Data Mining Techniques Assignment 2
Group 77

Pengju Ma[2764123], Yue Zhang[2772421], and Zhuofan Mei[2757904]

Vrije Universiteit Amsterdam
{p.ma2,y.zhang,z.mei}@student.vu.nl

Abstract. In this work, we apply machine learning strategies and hotel recommendation algorithms to produce lists of hotels that are highly recommended based on information gathered from Expedia users' online activity. Several machine learning techniques are evaluated and their performance is benchmarked, and a description of the data exploration, data pre-processing, analysis, feature engineering, and model selection methods is also provided.

Keywords: Machine Learning · Recommendation System · Ranking Algorithm

1 Introduction

Introduction: In this assignment, our task is to develop a recommender system to predict which hotel properties a user is most likely to click on based on their search query. This prediction can greatly assist companies like Expedia in organizing search results and providing users with the most relevant and suitable options.

Expedia is an online travel agency (OTA) and hotel booking service that has become increasingly popular in the last two decades. Along with other booking websites such as Trivago and Booking.com, Expedia allows users to explore and compare hotel options through online search. These platforms have largely replaced traditional brick-and-mortar travel agencies and now account for a significant portion of global travel sales.

For OTAs like Expedia, the quality of service is crucial. When a search query generates relevant and appealing hotel results, the chances of securing a booking increase. In order to improve the organization and sorting of search results, Expedia organized the "Personalize Expedia Hotel Searches" competition on Kaggle in 2013. Participants were provided with a dataset containing search queries, hotel results, and purchasing information. The goal was to build a model that could predict the likelihood of a user booking a particular hotel based on their search query, enabling Expedia to sort search results more effectively.

In this report, we present our solution to the Kaggle competition. We approach the problem as a learning-to-rank (LTR) problem, where the objective is to rank hotels in search results based on their predicted likelihood of being

booked. To achieve this, we conducted a literature review and performed data analysis to identify relevant features from the competition dataset. Subsequently, we trained a LambdaMART model on this data. The report provides detailed insights into our process, including data analysis, feature engineering, modeling, evaluation, and reflections on the results obtained.

The dataset used in this assignment originates from Expedia itself. It contains information about user search queries, along with a list of hotels that were seen in a session. By analyzing this data and building a recommender system, we aim to improve the search experience for users on the Expedia website, ultimately increasing clicks and bookings.

2 Business Understanding

The primary objective of this task is to understand the business implications of the ranking problem in the context of hotel bookings. Ranking problems have significant applications in various industries, including search engines and recommender systems. In particular, recommender systems are crucial in sectors such as retail (e-commerce platforms) and media (content recommendation, advertising). Improving the performance of recommender systems can have a substantial impact on company profits by enhancing the customer experience and increasing the likelihood of purchases.

Companies like Netflix invest millions of dollars in research and development to provide more relevant content recommendations to their users, aiming to boost revenue. Expedia, as a prominent player in the travel industry, recognizes the importance of improving the hotel sorting order to enhance the customer experience. By offering more relevant recommendations, Expedia increases the chances of users making a purchase on their website and choosing the suggested hotel. Additionally, Expedia generates revenue by displaying promoted hotels, further emphasizing the significance of accurate sorting.

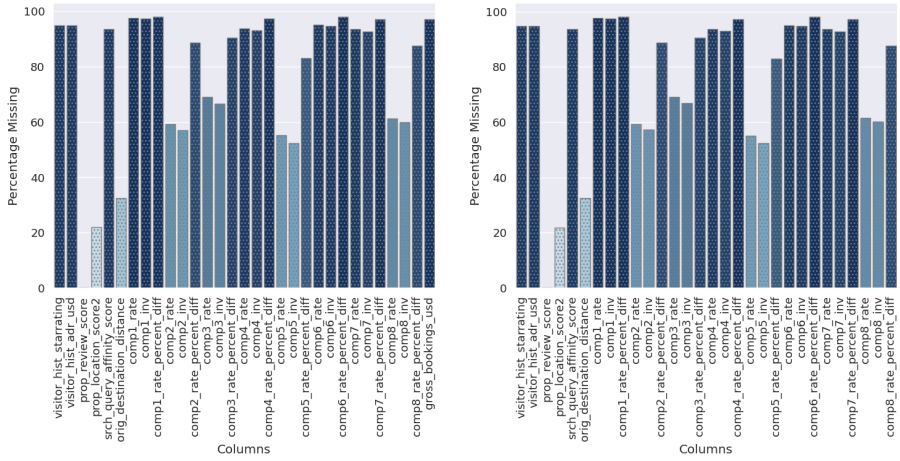
To address this business challenge, a competition was conducted to analyze and enhance the customer experience by improving the sort order of hotels. The ultimate goal was to provide more relevant recommendations to users, increasing the probability of successful bookings and customer satisfaction. The competition aimed to optimize ranking metrics such as Normalized Discounted Cumulative Gain (NDCG) or Mean Average Precision (MAP) through the use of ranking algorithms.

