Data mining assignment 2 report Group 15

Zhe Liu[1], Bowen Liang[2], and Junyi Ping[3]

[1] Vrije Universiteit Amsterdam, 2756066, z.liu4@student.vu.nl
[2] Vrije Universiteit Amsterdam, 2759388, b.liang@student.vu.nl
[3] Vrije Universiteit Amsterdam, 2771984, j.ping@student.vu.nl

Abstract. Recommender systems are used to display search results sorted by their relevance to the user. In this study, we employed various algorithms to solve the learning-to-rank problem using the Personalize Expedia Hotel Searches dataset from 2013. Through rigorous feature engineering and data preprocessing, as well as continuous improvement and iteration of the models, we achieved our best results using LGBMRanker on the publicly available competition test data, with an NDCG@5 score of 0.41120.

Keywords: Data Mining, LightGBM, Collaborative Filtering, KNN, Recommender Systems

1 Introduction

Expedia is a hotel booking service and online travel agency (OTA) where users can input search queries containing information such as desired booking dates, number of people, destination city, and more. They are then presented with a sequence of hotel recommendations. In this project, we explore, perform feature engineering, and model the Expedia dataset to design a recommender system that effectively ranks search results based on the likelihood of being booked (best-case scenario) or clicked (average case). We employ LGBMRanker and collaborative filtering models for implementation and evaluate their performance.

Our formatting in this presentation differs from previous ones. Chapter 4 presents the initial implementation of LGBMRanker, which did not perform optimally. In Chapter 5, we further improved the model introduced in Chapter 3, achieving the best results among our team in this competition. Chapter 6 focuses on the implementation of a collaborative filtering model. Through these chapters, we aim to show our iterative process of gaining a deeper understanding of the competition data and conducting experiments.

2 Business Understanding

In order to enhance the predictive accuracy of a prioritized list of recommended hotel properties based on users' likelihood to book or click on them, we conducted a comprehensive analysis of relevant projects to ascertain optimal feature extraction strategies. Given that the "Expedia Hotel Recommendation" contest was conducted on the Kaggle platform in 2013, we chose to utilize a top-performing solution as a reference point for our study.

The reference solution we used achieved fifth place in the competition. Based on the attached research paper[1] detailing its methodology, we have summarized the following key insights:

First, they evaluated the importance of each feature to other features. They found there are the most three important listwise ranking features: price_usd, prop_starrating, prop_location_score2. The listwise feature serves as a conduit for incorporating listwise information into point/pair models.

Then, they started to process data and composite some features. The missing values of some features are filled in with the first quartile calculated by the country. were consolidated into a single composite feature. For instance, *window_count = srch_room_count * max(srch_room_window) + srch_room_window*.

Subsequently, they split the data into the training dataset and testing dataset. They also point out that the training dataset is randomly sampled 10% from the original dataset. There are small differences between the model trained with the small dataset and the model trained with the total training dataset. In addition, they also used balanced data for training random forest and deep neural network. In terms of the training tree-based model, they split data by country to reduce training time.

Finally, they used Bucket to binarize float features, which allows the float features to be in smaller variance. The models they used are various, including Logistic Regression, Random Forest, Gradient Boosting Machine, Factorization Machine, LambdaMART and Deep Learning Approach. The random forest has the best performance.

In summary, these insights from the prediction of this team will guide our approach to the current task. The importance of feature engineering, the use of multiple modelling techniques, and the consideration of both user behaviour data and hotel properties will inform our strategy for predicting which hotel properties a user is most likely to click on or book.

3 Data Understanding

3.1 Dataset overview

The dataset for this competition is provided by Expedia and consists of two files: a training set and a test set. Based on the description of the dataset, we can categorize all the attributes into the following groups:

1. **User search criteria**: These are the attributes that users can see and choose when they search on the Expedia website. (Eg. search date and time, destination ID, length of stay, booking window, number of rooms etc.)
2. **Static Hotel Characteristics**: These refer to the unchanging attributes of a hotel, irrespective of user search queries. (Eg. hotel ID, star rating, user review score etc.)
3. **Dynamic Hotel Characteristics**: These refer to the hotel attributes that can vary during the search process based on user preferences or time. (Eg. rank position, price, promotion flag etc.)
4. **Visitor Information and purchase history**: These attributes include detailed information about the visitor as well as aggregated information about their past purchases. (Eg. visitor country, distance between visitor and hotel, mean price per night purchased by visitor etc.)
5. **Competitor comparison information**: These provide insights into the pricing and competitiveness of Expedia compared to other travel agency websites for the same hotels. (Eg. price comparison, availability comparison, percentage price difference etc.)

