Data mining techniques-Assignment 2

Ziyu Zhou, Luyao Xu, Congyi Dong

VU-Group 66

1 Business understanding

Before we start to do this assignment, We searched for the previous works which were related to this topic, enabling us to have a deeper understanding of such topic. We found a paper written by Xudong Liu, Bing Xu, Yuyu Zhang. [1].

From their paper, we learned that this hotel ranking problem was a Learning to Rank(LeToR) problem, and that our aim was to train the model which would output the rank of given hotel id. There were two key points of this topic: how to do a good feature selection and feature engineering, and how to build up the LeToR model.

a. Feature selection and engineering

First, they did the data dimension reduction by focusing on more important features, making the model training simpler. They considered 'price_usd', 'prop_starrating', 'prop_location_score2' as the most important features in lisewise ranking. Then, they composited several features, which not only reduced the number of features but also improve the relevance between features and class labels.

Then, they pre-processed the data by dealing with missing feature values, extracting 10% data as sample, balancing the sample for Random forest, and using bucket algorithm to binarize float feature.

b. Ranking models

They tried different methods which were all evaluated by NDCG:Pairwise logistic regression(0.51273), Random forest(above 0.52), Gradient boosting machine(0.50099), listwise method(above 0.51), deep learning method(0.52729) and so on. We also read the paper written by other people according to LeToR problem. It seems that Pointwise, Pairwise, Listwise algorithm families are the main methods to solve the relative problems.

2 Exploratory data analysis(EDA)

2.1 Overview of dataset

The training data contains 4,958,347 searching records, multiple browsing records per search id. There are 54 attributes in the dataset. 'Date time' is string data, while the others are numerical data including the already processed categorical data, e.g 'prop_brand_bool'. These attributes can be roughly divided into five groups, which are shown in Table 1 with count of missing values.

Table 1. Attributes which contain missing values(The attributes which are not in this table have no missing values)

Group name	Group member	Missing value	
User basic information	attributes start with 'visitor_', 'srch_id', and 'site_id'	visitor_hist_starrating	4706481
		visitor_hist_adr_usd	4705359
Hotel information	attributes star with 'prop_', 'position' , and 'price_usd','promotion_flag'	prop_review_score	7364
		prop_location_score2	1090348
Competitor information	attributes start with 'comp'	all attributes in this group have more than half missing value	
Search request	attributes start with 'srch_','random_bool', 'orig_destination_distance'	srch_query_affinity_score	4640941
		orig_destination_distance	1607782
Operation	'click_bool', 'booking_bool', 'gross_bookings_usd'	gross_bookings_usd	4819957

2.2 Characteristics of the dataset

a. Correlation analysis

Before starting the feature selection, we calculated the Pearson's correlation coefficient of each pairs of attributes. Table 2 shows the top 5 correlated attributes for 'click_bool' and 'booking_bool'.

Table 2. Top 5 correlated attributes for 'click_bool' and 'booking_bool'

	click_bool	booking_bool
1	position (0.164992)	position (0.147918)
2	prop_location _score2 (0.073807)	random_bool (0.088891)
3	srch_query_ affinity_score (0.041764)	prop_loca- tion_score2 (0.066405)
4	prop_starrating (0.030788)	promotion_flag (0.036047)
5	comp8_rate (0.023658)	prop_rev iew_score (0.025800)

We find the noticeable correlation within competitor information group data and within user information group data. Moreover, we find competitor data shows low correlation with data in other groups, while the correlation exists among other four data groups. The label group, operation group, shows the lower correlation with competitor data than that with other four data group.

b. The distribution of important features

First, we analysed the ratio of data in operation. The ratio is almost 13:8:473 (booked : only clicked : none) which is unbalanced.

Second, we were interested in some empirical intuitive reasons. For example, We analysed the distributions of 'prop_id' and 'position' in Fig 1, according to the booked hotels data. We can conclude that people prefer to book brand hotel and that people prefer to book 3-4 star rating hotel. What's more, people are more likely to choose hotels which have near positions no matter the hotels are shown randomly or not.

Third, we also evaluated the effect produced by history data. We analysed the distribution of the difference between the hotel data and history data(in Fig 2), finding that these differences were approximately normal distributed with expectation 0, which means people have stable booking preference which could be predicted on given history data. Thus, we decided not to drop the history data, but to make the whole data set into two group to do the pre-processing separately.

