Scientific Report - Group 19

Xiaoyu Yang [2640948, xyg230], Jiamian Liu [2632301, jlu510], and
Song Yang[2638728, syg340]

1 Introduction

Expedia is the largest online travel agency in the world and we want to help the company to give the best match hotels to their customers, and finally to organize the search results for a user in the most suitable way. In this project, we built a recommendation system for Expedia via different approaches to predict what hotel a user is most likely to book. We got the dataset including shopping and purchase data as well as information on price competitiveness. All the coding work is done in python.

The dataset contains users' hotel search query history and the hotel properties. The training data also contains whether the user clicked on the hotel and booked it. We can find a combination of a search query by a user in each line. There some features only for the training set, including position_bool, click_bool, booking_bool and gross_booking_usd.

The report is organized as follows. Section 2 outlines the dataset statistics, plots, distributions and variables Analysis. Section 3 discusses the dataset pre-processing and variables analysis. Section 4 shows the algorithm details and models. Section 5 introduces the model evaluation and final model. Finally, Section 6 is about what we have learned and the reflection from the project and this course.

2 Dataset Statistics, Plots, Distributions and Variables Analysis

We were offered 2 datasets, the training set is "training_set_VU_DM.CSV" and the test set is "test_set_VU_DM.CSV". From the given document we could know that there are 54 features in the training set and 50 features in the test set. In our project, "click_bool" and "booking_bool" were the main "y value" we use in this model. The features could be mainly divided into 4 types, namely hotel features, query features, user features and compare features. For example, hotel features included "prop_country_id", "prop_starrating" and so on, which represented attributes of the hotel. Similar features could be aggregated and combined for feature engineering except for "date_time". The type of all the other features are integer or float, which means the data in these columns could be used, calculated or processed directly.

Firstly, we needed to do some statistical and plot before data processing. Via "shape" instruction in python, we could know that there are 4958347 records in the training set and 4959183 in the test set. However, not all data is complete, a lot is missing. The missing value mainly exists in compare features: "visitor_hist_starrating", "visitor_hist_adr_usd", "prop_review_score", "prop_location_score2", "srch_query_affinity_score" and "gross_bookings_usd".

We should deal with the missing data carefully and we will illustrate the relevant work in section "Dataset Pre-processing" in detail.

Besides, data is not balanced, negative data is much more. We first defined the value of "click_bool" in positive sample equals to 1. Only around 4.5% of the data is positive. The statistical results of "booking_bool" and "click_bool" are as shown in Fig. 1.
We could know that data appears differently by "country id". So we then drew the graph

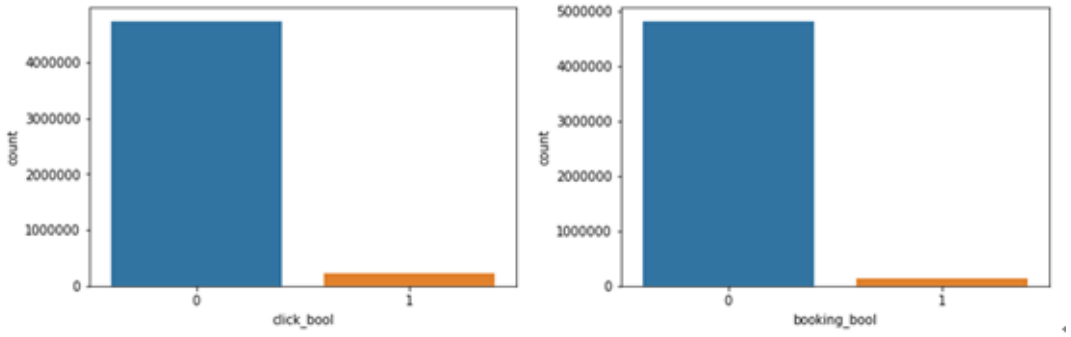

Fig. 1. Left: click bool, Right: booking bool

of "prop_country_id" and "visitor_location_country_id" as shown in Fig. 2. The result told us that we could split data by "country id" in further steps.
We then explored adult and children. The histogram of adults and children is as shown in