Previous iterations of the competition have demonstrated that **gradient boosting algorithms**, such as ensemble models and LambdaMART (a specific instance of gradient-boosted regression trees), achieved the best results in terms of ranking accuracy. These algorithms have proven to be highly effective in handling tabular data, making them a popular choice on platforms like Kaggle.

3 Data Understanding

3.1 Dataset Statistic

The dataset provided contained key information and attributes about the user, the hotel, the search query, as well as the competitor hotels that appeared in Expedia, which is shown in Table 3.1. And it had been divided into a training and a test set, with the training data containing 54 different attributes (`position`, `click_bool`, `booking_bool`, and `gross_booking_usd` were added), while the test data contained 50 different features. For both training and test data, only one type of variable is categorical, the rest are numerical. Figures 1(a) and 1(b) display the proportion of the missing values that exist in the columns of the training and test dataset. There are a large number of missing values in the columns containing all Competitors Info and in the few columns containing hotel and tourist information, so we intend to ignore columns with a large number of missing values (e.g., more than 80-90% of missing values), as these columns with too many missing values may not be helpful for our subsequent work.



(a) Missing value in train set (b) Missing value in test set

Fig. 1.

3.2 Exploratory Data Analysis

There are nearly 4.9 million observations in total, and the statistic related to the `srch_id` and `prop_id` is shown in Tables 3.2, which shows that almost every search query would generate a click action, and every hotel had been booked at least once. And, in order to improve the efficiency of statistics, we did not select all the variables, but only selected some of them to perform Pearson correlation statistics, and the results are shown in the heatmap in Figure 2. It shows that there is a strong relation between the variable `click_bool` and `booking_bool`,

Category	Variable
Visitor Info	visitor_location_country_id
	visitor_hist_starrating
	visitor_hist_adr_usd
	click_bool (*training*)
	booking_bool (*training*)
Hotel Info	prop_country_id
	prop_id
	prop_starrating
	prop_review_score
	prop_brand_bool
	prop_location_score
	price_usd
	prop_log_historical_price
	promotion_flag
	position (*training*)
	gross_booking_usd (*training*)
Query Info	srch_id
	srch_destination_id
	srch_length_of_stay
	srch_booking_window
	srch_adults/child/room_count
	srch_saturday_night_bool
	srch_query_affinity_score
	random_bool
Competitors Info	comp1_rate/inv
	...
	comp8_rate/inv

Table 1. The categories of attributes

and a medium correlation exists between users' country and hotels, as well as booking fee and length of stay.

Besides, we counted the changes in total user clicks and bookings over time, as shown in Figure 3. The number of clicks and bookings show similar trends, both undergoing a turbulent process of change, but generally increasing over time, which may indicate that users browse more on Expedia during peak travel seasons or holidays, and vice versa. Additionally, ignoring the effect of the `random_bool` variable, the clicking and booking probabilities of the relevant hotels in the different `position` were counted, and the results are shown in Figures 4(a) and 4(b). In general, the higher the `position` is, the higher the clicking and booking probabilities are in relative terms, but surprisingly, in terms of booking, the probability of the last `position` is much higher than its preceding `positions`. These findings would be helpful for our further modelling process.

