Assignment 2 - Advanced Data Mining Techniques Expedia

GROUP 47
Apala Medha Sahar$^{1[asa369]}$, Konstantinos Sakellariou$^{2[ksu230]}$, and Mansi Mundra$^{3[mma386]}$

Vrije Universiteit Amsterdam

1 Introduction

In recent years, competitions between organisations have increased to a great length. Thus, the companies want to understand how to keep their users satisfied. Expedia which is the world largest online travel agency (OTA) does this, by providing a list of potential hotels for millions of travel shoppers each day. To be considered as one of the best in the market and also to have a good sale, the OTA should provide a sorted list of best hotels for specific users with integration of good price and an edge towards the competitors [**1**]. In this assignment, we are using the dataset which was provided by Expedia, comprising of the data of purchasing and also the competitive prices. The dataset provided is based on the list of hotels that the user is provided after they search for hotels on the Expedia website.

2 Goal

The objective of this assignment is to understand the Expedia dataset and provide the customers with a better experience while searching for hotels. During this assignment, we are predicting what the user will most likely book for while searching for a hotel. Based on the predictions, we are using a ranking algorithm This ranking model will rank each hotel that the user is likely to book. The ranking will be provided in a decreasing manner, with the most likely to be booked hotel being on top and the least likely to be booked hotel being at the bottom. Based on this, we will provide a machine learning algorithm which can understand the user preference based on previous activities. This will help in maximizing the purchase rate of Expedia.

3 Relevant Work

For Personalize Expedia Hotel Searches- ICDM 2013, winning algorithms were published by the competition host Adam Woznica[**2**]. Owen Zhang from New

York, USA obtained first place by fulfilling the contest requirement and achieving the highest score. He divided the project into three tasks: preprocessing/feature engineering, Modeling, and Remark/Observation. For preprocessing of the data set, he used techniques such as missing value imputation, bounding numerical variables, and downsampling negative instances. For feature engineering, he divided features into five groups: all original features, numerical features averages over other important features, composite features, EXP feature, and estimated position. For Modeling, he used Gradient boosting Machines and implemented two types of models: with and without EXP features. He observed that, the most prominent features for his project were position, price, and location desirability. He also evaluated that random impressions were not completely randomized and training time and predictive performance were improved by downsampling negative instances.

Jun Wang and Alexandros Kalousis achieved second place in the competition [**3**]. They divided the project into four tasks: problem setting, modeling, feature engineering, and evaluation. For modeling, they used two methodologies: Linear and Nonlinear. Linear Regression model and linear learning-to-rank model (RankSVM and SGD approach) were used for Linear Methodology. For Nonlinear Methodology, Gradient Boosted Trees, Random Forest, and LambdaMART were used. After modeling, they used feature engineering techniques such as missing value estimation, feature extraction, and normalization. The most prominent predictors were “prop_id” and “prop_starrating”. They evaluated that what matters is feature engineering, the methodology used, and the number of learning instances.

4 Data Exploration & Preparation

The given dataset was provided in Kaggle. The training set consists of 4.959.183 samples with 54 attributes while the test set consists of 4.958.347 samples with 4 columns less, i.e, 50 attributes. The four attributes that are included in the training set are the following: ”position”, ”gross_booking_usd”, click_bool and booking_bool. The most crucial attributes are the last two attributes(see subsection 4.2), due to the description of our problem, where scores are assigned to properties based on clicking or booking a property. To better read the data and focus on the objective of the project, the data needs to be cleaned optimally. Integration of information is provided in the dataset. The dataset concentrates on a range of specific hotels along with a user’s search query. The primary column is the search_id, as this attribute can be used to identify multiple rows which constitute of one query. To elaborate further about the dataset, we have 3 types of variables in the sets, where we observe 16 categorical variables (”comp_inv” and ”comp_rate”) 8 for each competitor, 4 boolean variables (”prop_brand_bool”, ”promotion_flag”, ”random_bool” and ”srch_saturday_night_bool”) and the rest are numerical.