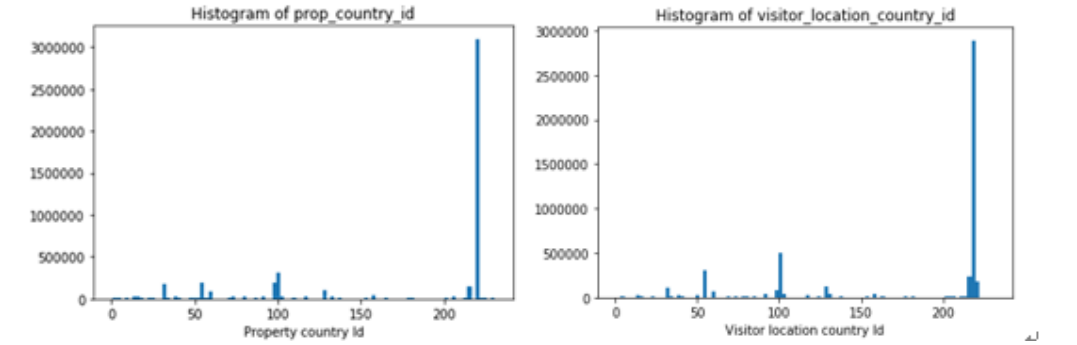

Fig. 2. Left: property country id, Right: visitor location id

Fig. 3. We could know that there are different kinds of booking, mainly booking for a family of parents with 1 or 2 kids.

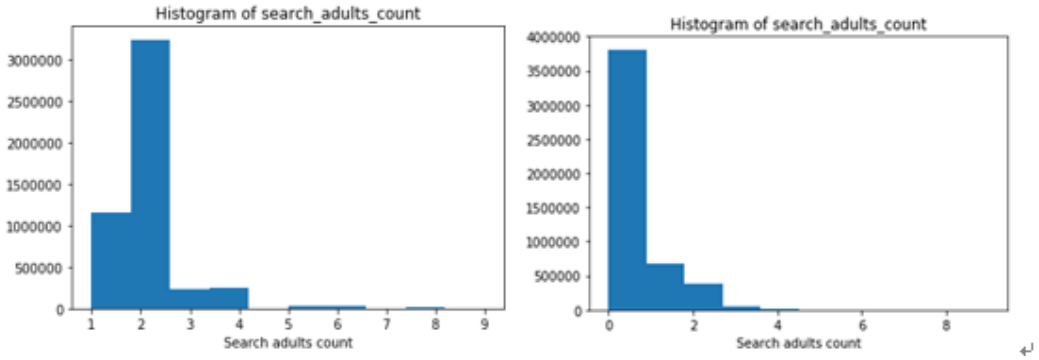

Fig. 3. Left: Adult count, Right: Children count

The length of stay is also an important feature for modelling. The figure is as shown in Fig. 4 left. We could know that most visitors tend to find accommodation for less than 5 days, mostly 1 day. And star rating is also an important factor. From Fig. 4 right we could know that 3 star is the most popular choice, while 2 or 4 star is also popular, which means we could consider more about these features. Also, we found that price influence booking or