srch_id	value
Total	199795
Avg Book	0.692
Avg Click	1.111

Table 2. srch_id

prop_id	value
Total	129113
Avg Book	1.072
Avg Price	276.557
Std	7254.486

Table 3. prop_id

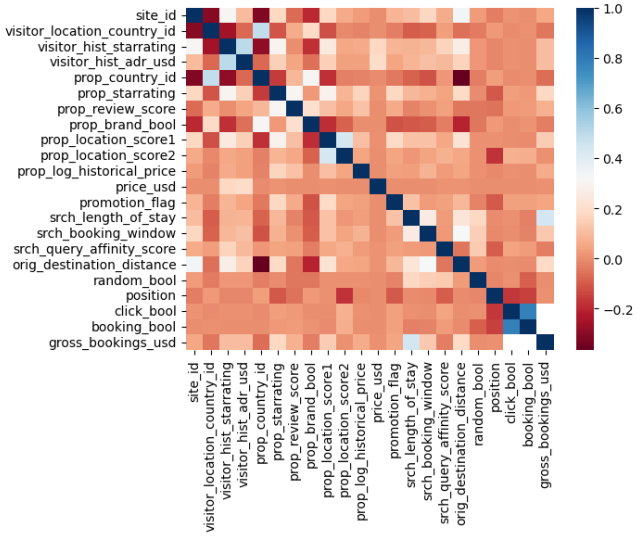


Fig. 2. Correlation

4 Data Preparation

4.1 Remove missing values

Initially, we calculated the percentage of missing values in each column of our dataset. Any column that had more than 70% missing values was considered to contain too many gaps to provide reliable information. Therefore, we decided to drop such columns from our dataset.

4.2 Data Engineering

Time-Based Features We noticed that our data contained a 'date_time' column. Recognizing that temporal aspects could hold valuable predictive power, we decided to enrich our dataset with additional time-related features. By transforming the 'date_time' column into a datetime format, we were able to create features such as 'hour_of_day', 'day_of_week', 'is_weekend', and 'season'. These new features represent the hour of the day, the day of the week, whether the day falls on a weekend, and the season of the year, respectively.

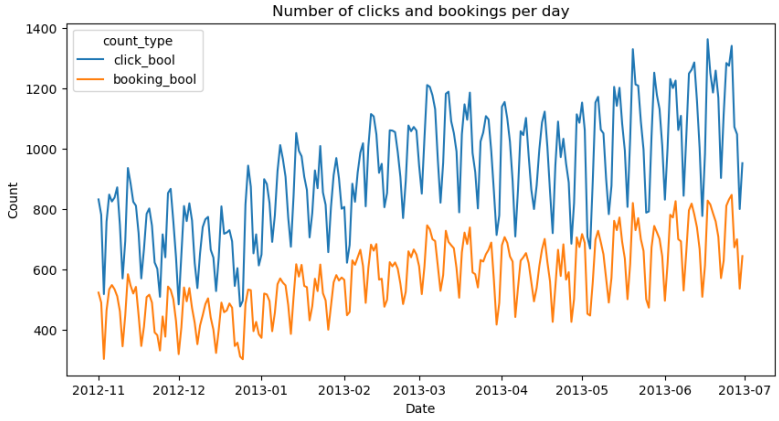
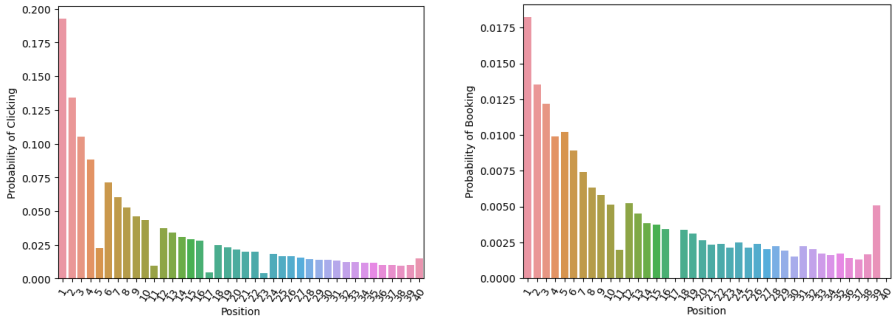


Fig. 3. Number of clicks and bookings change per day



(a) Clicking probability (b) Booking probability

Fig. 4.

Location-Based Feature Next, we created a location-based feature called 'domestic_search'. This feature is based on whether the visitor's location matches the location of the property. We assumed that domestic searches might influence the booking or clicking behavior differently than international searches.

Historical Price Feature We created a feature called 'price_difference' which is the difference between the historical log price of the property and the current price in USD. This could indicate if the property is currently priced above or below its historical average.

Competition-Based Features Next, we used the competition rate and availability data provided by various competitors to create several new features. We

created indicators for situations where a competitor had a lower price but no availability, as well as an overall count of competitors with data available.

Search Query Features We used the search query data to engineer a few additional features, such as the total number of people included in the search (both adults and children), as well as the number of children and adults per room.

4.3 Handling Missing Values

We then moved to handle missing values. In this stage, we focused on the numeric columns. We computed the median value for each numeric column in the training set and used these median values to fill in corresponding missing values in both the training and the test set. In this case, we chose to use the median instead of the mean to fill in missing values because using the mean could lead to meaningless results in some columns. For example, in the "season" column, the mean would not be applicable.