In addition, the training set contains some attributes that are not present in the test set, such as user click and booking behaviour data. The training dataset is massive, with 4,958,347 rows and 54 columns, including 199,795 unique searches and 129,113 distinct hotels. And it is imbalanced, with a large number of hotels not being clicked or booked.

3.2 Data Quality Analysis

Missing values After examining the composition of the dataset, we explored the proportion of missing values in the training and test sets. **Fig 1** illustrates the percentage of missing values in each column for both datasets, revealing a striking similarity in the distribution of missing values. Furthermore, both sets contain numerous columns with a high proportion of missing values (e.g., the "*visitor_hist_starrating*" column has missing values exceeding 90% in both sets), which could potentially impact the predictive performance of the models. Thresholds need to be set in subsequent steps to determine which columns to select or remove.

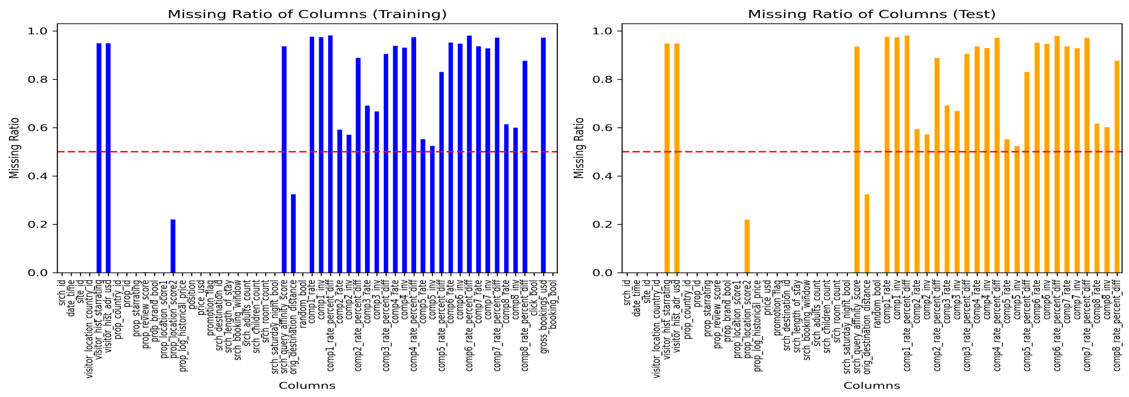


Fig. 1: Missing Ratio of Training Data and Test Data

Data correlations The purpose of the data correlation analysis is to examine the relationships between different attributes in the dataset and identify their linear relationships and mutual influences. This analysis is helpful for feature selection, feature engineering, and variable optimization in the process of model construction. Due to the large number of attributes, we selected the top 20 most correlated attributes for display. As shown in **Fig 2**, we observed the presence of columns with a high ratio of missing values(eg., the "*visitor_hist_starrating*" column). In subsequent feature engineering steps, we may consider dropping these columns.

Data Exploration After gaining a certain understanding of the dataset, we conducted exploratory analysis on some features. From **Fig 3** we can see that, the average prices of hotels exhibit peaks on certain dates, which may be attributed to those dates being holidays in certain countries. And as time progresses, more orders are being made. It could imply factors such as growing demand, expanding customer base, or improved marketing strategies leading to increased sales.

Continuing with our exploration, we examined the relationship between location score and star rating. From **Fig 4** it can be observed that as the star rating increases, the values of both scores also