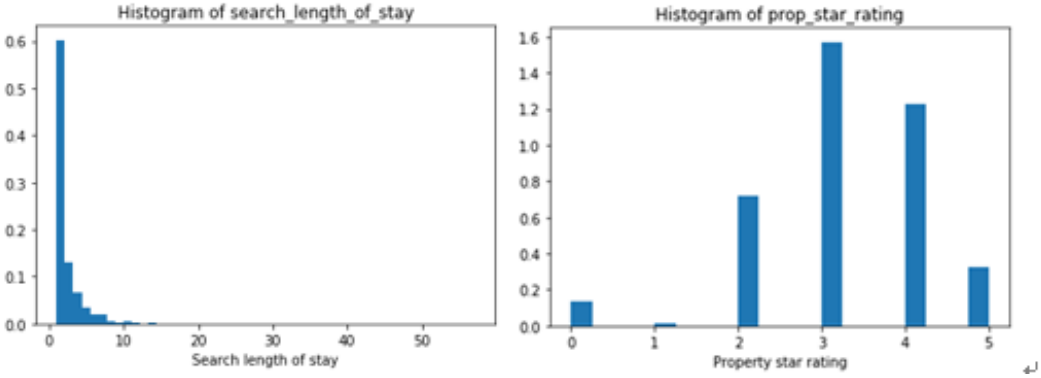

Fig. 4. Left: Stay days, Right: Star rating

click value a lot. So we drew the box-plot of booking and click with price as shown in Fig. 5. We set ylim [0:200] for better representation. In general, we could know that the "price_usd" that received a click or booking is always lower than those of did not get a click or booking. Furthermore, we found that the features are not equally important. We dropped the compare features in this figure for drawing, while this figure will be processed in section "Dataset Pre-processing". The correlation graph between different features is as shown in Fig. 6. From this figure, we could know clearly that most features have similar importance for "click_bool" or "booking_bool". However, features such as "visitor_location_country_id" and

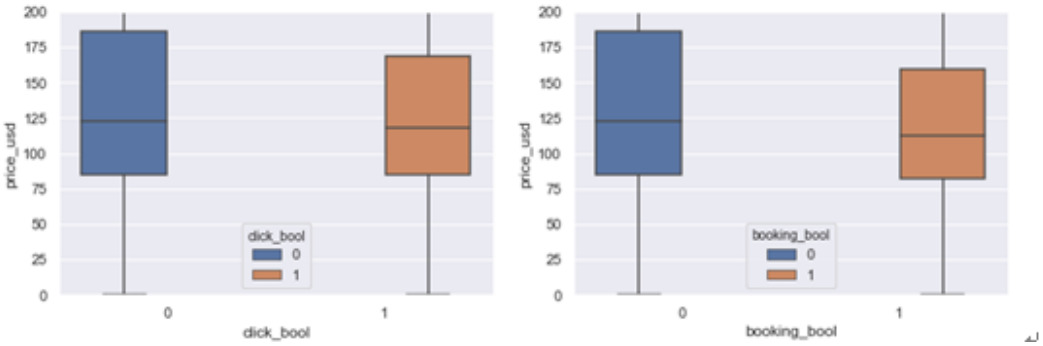

Fig. 5. Left: click and price, Right: booking and price

"prop_location_country_id" have a high correlation, we should concentrate more on these kinds of features, group or combine, to increase the accuracy of our model.

Moreover, from all the figures above and descriptive result generated via describe() in python, we could get a more detailed result of statistical characteristics, the sample is as shown in Fig. 7. And we further looked into the dataset, we found that there were also a lot of outliers in the given dataset. For instance, in "price_usd", this kind of features need to be processed in further steps.

3 Dataset Pre-processing and Variables Analysis

We first combined the training dataset and test dataset for the pre-processing and feature engineering work. The statistical result would be more accurate via combination. Then we assigned the value of "position", "click_bool", "booking_bool", "gross_booking_usd" as 0 temporarily. After the dataset pre-processing work, two datasets would be split again and the columns not belonging to the test dataset would be removed.

3.1 Treating Missing Values

As mentioned in previous sections, missing values affected a model a lot and a lot of missing values needed to be treated reasonably to train a better model. We filled different missing values with a different strategy in different situations.

For features like "prop_review_score", which did not have a lot of missing values, in order to process fast and high modelling, we filled the missing values with median values. As the median value was more stable than the mean value and 0 value was unrealistic in this situation.