4.1 Missing values

While exploring the data, it has been observed that the missing values present in the dataset are high. To improve the quality of the dataset, it is important to reduce the missing values from the dataset [**4**]. Missing values affect the performance of the models, so they should be handled accordingly. Based on our analysis, the columns providing a comparison with Expedia and its competitors contained a large amount of NaN values. Most of the columns named 'comp' had more than 50% as missing values. Along with those 8 competitors columns, we observed that both the training and the test set include a lot of missing values, which is summarized in Table 1. The values in Table 1 are presented in percentages of missing values of the column. The last 3 variables are presented as the mean of the 8 competitors. Along with the removal of the columns with NaN values, one of the following approaches can be implemented for the removal of the remaining missing values [**5**]:

1. Using the mean of the values of the column
2. Using the mode of the values of the column
3. Using respective median to fill the missing values of the column
4. Using unique values outside the data range

For the dataset, we opted for the last option. The missing values were replaced with 0, which was a unique value outside the data range, to avoid bias within data from each column. Using this, we filled the missing rows with 0 for the attributes "comp_inv", "comp_rate" and "comp_rate_percent_diff". For "comp_rate" 0 means that Expedia and competitor have the same price for the selected property. For "comp_inv" 0 means that both Expedia and competitor have availability and 0 for "comp_rate_percent_diff" means there is not percentage difference in the price of Expedia and competitor. Additionally, we did a similar imputation for the variable "prop_review_score" that refers to the score of the properties. It had 0.14% percentage of missing values as shown in Table 1. The similar approaches were applied for the test dataset as well.

Table 1: Missing values in training and test set

Missing Values		
	Training set	Test set
visitor_hist_starrating	94.90	94.88
visitor_hist_adr_usd	94.88	94.86
prop_location_score2	21.98	21.93
srch_query_affinity_score	93.58	93.58
orig_destination_distance	32.42	32.43
prop_review_score	0.14	0.14
gross_bookings_usd	97.19	-
mean_comp_rate	77.87	77.75
mean_comp_inv	88.21	76.25
mean_comp_rate_percent_diff	92.25	92.30

4.2 Outliers & Mergers & Transformations

A target variable plays an important role for prediction. In our case such a variable is not provided, so a target variable was created, based on the description of the project and the provided variables. Thus we introduced a new variable called "assign_score" and that was our target variable. This variable is created from two variables of the training set which are: "clicking_bool" and "booking_bool". Each time one of these categorical variables were 1 the "assign_score" variable got a score from the sum of these two columns. 1 for "clicking_bool" that means the user just clicked to see the information and 4 for "booking_bool" which provides the information of users booking a hotel room while 0 is considered when none of the actions are taking place. The similar approach was applied for the test dataset as well. Below in Figure 1 you can observe how the "assign_score" is distributed in the training set.

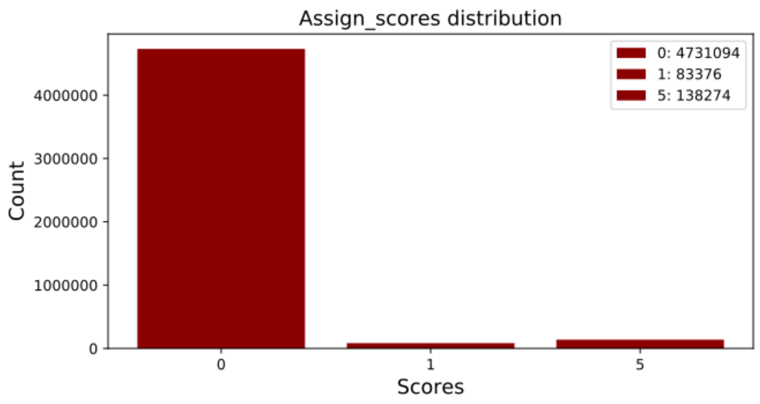

Fig. 1: Bar chart with scores for each search id

After further analysis of the attributes from the dataset provided to us, we observed outliers in the column for price named as: "price_usd". The observed outliers were because the prices are not only costs per night, but also costs for the whole stay. We decided to exclude values that are over \$1800 as we considered them as outliers. Although it not often to see the value of costs per night at \$1800, there are some properties that offer these values in the market [**6**]. After this transformation our training set consists 4.952.744 samples. Figure 2 below shows how the prices are distributed in the training set after the transformation.