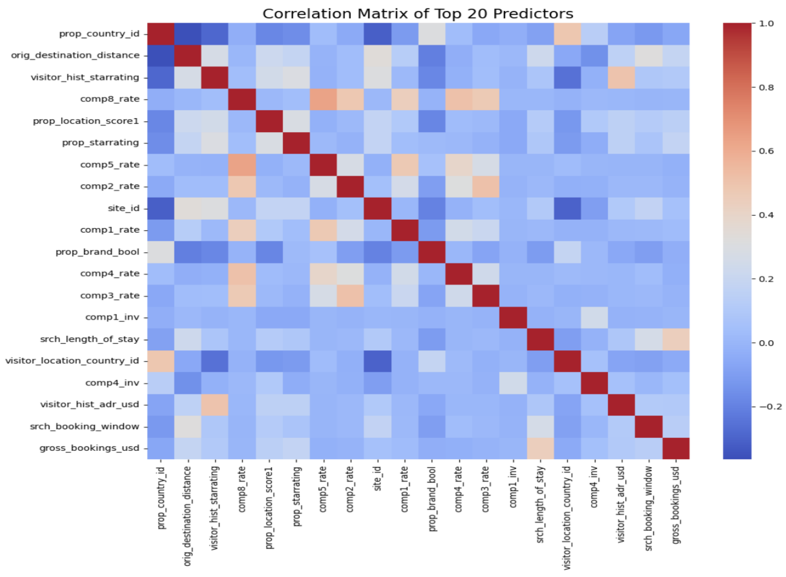


Fig. 2: Correlation Matrix of Top 20 Correlated Predictors

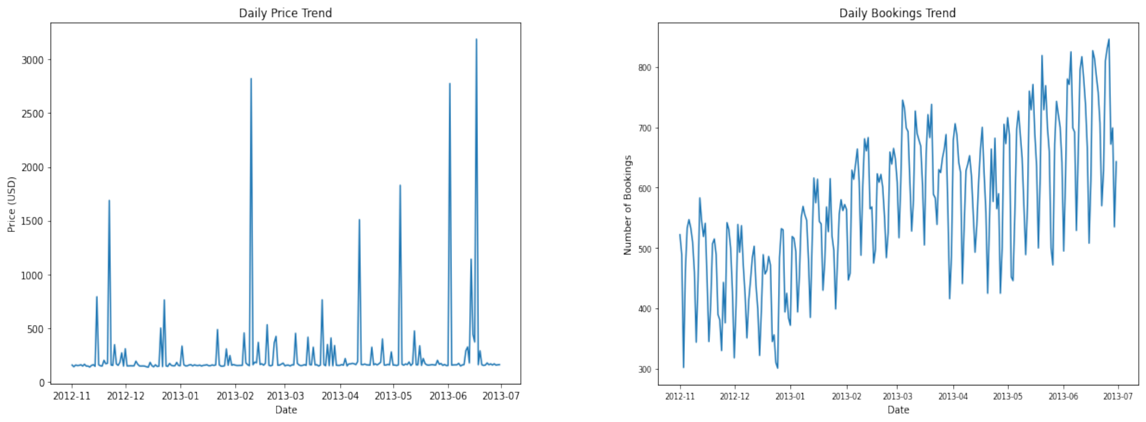


Fig. 3: Trend of data over time

show an increasing trend. Therefore, we can speculate that there is a positive correlation between the two attributes.