For features like "prop_location_score2" and "orig_destination_distance", which had around 20% missing values and could be found a high correlation with other values, we filled the missing values based on other values or the median of other values. For example, the correlation

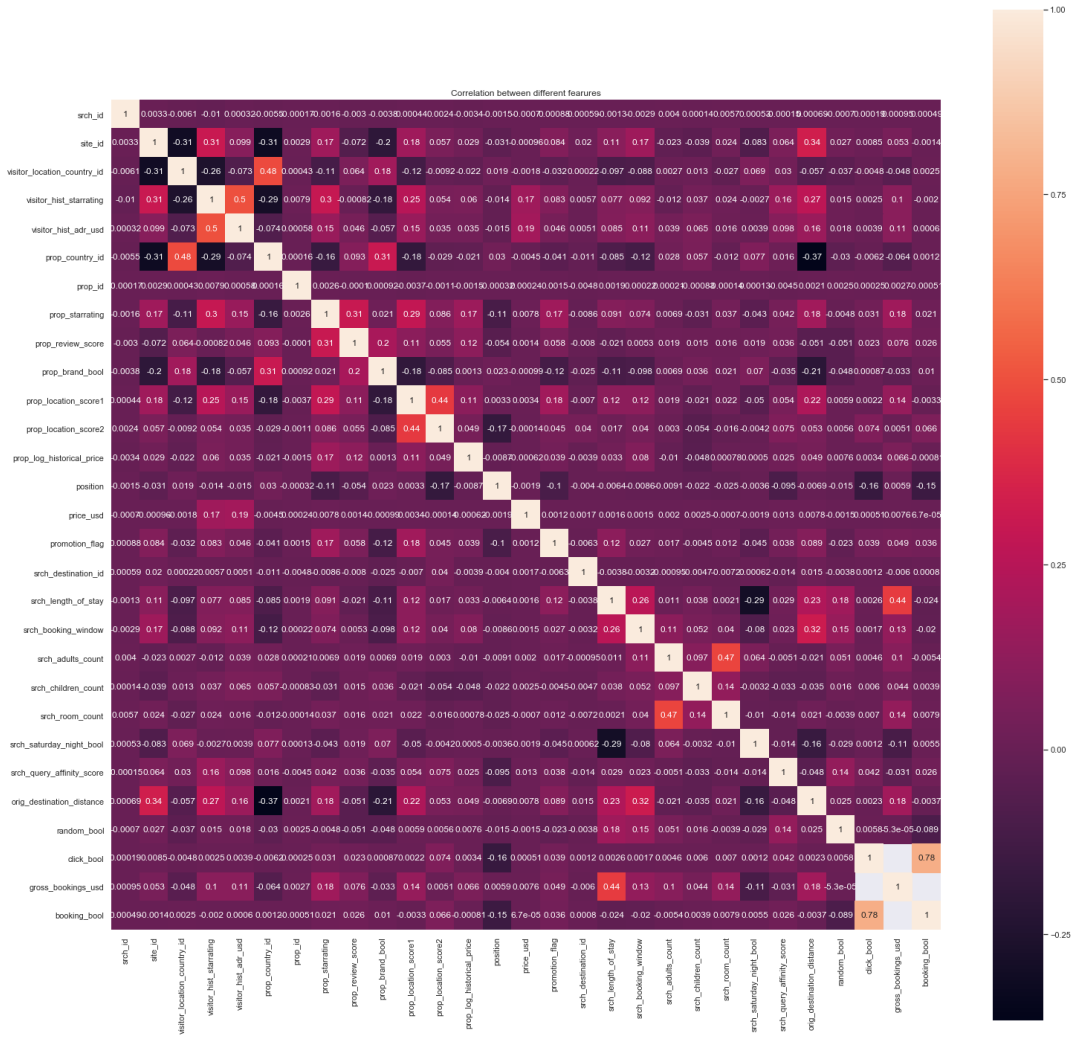

Fig. 6. Correlation between different features

	srch_id	site_id	visitor_location_country_id	visitor_hist_starrating	visitor_hist_adr_usd	prop_country_id	prop_id	prop_starrating
count	4.958347e+06	4.958347e+06	4.958347e+06	251866.000000	252988.000000	4.958347e+06	4.958347e+06	4.958347e+06
mean	1.663666e+05	9.953133e+00	1.753405e+02	3.374334	176.022659	1.739739e+02	7.007918e+04	3.180525e+00
std	9.611223e+04	7.646890e+00	6.591625e+01	0.692519	107.254493	6.834525e+01	4.060992e+04	1.051024e+00
min	1.000000e+00	1.000000e+00	1.000000e+00	1.410000	0.000000	1.000000e+00	1.000000e+00	0.000000e+00
25%	8.293600e+04	5.000000e+00	1.000000e+02	2.920000	109.810000	1.000000e+02	3.501000e+04	3.000000e+00
50%	1.665070e+05	5.000000e+00	2.190000e+02	3.450000	152.240000	2.190000e+02	6.963800e+04	3.000000e+00
75%	2.497240e+05	1.400000e+01	2.190000e+02	3.930000	213.490000	2.190000e+02	1.051680e+05	4.000000e+00
max	3.327850e+05	3.400000e+01	2.310000e+02	5.000000	1958.700000	2.300000e+02	1.408210e+05	5.000000e+00

Fig. 7. Sample of descriptive results within the dataset

between "prop_location_score2" and "prop_location_score1" is 0.44, which was rather high. Before we filled missing values, we grouped "prop_location_score1" and then filled median values. The group procedure would be described later.

For features like "visitor_hist_starrating", "visitor_hist_adr_usd" and "srch_query_affinity_score", whose over 95% values were missing values, we dropped these features. Although they had an obvious correlation with some other features, filling missing values with simple algorithms may cause negative effects on the model.

For compare features like "comp1_rate", "comp1_inv" and "comp1_rate_percent_diff", which had a lot of missing values and were different with each other a lot, the strategies are as follows: We first dropped the percent value since it could not be aggregated for us and was hard to process. We then filled the missing values for the other 2 with 0 and then combined these values respectively to generate new representative feature, namely "comp_rate_sum" and "comp_inv_sum". The final result showed these 2 new features could help to improve the model.