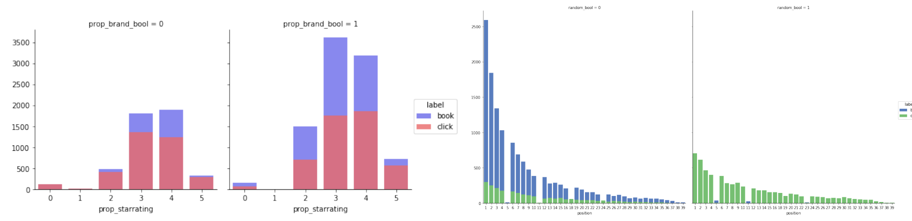

Fig. 1. The left two histograms show the discrete distribution of 'prop_starrating' of a booked individual hotel(first pink blue bar figure) and a booked brand hotel(second pink blue bar figure). The right two histograms show the distribution of 'position' of booked hotels shown in certain order(first green blue bar figure) and that of hotels shown randomly(second green blue bar figure).

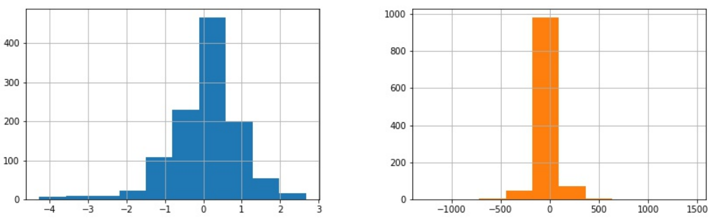

Fig. 2. The distribution of the difference between hotel star rating and user history star rating of hotels people booked(left), and the distribution of difference between the hotel price and user history booked hotel price of hotels people booked(right)

3 Data preparation

3.1 Feature engineering

a. Sample from the training set

We found the number of data in the training set was too large to run a respectively rapid algorithm, so we decided to randomly extract 10 per cent of the data by unique 'srch_id'. We found this sampling method didn't affect the distribution of features.

b. Build class label

We merged 'click_bool' and 'booking_bool' as the label of some kinds of algorithms. If a searcher(with the same search id) only clicked a hotel but not booked the hotel, we assigned value of 1 to the element of label with the corresponding hotel id and search id. If a searcher not only clicked but also booked the hotel, we assigned 5 to the corresponding element of label. If a searcher neither clicked nor booked a hotel, we assigned 0 to the element.

c. Features selection

We did the data dimension reduction because there are too many attributions which can lead the model overfitting and inefficient.

We noticed that the column named 'date_time', 'visitor_location_country_id', 'position', 'srch_destination_id' show little effect so we dropped them. Then we did feature sorting using method of XGBoost ranking to see what features showed relatively stronger relationships with the label.

We got the results shown in Fig 3.

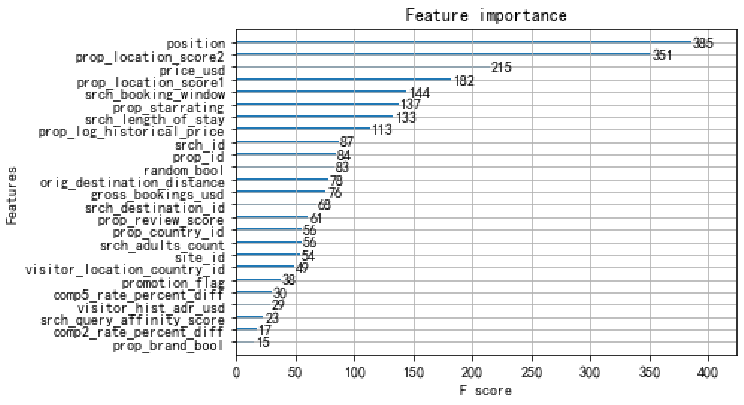

Fig. 3. The feature ranking before feature engineering using XGBoost ranking

d. Missing values

There are many missing values in many features. We have to mention that we found the two columns named $'visitor\_hist\_starrating'$ and $'visitor\_hist\_adr\_usd'$ have more than 80 per cent missing values. However, the searchers with history of searches can provide us with some useful information to help us improve our model. **So we decided to divide the data into two parts to do feature engineering separately: all the search records with history data and the search records without history data.** Then we would build two different models for the two groups of data respectively. After building models, we would also divide the test set into two parts and then test our models respectively.