4.4 Feature Scaling

For our machine learning models to work effectively, we wanted to make sure that all our features were on the same scale. For this, we used the StandardScaler from sklearn.preprocessing to standardize our features.

5 Modeling and Evaluation

5.1 Item-based Recommendation Approach

Firstly, we applied an item-based recommendation approach to this problem, we considered the characteristics of the properties (items) and how they matched with the user's preferences. This approach is also known as item-based collaborative filtering.

Item Similarity Calculation: First, we computed the similarity between items. This was done based on their attributes or based on user behavior. In this dataset, we had several property characteristics like 'prop_starrating', 'prop_review_score', 'prop_brand_bool', 'prop_location_score1', 'prop_location_score2', etc. We considered these attributes to calculate the item similarity. A common method to calculate item similarity is cosine similarity.

Matrix Creation: After calculating item similarity, we created an item-item matrix where each cell ij represents the similarity between item i and item j.

Rating Predictions: The next step was to predict the user's interest in unseen items. This was done by taking a weighted sum of the user's past item interactions, where the weighting is the similarity between the past interacted item and the unseen item.

Recommendation: Once we had predicted the user's interest in all unseen items, we recommended the items with the five highest predicted interests.

Evaluation: Finally, we evaluated the performance of our model using appropriate metrics. In this case, NDCG (Normalized Discounted Cumulative Gain) was used as the evaluation metric.

In terms of results, the NDCG score for the item-based approach was not high, so we proceeded to try two other strategies.

5.2 Light Gradient Boosting Machine

These algorithms are excellent at maximising ranking models by considering the relevance and using pairwise comparisons. In particular, hotels are prioritised based on how well they match user queries and historical user comments. Such algorithms improve the ranking model by reducing discrepancies between predicted and actual user preferences, and they capture the subtleties of hotel relevance by using pairwise comparisons to produce more accurate rankings.

LambdaRank We then tried to apply the LambdaRank algorithm for prediction and ranking provided by the `LGBMRanker` of `LightGBM`[1], which is based on the concept of learning-to-rank and aims to optimise the order of the list of items according to certain criteria. The pairwise loss function is used to measure the difference between the desired and actual ranking of two items, and a gradient boosting framework is also used to iteratively update a set of weak rankers to form a strong ranker [1]. It can also handle different types of features, text, numbers or categories, and can incorporate user feedback, in our scenario, the information related to clicks and bookings, to improve ranking quality and accuracy.

LambdaMART Inspired by the previous competition solutions, we decide to train our dataset using the LambdaMART algorithm by setting the `dart` boosting [2] based on `LightGBM` framework, it combines the advantages of MART (Multiple Additive Regression Trees) and LambdaRank. The LambdaMART algorithm also contributes to our final model and was applied in the final prediction. The general algorithm of LambdaMART is shown below, in general, it runs through a predetermined number of trees and starts by calculating the

[1] https://lightgbm.readthedocs.io/en/v3.3.5/

gradients of all sample pairs, taking into account their relevance labels. Then a regression tree is trained using the feature vectors and the calculated gradients $\Delta\lambda_{ij}$, and the weights N of each leaf node in the regression tree are determined. Finally, the score s_i of all samples is updated using the calculated weights [3], which leads to our final result.

Algorithm 1 LambdaMART Algorithm

Require: Training set $\{(x_i, y_i, q_i)\}$ where x_i is the feature vector, y_i is the relevance label, and q_i is the query identifier. Number of trees T Learning rate η

for $t = 1$ to T **do**
  Compute the gradients $\Delta\lambda_{ij}$ for all pairs (i, j)
  Train a regression tree T_t on $\{(x_i, \Delta\lambda_{ij})\}$
  for all leaf nodes N in T_t **do**
    Compute the weight w_N for leaf node N
  Update the scores s_i for all samples
function ComputeGradients($\{(x_i, y_i, q_i)\}$)
  Initialize gradients $\Delta\lambda_{ij} = 0$ for all pairs (i, j)
  for all queries q **do**
    Sort samples in q by their scores s_i
    for $i = 1$ to $n_q - 1$ **do**
      for $j = i + 1$ to n_q **do**
        Compute δ_{ij}
        Update gradients $\Delta\lambda_{ij} = \Delta\lambda_{ij} + \delta_{ij}$
  return gradients $\Delta\lambda_{ij}$
function ComputeWeight(N)
  Compute the weight w_N
  return weight w_N
function UpdateScores($\{(x_i, y_i, q_i)\}$)
  for all samples (x_i, y_i, q_i) **do**
    Update the score s_i

5.3 Hyperparameter Optimization

We apply the `Optuna` framework[2] aiming to find the best set of hyperparameters for our LightGBM ranker model training process. The selected parameter is shown in Table 5.3, among these, `lambda_l1` and `lambda_l2` are the regularization term for the L1 criterion and L2 norm, which helps prevent overfitting. Since the LambdaMART algorithm performs well in our ranking task, after parameter optimization, this model constitutes our final model.