3.2 Grouping Data

Group and reassign value. In the given dataset, the type of some features is float, which means not all features are discrete and not all features could be directly applied into our model. So grouping data is necessary to avoid over-fitting. For example, for feature "prop_location_score1", the maximum value was 6.8 and the minimum value was 0. So we grouped the score from 0 to 7, the step is 0.2. Also, it was good for filling missing values in "prop_location_score2".
Group for models. As it was analyzed in section 2, we knew countries vary a lot. So we split data by "prop_country_id" into 172 pieces, This method greatly reduced time on training tree-based models.

3.3 Date Data

"date_time" is special because it is the only feature whose type is neither "Integer" nor "Float". However, via statistical analysis, we found that visitors tend to click or bool at some specific time like Saturday night. So we split the data and re-generated the values of "month", "day", "dayofweek" and "hour".

3.4 Composite Feature

We did some simple composite feature work for feature engineering. For example, from what we discussed in Section 2 we knew that there was a high correlation among the number of adults, children and room numbers. So we calculated the children/adultRoomcount in this situation.

Besides, to evaluate the quality of hotels, we calculated the booking and click probability for each. The probability was calculated by the ratio of booking numbers or click numbers with counting numbers.

3.5 Drop Redundant Features

We dropped the features who had over 95% missing values and initial compare features. Also, the features which generated composite features are also dropped in this step.

3.6 Balance Data

As the positive sample in the training set is only around 4.5%, in order to train the proper prediction model, we first needed to balance the data. We extracted about 5% of the training data for the training dataset. This step was quite important in the tree-based models.

After data processing procedure, we split the combined data, drop "position", "click_bool", "booking_bool" and "gross_bookings_usd" in test dataset and saved them respectively into "training_new.CSV" and "test_new.CSV" for further process.