In addition, we observed test data and found that there is no feature of 'position' in the test set so we also dropped the feature of 'position' in the train set.

e. Composite features

We sorted the prices of hotels and their competitors. If a hotel's price was located in the top 1/2, we thought the hotel got a competitive advantage and assigned 1 to the corresponding element of the new composite feature. If a hotel's price was ranked in the last half, we assigned 0 to the element of the new feature. We named this new feature $'price\_adv'$.

Similar to the procedure above, if the number of empty rooms in a hotel is located at the top one second among all competitors, we would assign 1 to the other new feature's element. Then if the number of empty rooms was ranked in the last half, we would assign 0 to the element of the new feature instead. We named this new feature $'avail\_adv'$.

According to the correlation matrix, we found many features showed very little correlation to the label. Also, for reducing dimension of features, we decided to merge sparse features [1]

$$ump = exp(prop\_log\_historical\_price) - price\_usd \tag{1}$$

$$price\_diff = visitor\_hist\_adr\_usd - price\_usd \tag{2}$$

$$starrating\_diff = visitor\_hist\_starrating - prop\_starrating \tag{3}$$

$$per\_fee = \frac{price\_usd * srch\_room\_count}{srch\_adults\_count + srch\_children\_count} \tag{4}$$

$$total\_fee = price\_usd * srch\_room\_count \tag{5}$$

$$score1d2 = \frac{prop\_location\_score2 + 0.0001}{prop\_location\_score1 + 0.0001} \tag{6}$$

f. Balance data

Balanced data is used in training random forest and deep neural network. For there are only a very small per cent positive data points among the data points, we chose one positive example and randomly choose one negative example until we got enough data set for training. In this way we can train tree-based models with a large amount of trees in a reasonable time.

3.2 Does feature engineering work?

In terms of history data feature engineering group, we used Gradient Boosting Machine to draw the feature importance before and after feature engineering.From Fig 4 , we can see that the feature importance indeed improve: the highest coefficient increase from 0.05 to 0.35.

We also did the classification test to prove that with new feature matrix the accuracy would be in a good level. We extracted 80% sample data as training set, the rest 20% as test set. The results are shown in Table

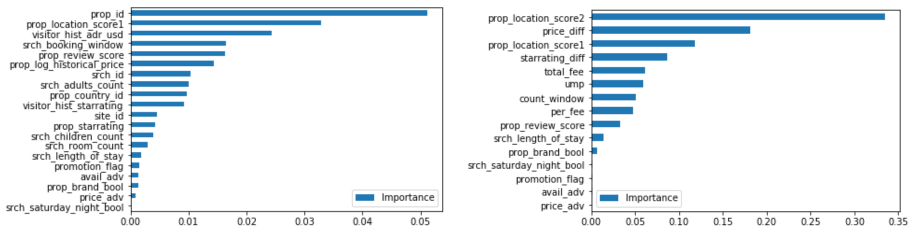

Fig. 4. The feature importance for history data group before(left) and after(right) feature engineering

To verify our feature engineering was meaningful and correct, we used two classifiers to do the classification based on new training feature matrix. Then we substituted the features into the model and compared the result predicted with the label. The results and accuracy of each classifier are as follows:

Gradient Boosting Classifier:0.9553413322957446.

Decision Tree Classifier:0.9999483852524755.

4 Modelling

4.1 Pointwise-PRank

a. Algorithm outline

We tried the PRank algorithm in Pointwise class to solve the LeToR problem. In PRank algorithm, the training set labels are not scores but several levels from strong to week, for instance, perfect > excellent > good > fair > bad. The aim of PRank is to train a ranking rule H to map the sample feature matrix to k different levels, so how to find the threshold value to separate scores is important. The steps of the algorithm is as follows:

Step 1: Using One-hot coding to map each k-class training set label to a $1 \times k$ matrix. For example, in 4 class set, we map label $y = 3$ to $[1, 1, 1, -1]$.

Step 2: Build the model to calculate the score of each feature vector. The model is as follows:

$$Score = w \cdot x \tag{7}$$

, where w is the weight vector, x is the feature vector. Set the initial value of weight vector w and threshold value vector b.

Step 3: Train model. Substitute the training set sample feature vector in score model, map it to exact level, and calculate the error. Update w and b, by reduce the error. Get the ranking rule H:

$$H(x) = min_{\{r=1,2,..,k\}}\{r : w \cdot x - b_r < 0\} \tag{8}$$

Step 4: Applying the ranking rule to testset.

b. Disadvantages of Pointwise model

After trying pointwise methods, we found the results were not that satisfying. Comparing with pointwise methods, pairwise approach to rank have many advantages.