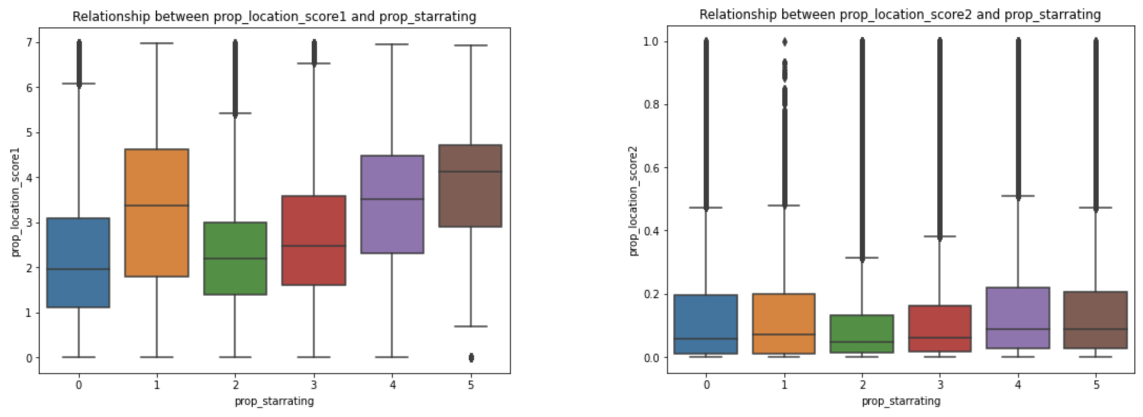


Fig. 4: Relation between location scores and star rating

4 LGBMRanker model version 1

4.1 Data preperation

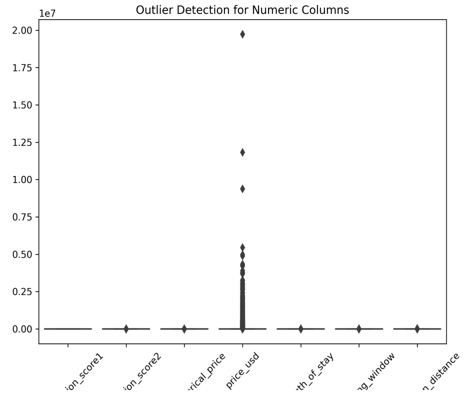
Outlier detection Firstly, we calculate the number of records and unique values for each column. Then, we plot histograms to visualize the distributions of the numerical variables. Based on the histogram results, we employ the **Z-score** method to detect and handle outliers in the numerical columns. The outliers are replaced with the median value of each respective column. From **Fig 5**, it can be observed that we have successfully dealt with the outliers.

Missing value imputation Upon observing the data, we noticed a significant number of missing values within the dataset. We set the threshold for missing values to 0.9 and remove columns that exceed this threshold in terms of missing values. For randomly missing numerical columns, we employed **mean** imputation, while for categorical features, we utilized **mode** imputation.

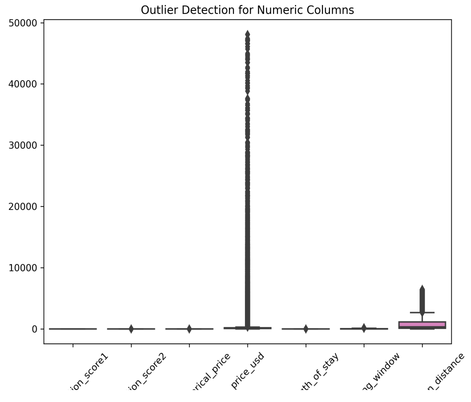
4.2 Feature Engineering

In the initial version of implementation, the following feature engineering were constructed. As shown in **Table 1**, the construction of the first two features is based on user expectations. We assume that the probability of a hotel being clicked or booked is closely related to the difference between the user's expectations and the hotel's attributes.

The construction of the remaining features is based on user behavior. When users perform a search query, they tend to prefer hotels that are cheaper or closer to their expected price. Additionally, when hotels have similar prices, people take into consideration factors such as higher star



(a) Detection before outlier handling



(b) Detection after outlier handling

Fig. 5: Outlier Detection for Numeric Columns

ratings, better review scores, and better geographical locations. Therefore, we have created several new features, including "*price_diff*," to capture these aspects.

New Features	Description
price_hist_diff	The user's payment records − the hotel's current price
star_hist_diff	The user's star rating records − the hotel's current star rating
price_diff	The hotel's price − the mean price of the hotels shown in the query
star_diff	The hotel's star rating − the mean rating of the hotels shown in the query
loc_score1_diff	The hotel's locaton score 1 − the mean socre 1 of the hotels shown in the query
loc_score2_diff	The hotel's locaton score 2 − the mean socre 2 of the hotels shown in the query
review_score_diff	The hotel's review score − the mean socre of the hotels shown in the query

Table 1: Feature Engineering

As shown in **Fig 6**, We plotted the booking probabilities of all user searches by month and identified the top five countries with the highest search frequency. Then, we plotted the booking probability trends for the country with the highest search frequency (ID: 219) compared to the remaining countries. It can be observed that the trend of the curve for country ID 219 closely resembles the overall curve for all users, while the curves for the remaining countries are not entirely similar. The irregular search peaks in 2022−11 and 2013−02, leading to actual bookings, can be attributed to tourists from the country with ID 219. Based on my findings, the month of the year serves as a significant indicator of tourist travel behavior. Therefore, I will reduce the "*date_time*" feature to its monthly component. It is necessary to split the dates, and as a result, we will transform the "*date_time*" column into four features: "***year***," "***month***," "***dayofweek***," and "***hour***."