4 Algorithm Details and Model

The features "position", "click_bool", "booking_bool" and "gross_bookings_usd" are the features which only exists in training set but not in test set, so these are the cadidate features for Y value in model. From Section 2 we knew that the missing value of "gross_bookings_usd" was over 90%, so this value should be dropped. Except for the 4 features mentioned above, the left features were the X values for processed training set and test set. According to the evaluation metric, 5 means the user purchased a room at this hotel and 1 means the user clicked through to see more information on this hotel. So the weight ratio between "booking_bool" and "click_bool" should be 4:1 since the necessary condition of booking is done by clicking. And for "position", we could know that the maximum value is 40 and minimum value is 1. 40 different values are reasonable in this dataset and do not need group values. The correlation coefficient between (4*booking_bool+click_bool) and position was about -0.158. So we defined the weight between (4*booking_bool+click_bool) and position as 1:(-0.16). To sum up, the final Y value was 4*booking_bool(y)+click_bool(y)-0.16*position(y). We predicted 3 y values for the final y value via the following different models using the X values mentioned above.

4.1 Random Forest Classifier

The most efficient, simplest and often used classifier is Random Forest Classifier. It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. What's more, we do not need to spend a lot of efforts on adjusting parameters in this classifier. The feature number fits this classifier well and it is efficient for the dataset. We directly used this classifier from python library, via: from sklearn.ensemble import RandomForestClassifier.

After several attempts, we set the parameters as RandomForestClassifier(n_estimators=500, oob_score=True, min_samples_leaf=7, n_jobs=-1). We set min samples for avoiding over-fitting, n_jobs=-1 for faster execution and n_estimators=500 performs best in attempts.

4.2 Gradient Boosting Classifier

Gradient Boosting Classifier builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. What important is that in each stage n_classes_regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Similar to Random Forest Classifier, it is also highly efficient in modelling procedure. Besides, a lot of high-level algorithms are the improvements of Gradient Boosting Classifier. It is necessary and meaningful to use this algorithm in this project. We directly used this classifier from python library, via: from sklearn.ensemble import GradientBoostingClassifier.

After several attempts, we set the parameters as GradientBoostingClassifier(n_estimators=500, learning_rate=0.3, loss='exponential', min_samples_split=7). We set loss and min min_samples_split in avoid of over-fitting and n_estimators=500 performs best in attempts.

4.3 LambdaMart

LambdaMART is one of the Learning to Rank algorithms. Learning to Rank is a class of techniques that apply supervised machine learning to solve ranking problems. LTR solves a ranking problem on a list of items. The aim of LTR is to come up with an optimal ordering of those items. As such, LTR doesn't care much about the exact score that each item gets, but cares more about the relative ordering among all the items. LambdaMART combines LambdaRank and MART (Multiple Additive Regression Trees). While MART uses gradient boosted decision trees for prediction tasks, LambdaMART uses gradient boosted decision trees using a cost function derived from LambdaRank for solving a ranking task. We need to rank items based on various features with a lot of inner correlations. For sure, it is the ideal algorithm for this assignment.

According to the instruction on Github, we implemented the LambdaMART in python. The default API of LambdaMART is: LambdaMART(training_data=None, number_of_trees=5, leaves_per_tree=0, learning_rate=0.1). After several tests, we set the parameters as: (training_data=df_train_new, number_of_trees=150, leaves_per_tree=7, learning_rate=0.5)

4.4 Ensemble Models

Using a single model usually could not get the best result. We needed to try some diverse methods to get a better result. Except for feature diverse, model diverse was necessary. Simply, a Z-score is the number of standard deviations from the mean a data point is. But more technically it is a measure of how many standard deviations below or above the population mean a raw score has. Z-score ensemble models are simple but usually the best ensemble choice according to experience. The formula of Z-score is listed as below.

$$Z(x) = \frac{x - \bar{x}}{\sigma(x)} = \frac{x - \bar{x}}{\sqrt{(x - \bar{x})^2/n}} \tag{1}$$

We firstly established and calculated each individual model. Then in each query we applied the training model in Z-score formula. Thus, a more accurate result would be generated.

After modelling, we split the metric set and prediction set and applied the training set in the trained model. We then sorted the final Y value of the prediction and then generated the output as CSV files for assessment.