Additionally, we performed a number of mergers and transformations both to training and test set to get as much information as possible, that can be helpful to the ranking task. Furthermore we created three new variables, for both sets, with the aggregated values of the competitors variables to reduce dimensionality as the sets consisted of 24 variables with competitor's information. The new variables were "comp_rate", which is the sum among the competitors and positive

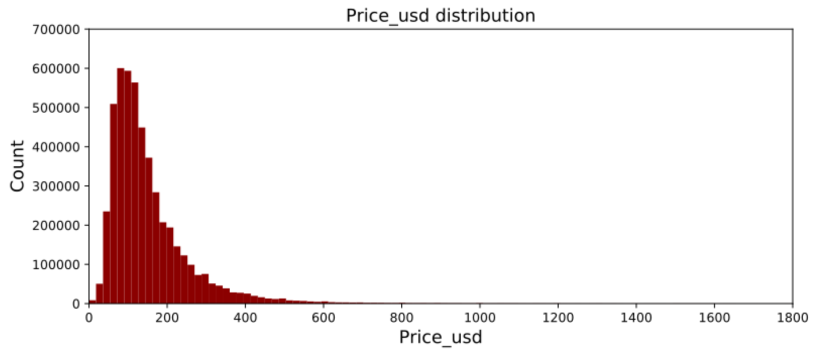

Fig. 2: Histogram of Price distribution after transformation

values mean that Expedia has lower prices than competitors, "comp_inv" which is again the sum among all competitors and positive values mean that Expedia has availability against competitors and "comp_rate_percent_diff" is the mean percentage difference of Expedia against the competitors.

4.3 Formation of Training and Evaluation Set

In this section we will describe our finalized training and evaluation set. After transformations, mergers and filling of missing values we had to decide which attributes will do best for our model. Of course attributes such as "position", "gross_booking_usd", click_bool and booking_bool, that where only on the training set and not on the test set are dropped. Also attributes with a lot of missing values such as "visitor_hist_starrating", "visitor_hist_adr_usd", "srch_query_affinity_score" are dropped. Also "datetime" and "orig_destination_distance" are dropped, because they cannot affect our target variable.

To have a clear overview of the distributions of some important variables in our sets we created histograms that can be seen in Figure 3 below. The distributions of scores of the properties are on a scale out of 5, rounded to 0.5 increments. The star rating of the hotel, from 1 to 5, in increments of 1. A 0 indicates the property has no stars, the star rating is not known or cannot be publicized. The length of stay distribution shows that almost every search has a high length of 10 days of stay, where the maximum value was 57 days. Finally the new variable we introduced earlier, the rates among the competitors. Most of the values are 0, which is a neutral value, where our model can benefit from the other attributes, when value of rate is not 0.

As a last thing, we end up with 24 attributes to train our ranking models. Before the modelling part we split the training set into training and validation set. The training set consists now of 3.466.929 samples and the validation set of 1.485.824 samples. For the modelling part we are going use decision trees, where we have a reconsideration of the importance of each variable to our model.

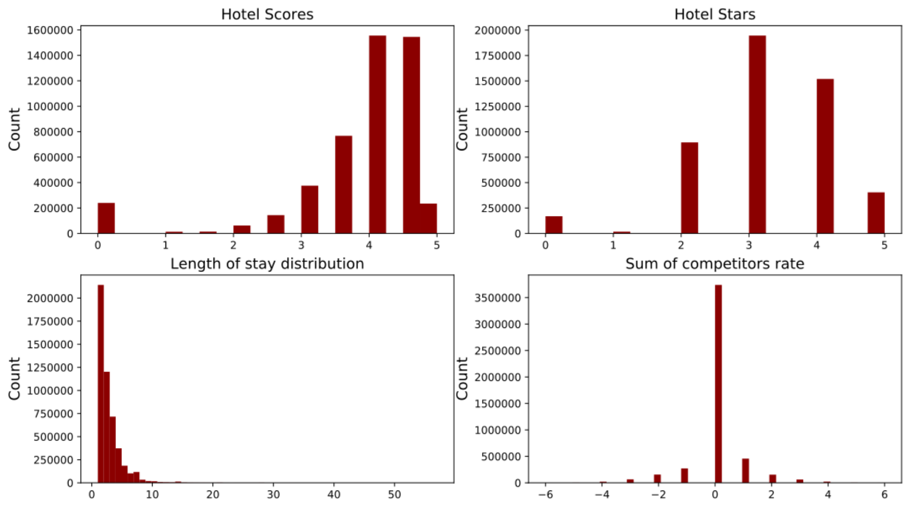

Fig. 3: Histograms of the training set

5 Modelling

After the data preparation was completed, we will now describe the models that we decided to work on to solve the ranking problem. We looked into multiple LTR(Learning to rank) algorithms ideal for our data-set and decided to use the following models. These models are popularly used for multiple Kaggle competitions and are known to have a good success rate. For the training of our models we used a validation set to monitor how the models perform. The selection of our models was based on [**7**], which explains how suitable are LambdaMart and LambdaRank models for ranking problems. LGBM implements LambdaRank while XGBRanker used LambdaMart, which is a better version of LambdaRank. The approach followed for LambdaRank and LambdaMart are a pairwise/listwise approach. Both use maximizing information retrieval functions, like NDCG [**8**], which is our metric based on our description problem.