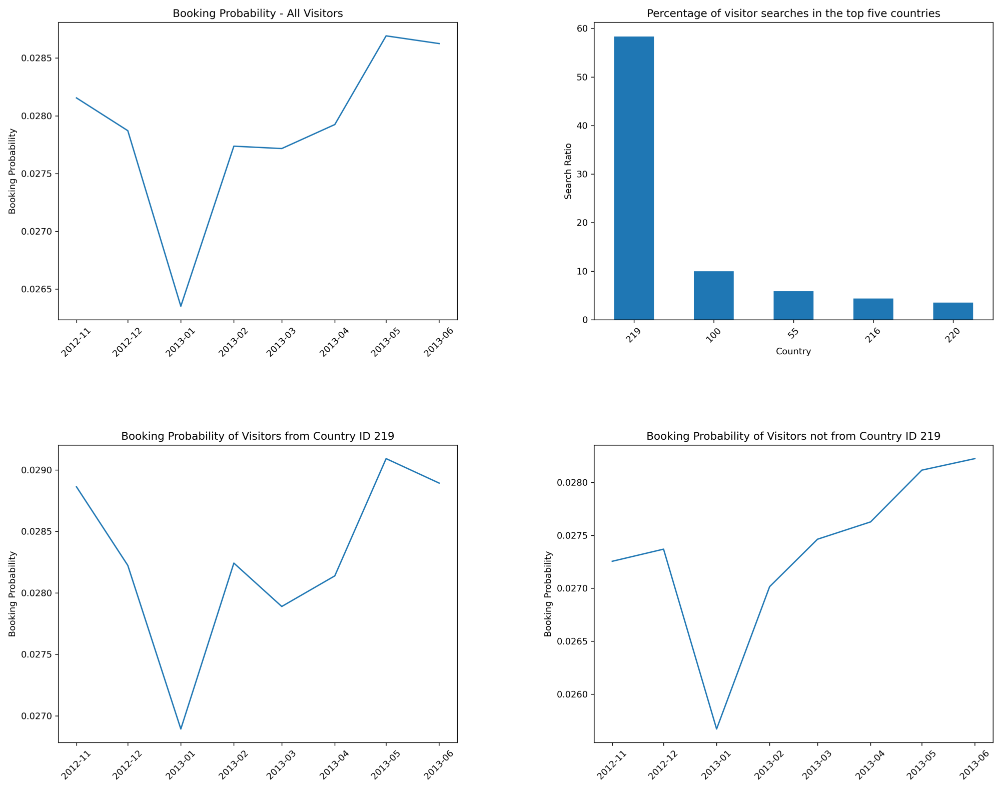


Fig. 6: Figure with four subfigures

4.3 Modeling and Evaluation

Model **LightGBM** is a gradient boosting framework developed by Microsoft. It is designed to efficiently handle large-scale datasets and provides high-performance machine learning algorithms. LightGBM is known for its speed, accuracy, and ability to handle categorical features effectively, making it a popular choice for various machine learning tasks.

We opted for the **LightGBMRanker** algorithm from Microsoft's LGBM framework for its suitability in our task and its built-in NDCG scoring method. LightGBMRanker incorporates LambdaRank, which transforms it into a learning-to-rank approach. By utilizing the gradients of the cost function with respect to the model scores (denoted as λ), LambdaRank adjusts the rankings of item pairs. This adjustment involves applying equal and opposite "forces" to the ranks of each item, pushing one item up by $|\lambda|$ while simultaneously applying an equal and opposite force to the other item.

Hyperparameter We used **GridSearchCV** from the Scikit-Learn library to search the hyperparameter space. The parameter grid we used is shown in **Table 2**. The best hyperparameter instances for our purposes, and parameters we used are shown in **Table 3**.

n_estimators	[100, 500, 1000
learning_rate	[0.01, 0.1, 0.15]
max_depth	[5, 7, 9]

Table 2: Hyperparameter

n_estimators	1000
learning_rate	0.1
max_depth	9

Table 3: Parameter setting

Results Paired with our feature engineering, and incremental adjustments to the learning rate and number of estimators, we were able to achieve ***NDCG@5*** = 0.30177 on the public task data. The final prediction results were not that good, so we reflected on our model and made further improvements.

4.4 Reflection

We have reflected on the poor performance of the model and upon revisiting the dataset, we discovered the presence of an attribute called "*random_bool*," which indicates whether the search results are in random order or in normal sequential order. As shown in **Fig 7**, there is a positional bias in the search results, as people are more likely to click on or book hotels that are ranked higher in the list, even if the ranking is randomized. We did not take this into account during the implementation, which could be one of the reasons for the underperformance of our model.

In addition, we have realized that the current constructed features are not enough, which may led to poor prediction performance. Furthermore, we have not applied data normalization, which may also affect the model's predictive ability. Therefore, we plan to further process the data, improve the feature engineering techniques, and enhance the model accordingly.

