. The following is the NDCG@K's calculating formula:

$$NDCG_u@K = \frac{DCG_u@K}{IDCG_u}$$

$$NDCG@K = \frac{NDCG_u@K}{|u|}$$

where DCG is able to rate a user's list of recommendations, the DCG is normalized since the comparison of DCG between different users is meaningless due to the lengths differences the users' genuine lists. DCG's formula is as follows:

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

where $r(i)$ stands for the relevance score of each item in the list. In our cases, $r(i)$ equals to 5 when the user purchased a room at this hotel. $r(i)$ equals to 1 when the user clicked through to see more information on this hotel. Otherwise $r(i)$ is set as 0. What's more, in DCG, higher gain for top ranked item and discount for bottom ranked item.

5.2 Result

On the training set, the final ndcg@5 result for lightgbm is 0.461903. On the validation set, the final ndcg@5 result is 0.416418. We also implement a benchmark model provided by the Expedia, the result of using different models to rank test sets is shown in Table 2. The lightgbm algorithm offers substantial benefits over other approaches, as seen in the table. This is due to improved feature engineering and the fact that lightgbm considers both the booking and click bools in the target setting. Furthermore, unlike typical binary classification approaches, lightgbm employs ndcg as a judgment signal of training efficacy rather than classification accuracy, allowing the model to tackle the ranking problem more effectively. The logistic regression model also takes both booking bool and click bool into consideration and calculate the weighted score. However, compared to the tree model, it does not handle missing values and classify them as good as the tree model such as lightgbm.

5.3 Important features

Figure 7 shows the top 10 relevant features chosen by lightgbm, with the values representing the relevance of the features for the model. These characteristics are grouped into three categories. The first are attributes specific to a single hotel, such as prop country id and prop location score2, which demonstrate that a property's ranking results are highly impacted by its unique qualities. The second

Model	result
benchmark	0.25694
deepfm	0.36047
logistic regression	0.32027
lightgbm	0.40530

Table 2. The result comparison for different models

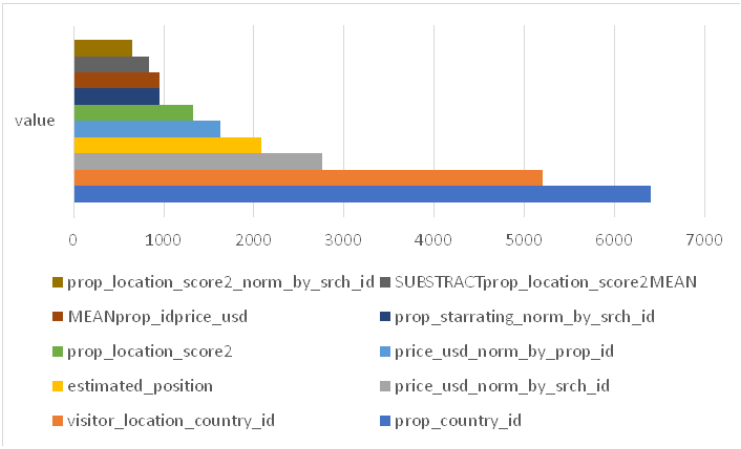

Fig. 7. Top 10 features selected by lightgbm

category includes characteristics such as visitor location country id and price usd norm by srch id that are connected to single searches. This demonstrates that the customers' origins and consumption patterns have a stronger impact on the ultimate decision. Seven of the top ten significant traits were derived using feature engineering and are categorised as normalized by destinations. This demonstrates that the feature engineering in this project has effectively refined the essential data.

6 Conclusion

In this project, we are dealing with a CTR estimation of addressing a multivariate data. Generally speaking, there are three mainstream solutions to this problem, pointwise, pairwise and listwise. The dataset contains information about how people behave when looking for a hotel. We altered the format and values of several variables after data analysis and provided structured multivariate data that the ML model could utilize. Simultaneously, we use feature engineering to change some of the data's features into more useful ones. As a result, we chose Logistic Regression to represent the traditional machine learning method used by point-wise approach, DeepFM to represent deep learning method and light-

gbm to represent pairwise machine learning method. To implement the Logistic Regression and lightgbm, we performed missing value interpolation, normalization, and train-test splits on the preprocessed data. The models are then trained using customized parameters.

For all three models, the ndcg is used as the evaluation metric. Lightgbm has the highest ndcg value on the test set, followed by DeepFM and Logistic Regression. It is shown that when dealing with recommender system issues with fewer characteristics and bigger data volumes, the direct pairwise technique is more successful than the pointwise approach. This is because the pairwise approach can directly use ndcg as an evaluation metric in training rather than the precision metric used in the pointwise approach for binary classification. At the same time, the tree model, as opposed to the standard logistic regression model, can better handle data with more categorical characteristics and null values. We will use the listwise technique in future development. Using the blending strategy to increase the final ndcg value and improving feature engineering to better utilize data

References

1. Hamid R Arabnia and Quoc Nam Tran. *Emerging trends in applications and infrastructures for computational biology, bioinformatics, and systems biology: systems and applications*. Morgan Kaufmann, 2016.
2. Hoss Belyadi and Alireza Haghighat. *Machine Learning Guide for Oil and Gas Using Python: A Step-by-Step Breakdown with Data, Algorithms, Codes, and Applications*. Gulf Professional Publishing, 2021.
3. Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.
4. Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with non-smooth cost functions. *Advances in neural information processing systems*, 19, 2006.
5. Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.
6. R Amaro e Silva, LCC Teixeira da Silva, and MC Brito. Support vector regression for spatio-temporal pv forecasting.
7. Paweł Jankiewicz, Liudmyla Kyrashchuk, Paweł Sienkowski, and Magdalena Wójcik. Boosting algorithms for a session-based, context-aware recommender system in an online travel domain. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, pages 1–5, 2019.
8. Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.
9. Hang Li. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011.
10. Hongzhi Liu, Zhonghai Wu, and Xing Zhang. Cplr: Collaborative pairwise learning to rank for personalized recommendation. *Knowledge-Based Systems*, 148:31–40, 2018.

11. Ya-Qin Liu, Cheng Wang, and Lu Zhang. Decision tree based predictive models for breast cancer survivability on imbalanced data. In *2009 3rd international conference on bioinformatics and biomedical engineering*, pages 1–4. IEEE, 2009.
12. Robert A Meyers. *Encyclopedia of physical science and technology*. Academic, 2002.
13. Abdulhamit Subasi. *Practical Machine Learning for Data Analysis Using Python*. Academic Press, 2020.
14. Masahiro Sugimoto, Masahiro Takada, and Masakazu Toi. Comparison of robustness against missing values of alternative decision tree and multiple logistic regression for predicting clinical data in primary breast cancer. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3054–3057. IEEE, 2013.
15. Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 1313–1322, 2018.
16. Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.
17. Yanwu Yang and Panyu Zhai. Click-through rate prediction in online advertising: A literature review. *Information Processing & Management*, 59(2):102853, 2022.