5 Model Evaluation and Final Model

5.1 Model Evaluation

According to the requirements of Kaggle competition and the given instruction document, the evaluating metrics are described as follow. We used NDCG@38 (Normalized Discounted Cumulative Gain) as an evaluation metric. NCDG is a commonly used benchmark in information retrieval and was the measure used to determine the winner of the official Expedia contest. NDCG simply divides by the best possible DCG for X, which is desirable because different test sets may have different best-case DCGs. Hence, NDCG returns a score that falls between zero and one, with one being a perfect score, that is, one where documents display monotonically decreasing relevance scores. The relevant function is defined as follows:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{2}$$

Hotels for each user query are assigned relevance grades as follows 5 - The user purchased a room at this hotel 1 - The user clicked through to see more information on this hotel 0 - The user neither clicked on this hotel nor purchased a room at this hotel We then uploaded the results to kaggle, the best performance for each model are listed in the following table: From

Table 1. Public Score tested by Kaggle

Model	**Kaggle Score**
Z-score ensemble	0.35905
LambdaMART	0.35137
Random Forest Classifier	0.35026
Gradient Boosting Classifier	0.34885

above, we knew that z-score ensemble got the highest, LambdaMART also got a good score in assessment, while Random Forest and Gradient Boosting scored kind of lower but also got an acceptable score.

5.2 Final Model

The final model was the ensemble model which was based on the Z-score method. We applied Random Forest Classifier, Gradient Boosting Classifier and LambdaMart in the final model.

The detailed information about each classifier and how the classifiers are combined with the final model has been discussed in Section 4. The result of the final model was 0.35905,which met the expectation of our project.

6 What We Learned and Reflection

Through this amazing project, we learned the complete data mining process, including business understanding, data preparation, modelling, and evaluation. Also, through continuous improvement of models and feature engineering, we deeply understood Expedia dataset and learned various skills to handle the dataset, including treating missing values and grouping data and so on. We also learned lots of useful methods, such as random forests, gradient boosting classifier, LambdaMart, ensemble models, and cross validation.

Meanwhile, we learned how to handle large datasets. What is more, we also gained valuable experience to build effective teamwork and share knowledge. We used Git and Slack to cooperate and adopt the concept of agile development and improve dour model through continuous iterations. Furthermore, every Sunday evening, there was a small meeting between the team members to share the relevant knowledge of data mining. At weekly meetings, we discussed the latest technology of different aspects of data mining and how to continuously improve our model. Moreover, we have also established a small knowledge sharing library in Google drive to share state-of-the-art methods and papers. In conclusion, we not only learned advanced data mining technology, but also improved our teamwork and communication ability. Moreover, we appreciated our teacher Prof.Mark Hoogendoorn and our teaching assistants' efforts and assistance, which helped us a lot to keep learning and make efforts on the project.

References

1. Personalize Expedia Hotel Searches - ICDM 2013, https://www.kaggle.com/c/expedia-personalized-sort/discussion.
2. Combination of Diverse Ranking Models for Personalized Expedia Hotel Searches, https://www.researchgate.net/publication/25901039.
3. IEEE ICDM 2013 Data Mining competition: Expedia, https://www.kdnuggets.com/2013/09/ieee-icdm-2013-data-mining-competition-expedia.html.
4. Combination of Diverse Ranking Models for Personalized Expedia Hotel Searches, https://arxiv.org/abs/1311.7679.
5. The Power of Rankings, http://pages.stern.nyu.edu/ lbornkam/F15Seminar/UrsuPaper.pdf.
6. Scikit learn randomforestclassifier, https://scikit-learn.org/stable/modules/generated/sklearn. ensemble.RandomForestClassifier.html.
7. Intuitive explanation of Learning to Rank (and RankNet, LambdaRank and LambdaMART), https://medium.com/@nikhilbd/intuitive-explanation-of-learning-to-rank-and-ranknet-lambdarank-and-lambdamart-fe1e17fac418.
8. Scikit learn GradientBoostingClassifier, https://scikit-learn.org/stable/modules/generated/sklearn. ensemble.GradientBoostingClassifier.html.
9. LambdaMart implementation, https://github.com/lezzago/LambdaMart.
10. Z-Score: Definition, Formula and Calculation, https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/z-score/.