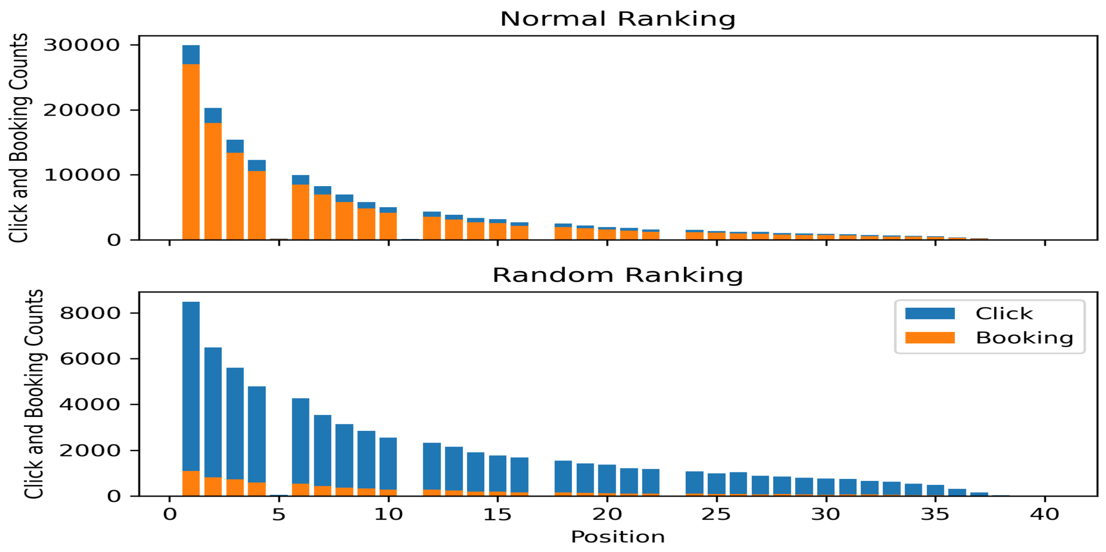


Fig. 7: Click and Booking counts of Random and Normal Ranking

5 LGBMRanker model version 2

5.1 Feature Engineering

Taking lessons from previous experiences, we retained the date-related features and performed reorganization and improvement on other features. The process of constructing feature engineering is as follows:

1. **Feature normalization**: Normalize feature columns such as "*price_usd*", "*prop_starrating*", "*prop_location_score2*", "*prop_location_score1*", "*prop_review_score*" etc.
2. **Target value assignment**: Set the target value column based on the values of the "*click_bool*" and "*booking_bool*" columns. Assign 1 for a click, 2 for a booking, and 0 for neither a click nor a booking. This column is used for model training.
3. **Aggregation operations**: Perform aggregation operations based on "*prop_id*," "*srch_id*," and "*srch_destination_id*." Calculate the mean values of "*price_usd*," "*prop_starrating*," "*prop_location _score2*," "*prop_location_score1*," "*prop_review_score*," and "*promotion_flag*." Use the calculated mean values for subtraction feature transformation.

4. **Mean position calculation**: groups the hotels in the search results by "*srch_destination_id*" and "*prop_id*", and calculates the average value of position for each group. It then adds the estimated position information for each combination of search destination and hotel to the training data.

5.2 Modeling and Evaluation

Model We continued using the **LGBMRanker** model and performed a simple split of the dataset into training and validation sets.

Hyperparameter We used **GridSearchCV** from the Scikit-Learn library to search the hyperparameter space. The following parameter grid was used:

n_estimators	[100, 500, 1000, 2000, 5000]
learning_rate	[0.01, 0.1, 0.2]
max_depth	[5, 7, 9, 11]

Table 4: Hyperparameter

The best hyperparameter instances for our purposes, and therefore the ones we used for the Kaggle competition were:

n_estimators	2000
learning_rate	0.1
max_depth	11

Table 5: Parameter setting

Results Paired with our feature engineering, and hyperparameter tuning through GridSearchCV, we were able to achieve ***NDCG@5*** = 0.40754 on the public task data and ***NDCG@5*** = 0.40742 on our validation. This is the **best** performance achieved among all our submissions, indicating that the improvements made in our second version were highly effective.

6 Collaborative Filtering model

6.1 Key Idea

Collaborative filtering is a widely used technique for recommendation systems and KNN is a non-parametric algorithm that operates on the principle of similarity. Leveraging this characteristic of KNN, we integrated these two algorithms together to improve overall system performance. In our implementation, collaborative filtering is user based and the measuring method for calculating the similarity between two items is cosine similarity. The KNN algorithm we adopted is unsupervised, which can automatically cluster similar hotel properties.