To be specific, pointwise approach to rank has drawbacks as follows:

1.The ranking pursues the sorting result and does not require accurate scoring, as long as there is a relative score.

2.The pointwise class method does not take into account the internal dependencies between docs for the same query. On the one hand, the samples in the input space are not i.i.d, which violates the basic assumptions of ML. On the other hand, the structurality between such samples is not fully utilized. Second, when different queries correspond to different numbers of docs, the overall loss will be dominated by the query group with a large number of docs. It should be said that each group of queries should be equivalent.

3.The loss function also does not have position information from the model to the predicted sort. Therefore, the loss function may inadvertently emphasize too much of the docs that are not important, that is, those docs whose ordering has a small impact on the user experience.

The loss function of pairwise approaches evaluates the difference between the predicted preference and the true preference of the doc pairs.

4.2 Pairwise

We picked XGBoost approach as our method of modelling and ranking. XGBoost is fast, effective, capable of handling large data, supporting multiple languages, supporting custom loss functions and more. Also, XGBoost is natively supported by rank, just set the objective in the model parameter to objective="rank:pairwise".

a. XGboost for ranking

Since xgboost is a lifting tree model, it is closely related to the decision tree. It integrates many decision trees to get a strong classifier. The main idea in xgboost is very similar to the CART regression tree. Suppose that

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant \tag{9}$$

Then use Taylor expansion to approximate our original objective:

$$f(x+\Delta) \approx f(x) + f^{'}(x)\Delta x + \frac{1}{2}f^{''}(x)\Delta x^2 \tag{10}$$

Then define $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$. Then the objective function becomes:

$$Obj^{(t)} \approx \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) + constant \tag{11}$$

The algorithm can further simplify the objective function given above.

b. RankSVM

The basic idea of RankSVM is to convert the sorting problem into a pairwise classification problem, and then use the SVM classification model to learn and solve:

$$f(x_i) > f(x_j) \Leftrightarrow < w, x_i - x_j > 0 \tag{12}$$

Ranking SVM uses SVM for classification:

$$min_{w,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \tag{13}$$

$$s.t. y_i \langle w, x_i^{(1)} - x_i^{(2)} \rangle \leq 1 - \xi_i \tag{14}$$

Using Ranking Svm for ranking, we could get the rank of hotels according to the degree that customers would like to book of every search id. The rough algorithm is as follows:

Step 1: Calculate the difference vectors of each pair feature vectors of samples in train set, e.g calculate A-B, where A and B are the feature vector for a pair of samples.

Step 2: Assign label 1 to difference vector if A has a higher rank than B, in this case booked hotel has rank higher than only clicked, and only clicked hotel has rank higher than none operation hotel. Assign label 0 to this vector if B has higher rank than A.

Step 3: Use SVM to do the two-class classification to obtain the relative ranking.

Step 4: Output the rank, and use NDCG to evaluate the model.

where $C = (S_1 - K)^+$. Then $\pi(C)$ represents the set of all AF prices of the call option C.

Then according to theorem 1.16 and $d = 1$, we can get that

$$\begin{aligned} M_0 &= m \in [0,\infty] | \exists \xi \in Rwith m + \xi \cdot y \leq \frac{C}{1+r} Pa.s \\ M_1 &= m \in [0,\infty] | \exists \xi \in Rwith m + \xi \cdot y \geq \frac{C}{1+r} Pa.s \end{aligned} \tag{15}$$

Substitute $y = y_1 = \frac{S_i}{1+r} - \pi_1$ in to the above sets, we can get

$$\begin{aligned} M_0 &= m \in [0,\infty] | \exists \xi \in Rwith m + \xi \cdot (\frac{S_i}{1+r} - \pi_1) \leq \frac{C}{1+r} Pa.s \\ M_1 &= m \in [0,\infty] | \exists \xi \in Rwith m + \xi \cdot (\frac{S_i}{1+r} - \pi_1) \geq \frac{C}{1+r} Pa.s \end{aligned} \tag{16}$$

Then the lower limit is $inf\pi(C) = maxM_0$ and the upper limit is $sup\pi(C) = minM_1$.