[2] https://optuna.readthedocs.io/en/stable/

Parameter	value
bagging_fraction	0.638
feature_fraction	0.840
lambda_l1	9.607e-07
lambda_l2	0.811
learning_rate	0.075
num_leaves	31

Table 4. Parameter

5.4 Evaluation

As mentioned before, the evaluation metric is based on the $NDCG@5$ [4], the formula is shown below, by dividing $DCG@5$ by $IDCG@5$, the $NDCG@5$ score can be obtained, which is a value between 0 and 1. A higher $NDCG@5$ indicates better ranking performance, where the top 5 hotels are more relevant to the user's preferences.

$$NDCG@5 = \frac{DCG@5}{IDCG@5}$$

where:

– $DCG@5$ (Discounted Cumulative Gain at 5) represents the sum of the discounted relevance scores for the top 5 ranked hotels. The value of rel is the relevance score of the top 5 hotels in the ranking.

$$DCG@5 = \sum_{i=1}^{5} \frac{rel_i}{\log_2(i)}$$

– $IDCG@5$ (Ideal Discounted Cumulative Gain at 5) is the maximum possible $DCG@5$, achieved when the top 5 hotels are perfectly ranked in descending order of their relevance scores. To calculate $IDCG@5$, which considers the ideal ordering of hotels based on their relevance scores, suppose the ideal ranking has relevance scores $idcg_i$ for the top 5 properties.

$$IDCG@5 = \sum_{i=1}^{5} \frac{idcg_i}{\log_2(i)}$$

$$score = \begin{cases} 5 & \text{if } \texttt{click\_bool} = 1 \text{ and } \texttt{booking\_bool} = 1 \\ 1 & \text{if } \texttt{click\_bool} = 1 \text{ and } \texttt{booking\_bool} = 0 \\ 0 & \text{if } \texttt{click\_bool} = 0 \text{ and } \texttt{booking\_bool} = 0 \end{cases}$$

Based on this, we map the `click_bool` and `booking_bool` according to the formula above, which would facilitate our training and validation process, and also help us compare the performance of different models as well as the benchmark. The results for the model we applied mentioned above are shown in Table 5.4, which is based on the public score of the Kaggle competition.

Model	NDCG@5
`LambdaRank` without statistical feature	0.25834
`LambdaRank`	0.33363
`LambdaMART`	0.37870

Table 5. Result

5.5 Content-Based Collaborative Filtering

We also try to use content-based collaborative filtering during the task, I selected the features of search information and hotel attributes from the datasets. Prior to that, I mapped the "clicking-bool" and "booking-bool" columns to obtain ratings of 0, 1, or 5. For missing values, I dropped the corresponding rows after extracting the required columns.

Next, I created a function called "scale-train" to normalize the features using the "StandardScaler" function. This normalization process helps to compact the data and improve training effectiveness.

Subsequently, I built a model consisting of two three-layer neural networks. Each layer is fully connected, and the first two layers utilize the ReLU activation function. After obtaining the outputs of the two features in their respective neural networks, I performed a dot product between the two vectors to obtain the numerical output of the content-based filtering. This result was then used for sorting and generating the final output of recommended hotels.

In summary, I implemented a hotel recommendation system using content-based collaborative filtering. I selected relevant features from the dataset, performed mappings for ratings, handled missing values, and applied feature normalization using the "StandardScaler" function. The model architecture consisted of two three-layer neural networks with ReLU activation. The final output was obtained by taking the dot product of the feature vectors and used for sorting and generating the recommended hotel list.