6.2 Data Preprocessing and Feature Engineering

The data processing steps of Collaborative Filtering are similar to LightGBM. However, in the stage of feature engineering, we created a feature "*window_count*" by combining "*srch_room_count*" and "*srch_room_window*", as described in [1]. In addition, to reduce the calculation, the features of price comparison with other websites are integrated into one feature "*is_cheapest*" to show whether "Expedia" has the cheapest room. Finally, we removed the features has less relation to "*booking_bool*" or "*click_bool*" to reduce the disturbance from noise.

6.3 Implementation

The specific implementation is that we use KNN to select the top five similar search records in the training dataset for each unique search based on search properties, such as "*srch_saturday_night_bool*", in the test dataset in advance, and these top similar search records are used as a baseline to find hotels with similar properties in collaborative filtering. The basic workflow is displayed in

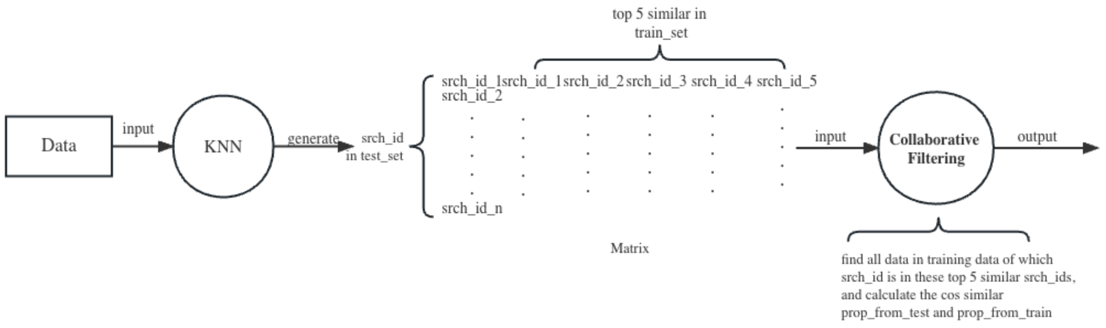

top 5 similar in train_set
srch_id_1 srch_id_1 srch_id_2 srch_id_3 srch_id_4 srch_id_5
srch_id_2
srch_id_n
Data
input
KNN
generate
srch_id in test_set
Matrix
input
Collaborative Filtering
output
find all data in training data of which srch_id is in these top 5 similar srch_ids, and calculate the cos similar prop_from_test and prop_from_train

Fig. 8: The basic workflow of the Collaborative Filtering with KNN system. The Collaborative Filtering is responsible for finding all data in training data of which *srch_id* is in these top 5 similar *srch_ids*, and calculate the cos similar *prop_from_test* and *prop_from_train*.

6.4 Results and Evaluation

To test the performance of Collaborative filtering with KNN, we conducted four training sessions and validations. The dataset for each training and validation is sampled 1/1000 from the total training set. According to the [1] and mentioned in Section 2, there are small differences between the model trained with the small dataset and the model trained with the total training dataset. However, the result is not as good as LightGBM. The results of the four times validation are 0.1399, 0.1612, 0.1434 and 0.1974.

We raised two reasons for the lower score:

1. The training and validation dataset size is limited in size, resulting in a limited presence of similar search records within the dataset. However, based on the assumption of a uniformly distributed training dataset, we contend that even with a larger dataset, collaborative filtering with KNN would not achieve comparable performance to LightGBM.
2. The features for KNN to find the top 5 nearest neighbours are not enough. Deleting too many features in the stage of feature engineering, caused the inadequacy of the available features for KNN to cluster. Therefore, the accurate and effective clustering performance of KNN and the calculation of collaborative filtering are affected.

7 Overall results

The overall results are displayed in Table 6.

Model	Training	Testing	Validation	Public
LightGBMRanker	0.48809	0.43763	0.40742	0.40754
Collaborative Filtering	0.1399	0.1505	0.1434	0.1604

Table 6: Overall results

8 Deployment

By leveraging the methods introduced in the big data engineering and big data infrastructures lectures, Expedia could deploy our collaborative filtering and GBM models that can handle large amounts of data and adapt to changing data characteristics in a scalable way below.

Deployment of LightGBM LightGBM is a gradient-boosting framework using tree-based learning algorithms, which is well suited for handling big data. Because it is designed to be more efficient than traditional gradient-boosting decision trees (GBDT), it can efficiently process large-scale datasets. Likewise, it supports parallel training on multiple GPUs/CPUs[3]. In addition, it uses advanced features such as GPU acceleration, automatic parallelization, large-scale tree-boosting support, and efficient memory usage to optimize the program. Ideal implementations suitable for LightGBM setups include predictive modelling and feature selection or engineering tasks.