4.3 Listwise-LambdaMART

In addition to pairwise ranking, XGBoost can perform listwise ranking using LambdaMart algorithm. This algorithm combines LambdaRank and MART by using λ as gradient. That is, MART generates multiple weak learners and combines them using weighted average, while λ is used to determine the next learner. The recipe is the following:

step 1: Before training a new regression tree, sort the documents according to the current model.

step 2: Compute lambdas.

step 3: Do a single step of Newtons method to optimize gradient (lambda) predictions in terminal nodes.

5 Evaluation of models

We used two method to evaluate the ranking model-cross validation and NDCG.

There is the basic idea of NDCG.

"The evaluation metric for for this competition is Normalized Discounted Cumulative Gain(NDCG), which is used in ranking"[1]. "NDCG is a commonly used benchmark in information retrieval" [3]. There we have a ranked sequence $X = \{x_i\}_N$ in the order of decreasing predicted relevance, and a relevance function $f(X)$[3], we could define DCG(Discounted Cumulative Gain) as follows:

$$DCG@K(X) = \sum_{i=1}^{K} \frac{2^f(x_i) - 1}{log_2(i+1)} \tag{17}$$

, where K means the we consider the top K ranking to do the evaluation. And in this case the relevance function $f(X)$ is defined as follows[3]:

$$f(X) = \begin{cases} 5 & \text{if hotel booked,} \\ 1 & \text{if hotel only clicked,} \\ 0 & \text{none operation.} \end{cases}$$

The core models of NDCG are as follows:

$$NDCG@K = \frac{DCG@K}{idealDCG@K} \tag{18}$$

, where idealDCG@K is calculated directly by the given label, while DCG@k is obtained based on the output rank of LeToR ranking model.

6 Numerical experiments and parameter adjustment

a. Pointwise

We firstly tried PRank to do the ranking, but we found that key point of PRank was to set up a proper threshold value to map the feature vector to a certain level, which was not very suitable to solve hotel ranking problem. We still used PRank to do the 3 class classification(booked, only clicked and none). We did the three-class classification validation based on the training set, the accuracy was 0.6623. It was hard for us to obtain the rank with in certain class.

b. Pairwise

Then, we used Pairwise method: XGboost and Ranking SVM(with auto-parameter SVM) to do the pair wise rank. The NDCG scores are on the leaderboard of Kaggle, 0.16246 and 0.14341 respectively.

We found that the XGboost had a higher score, so we decied to focus on XGBoost. XGboost was easily to be overfitting, so we change the parameter to try to improve the NCDG score. We set 'objective':'rank:pairwise','eval-metric':'ndcg@10-','n-estimators':40, and 'max_depth':3. We evaluated the model by performing 5-fold cross validation, which gives the mean test score of 0.1774 and the standard deviation of 0.0361. The NDCG score is 0.16430.

c. Listwise

XGboost is used to perform LambdaMart algorithm. The obejective of ranking is to maximize NDCG. To avoid over-fitting, we set the number of estimators to 40 and the maximum depth of the tree is 3. We evaluated the model by performing 5-fold cross validation, which gives the mean test score of 0.5397 and the standard deviation of 0.2557. And the accuracy score on Kaggle is 0.16219.

7 What we learned

We gain a lot of things from this data mining project.

First, we have a deeper understanding about how to do the feature engineering. We learned how to use EDA to mine the relationship between features to find the way of improvement, and read the papers about Dimension reduction, for example PCA(we didn't use since after feature merging, the dimension was reduced already), and about feature importance ranking, for instance GBM and XGBoost feature ranking. Second, LeToR is a completely new field to us. We learned the basic rules of LeToR, having access to Pointwise, Pairwise, and Listwise, and knew the root different of them. We knew XGboost the best among the models we used.

We also met difficulties that we were not that familiar with java and there are no enough packages for us to use, which means we need to use more time to explore how to code. We made the data set into two groups: one contains data with history, the other contains data without history. The expected outcome is that the NDCG score of the former group is higher than the latter group, and that the feature importance scores is improved, which means our feature engineering works. The unexpected result is that although after parameter adjustment, we improved the score, in general, our ranking code shows a relatively lower NDCG score in ranking, which means there still exist space for us to improve our feature engineering.

References

1. Xudong Liu, Bing Xu, Yuyu Zhang, ”Combination of Diverse Ranking Models for Personalized Expedia Hotel Searches”.
2. Xin Xing, Yao Xiao, Shunji Li, ”Personalized Expedia Hotel Searches”, Stanford University.
3. Saurabh Agarwal, Luke Styles, Saurabh Verma, ”Learn to Rank ICDM 2013 Challeng Ranking Hotel Search Queries”.