6 Deployment

To deploy our approach to Expedia's system in a scalable way. The amount of data Expedia own is the first issue to address, as it may not fit in memory or take too long to train. One solution is to speed up the training process by using a distributed computing framework such as Spark or Dask. The second need is to address the dynamic nature of the data, which can change over time due to seasonality, trends or user preferences. To keep the model up to date with the latest data, one solution is to use online learning techniques such as incremental updates or sliding windows. In addition, to avoid overfitting and improve generalisation, it is necessary to ensure that the model is robust and reliable in dealing with missing values, outliers or noisy labels. This can be done using regularisation techniques, L1 or L2 penalties, or feature selection methods. We can use Apache Kafka to stream data from a variety of sources,

including user profiles, hotel databases and weblogs. In addition, we can use Apache Airflow to coordinate data pipelines and workflows such as data preprocessing, feature engineering, model training, testing and deployment, Apache Spark MLlib to train and evaluate our LambadaMART models on large datasets based on distributed algorithms and GPU support, Apache Hadoop to store and manage data in distributed file systems such as HDFS or S3, and Apache Hive to query and analyze data using an SQL-like language.

7 Discussion

Indeed, our model leaves much to be desired and much room for improvement. Looking at the solution in the previous competition, Jiang et al. implemented model selection and feature selection to improve classification results, they also used different classifiers and implemented backward search algorithms, as well as collaborative filtering to rank hotels for each search term [5]. And Liu et al. combined the results of different models such as logistic regression, random forests, gradient boosters, extreme random trees, decomposition machines, as well as deep learning techniques using list methods, linear methods and deep neural network techniques. They also introduced ensemble techniques to combine separate models, such as z-score, GBM deep learning and list integration [6]. The combination of models greatly improves the accuracy of NDCG. These works give us much inspiration for possible future work, we can try to find a better method to avoid overfitting issues, as well as improve the model accuracy by using ensemble modelling methods and enriching the methods for imputing the null values, approaches to feature selection and engineering.

8 What we learned

Zhuofan Through the whole process of this assignment, I have a more profound comprehension of the importance of data engineering in machine learning tasks. Then, also appreciates the significant effort required to improve the predicted results, albeit by a fractional increase.

Yue Upon completing the background research, data preparation, and exploration of the item-based approach in this assignment, I have gained an understanding of how to establish a general framework for a recommendation system.

Pengju Implement a content-based collaborative filtering recommendation system, along with its principles, and gained a deeper understanding of machine learning.

References

1. C. Burges, R. Ragno, and Q. Le, "Learning to rank with nonsmooth cost functions," *Advances in neural information processing systems*, vol. 19, 2006.

2. R. K. Vinayak and R. Gilad-Bachrach, "Dart: Dropouts meet multiple additive regression trees," in *Artificial Intelligence and Statistics*. PMLR, 2015, pp. 489–497.
3. C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, no. 23-581, p. 81, 2010.
4. Y. Wang, L. Wang, Y. Li, D. He, and T.-Y. Liu, "A theoretical analysis of ndcg type ranking measures," in *Conference on learning theory*. PMLR, 2013, pp. 25–54.
5. X. Jiang, Y. Xiao, and S. Li, "Personalized expedia hotel searches," 2013.
6. X. Liu, B. Xu, Y. Zhang, Q. Yan, L. Pang, Q. Li, H. Sun, and B. Wang, "Combination of diverse ranking models for personalized expedia hotel searches," *arXiv preprint arXiv:1311.7679*, 2013.

Appendix: Process Report

Schedule and Contribution

Member	Contribution
Pengju Ma	On May 9th, explored the dataset and looked for suitable approaches for the task at hand. From May 10th to 15th, cleaned up the data and did some feature engineering and implemented a content-based collaborative filtering system using self-made neural networks. From May 16th to 23rd, try to test the result using the self-made model.
Zhuofan Mei	On May 9th, explore the dataset and do the EDA task. From May 10th to 17th, do the literature review and find the suitable model and algorithm. From May 18th to 25th, apply the processed data to the model, performed the parameter optimization, as well as submit the results.
Yue Zhang	May 9-10, explored past competitions, gained an understanding of the background knowledge of the competition, and grasped the model ideas of the top-ranking teams. Completed the Introduction and Business Understanding sections. May 11-17, added several new feature columns based on the understanding of the dataset and the requirements of subsequent tasks. Completed the Data Preparation section. May 18-26, based on the understanding of the course content and the division of tasks within the group, used the item-based method for modelling.

Table 6. Schedule and Contribution

For the report part, each of us wrote the tasks we are responsible for, corresponding to the respective section in the report.

Reflection

We reflected and concluded on the ability to work together as a team. We value the value and contribution of each member, create a positive work atmosphere, and work together to pursue clear team goals. Mutual support and cooperation are key to our success, and we will continue to strive to stay united and achieve greater results. The completion of the project is the result of the interplay between the members and their respective roles.