Deployment of Collaborative Filtering Collaborative filtering can be scaled using distributed computing frameworks like Apache Spark, which has a built-in collaborative filtering algorithm in its machine learning library. This algorithm can handle large amounts of user-item interactions distributed across multiple machines. Even though, in our implementation, the performance of Collaborative Filtering with KNN is not ideal. However, there are many alternative algorithms to replace KNN to work with Collaborative Filtering. The paper [2] presents various correction techniques, including RMSE, BRISMF and so on, that can lead to significant accuracy improvements and shows that they compare well with current methods in terms of prediction accuracy measured in terms of RMSE and time complexity. RMSE and time complexity. Experiments demonstrate that the proposed method can be extended to a large scale with hundreds of millions of messages recommender systems with hundreds of millions of ratings.

Continuous Learning and Adaptation Both collaborative filtering and LightGBM models can be updated as new data comes in. For collaborative filtering, this could involve updating the user-item interaction matrix and recalculating similarity scores. For LightGBM, this could involve retraining the model with the new data.

Monitoring and Evaluation Once the models are deployed, it's important to continuously monitor their performance and evaluate how well they're meeting their objectives. This could involve tracking metrics like precision, recall, or click-through rate, and setting up alerts to notify you if these metrics fall below a certain threshold[4].

Conclusion In conclusion, LightGBM exhibits strong compatibility with large-scale datasets, while collaborative filtering, in conjunction with supplementary methodologies, demonstrates efficacy in managing such extensive data. By drawing inspiration from the successful implementation of LightGBM and collaborative filtering techniques at Netflix, we posit that these methodologies possess considerable potential for application within the realm of "Expedia."

9 What we learned

Zhe: First of all, through this assignment, I have a deeper understanding of the recommendation system and its related algorithms and a more intuitive understanding of its recommendation principles. Secondly, through the collaborative filtering algorithm, I realized that it is important to combine, delete, and decompose the features in a reasonable way in the stage of feature engineering, and it is not wise to keep only the features with high relevance. Also, by reviewing how to apply collaborative filtering to big data, I learned that collaborative filtering can be combined with many other algorithms to improve performance. Finally, the optimization of data parameters and model optimization is a delicate task that requires patience.

Bowen: This project has taught me how to approach a seemingly large and complex dataset and gradually become familiar with it. Once we found a promising algorithm, feature engineering became a crucial part of the challenge, making it an unforgettable experience for me. Through continuous refinement of feature engineering, exploring different data transformations and aggregations, and iteratively optimizing the model, I have greatly improved my coding skills. I have experienced moments of despair, such as when the model performed poorly or issues arose with feature engineering, but also moments of joy when the adjusted model showed improvement and I became more proficient in handling the data. As the project progressed, the importance of writing clean code and maintaining proper documentation became increasingly evident within the team. This experience will undoubtedly change the way I approach my next project. I have gained a lot from this course and I am grateful to have been a part of such an excellent course!

Junyi: Through this project, I gained a deeper understanding of applied machine learning, particularly the importance of feature engineering and techniques for handling big data. I also encountered ranking problems for the first time, realizing that using ranking-related models can enhance precision and speed for certain issues. This experience has deepened my understanding of how to apply machine learning theory to practical problems.

References

1. Liu, Xudong & Xu, Bing & Yuyu, Zhang & Yan, Qiang & Pang, Liang & Li, Qiang & Sun, Hanxiao & Wang, Bin. (2013). Combination of Diverse Ranking Models for Personalized Expedia Hotel Searches.
2. Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. The Journal of Machine Learning Research, 10, 623-656.
3. Qi Meng, Guolin Ke, Taifeng Wang, Wei Chen, Qiwei Ye, Zhi-Ming Ma, and Tieyan Liu. A communication-efficient parallel algorithm for decision tree. In Advances in Neural Information Processing Systems, pages 1271–1279, 2016.
4. R. Hu, W. Dou and J. Liu, "ClubCF: A Clustering-Based Collaborative Filtering Approach for Big Data Application," in IEEE Transactions on Emerging Topics in Computing, vol. 2, no. 3, pp. 302-313, Sept. 2014, doi: 10.1109/TETC.2014.2310485.