5.1 LightGBMRanker

LightGBM is an efficient and scalable gradient boosting tree framework. With the help of LightGBM, the training process of conventional Gradient Boosting Decision Tree can be sped up to achieve a high accuracy. This machine learning algorithm is as LTR algorithm. Gradient Boosting Machine(GBM) is used for sequence training. With every iteration, the decision tree is learnt by the algorithm after looking at the residual errors. However, as cited in [**9**], LightGBM provides an updated aprroach over GBM. During traditional frameworks, the growth of the trees are shown per level. However, for LightGBM, the leafs are the concentrated for growth. As discussed in the paper, the ranking algorithm of LightGBM works as an effective ranking function. This provides the option to understand the progress of the training set in comparison to the NDCG score of the validation set. To provide an efficient algorithm, we chose the hyperparameters carefully. The hyperparameters were chosen with a trial method to understand the increase the efficiency of the algorithm. The hyperparameters used for this algorithm are in Table 2. We experimented with the parameters and tuned the learning rate the number of estimators and the number of leaves, which are important for our model. We choose subsample: 0.8, that means that this will randomly select 80% of data without resampling. Also we choose to enable the bagging option by setting subsample_freq= -1. The rest of the parameters are set to default.

Table 2: Parameters used for LGBMRanker

LGBMRanker	
Parameters	Values
learning_rate	0.1
n_estimators	120
subsample	0.8
subsample_freq	-1
num_leaves	150
objective	"lambdarank"
score	"ndcg"

5.2 XGBoost

XGBoost which is also known as Extreme Gradient Boosting, is a decision tree based ensemble Machine Learning algorithm. It is easy to implement and the level of accuracy achieved with this model is high. In this algorithm, Boosting is used to create ensembles of learners, where each new one is designed to decrease the errors made by the previous learner. Based on This process can be implemented by using bagging were new samples are constantly created to highlight the errors of the previous samples while assigning different weights to new learners [**10**]. Thus, XGBoost differs from other commonly used ensemble algorithms like RandomForest. XGBoost has a specifically designed model used to rank items known as XGBRank. Similar to LightGBMRanker, XGBRank uses validation set to monitor the learning progress.When the algorithm detects that no further improvement in score of (NDCG) validation set can be implemented by increasing the model fitting, then the algorithm terminates [**6**]. Similar to LGBMRanker, we use multiple hyperparameters for this alogrithm, which are in Table 3. In this case we tuned the number of estimators the learning rate and gamma, which is the minimum loss reduction required to make a further partition on a leaf node of the tree. Also here we chose subsampling and for the objective we used the "rank:pairwise", which produced slightly better results than the objective "rank:ndcg". The rest of the parameters are set to default.

Table 3: Parameters used forXGBRanker

XGBRanker	
Parameters	Values
learning_rate	0.1
n_estimators	120
subsample	0.7
gamma	-1
num_leaves	1.0
objective	"rank:pairwise"
max_depth	8
eval_metric	"ndcg"

6 Feature Selection

Previously, we explained the features that we created and removed for while exploring the dataset to understand and work with the data better. However, using all the features for both of our models will be unnecessary as it will not yield us the best results. Thus, it is essential to understand the ideal features that can be selected for our model. Feature selection is important as it can reduce unnecessary noise created. Selecting the right features will help to reduce overfitting, training time while increasing accuracy of the model. For this process, we use Decision Trees own evaluation process. Decision Trees have their own evaluation about the importance of each feature. We used this process to extract details about how our models processed each feature. Based on our findings, the most important features for both of our models are "price_usd", "prop_log_historical_price" and "prop_id". While we found the important features, we also excluded the least important features, which are "srch_saturday_nig ht_bool","competitor_inv", "srch_room_count","promotion_flag". However, we created two attributes "competitor_rate" and "competitor_rate_percent_differ", which considered as important features for our models.

7 Evaluation

To evaluate the performance of our primary model,the LGBMRanker and also to confirm the prediction capability, we performed cross-validation. Cross validation is a statistical method of evaluating generalization performance. We used a 5-fold cross validation with the following parameters: objective: lambdarank, num_leaves: 150, score: ndcg, learning_rate: 0.01 and num_boost_rounds: 120. These are exactly the parameters we used also for the training of our model. In Table 4 we summarized the scores of cross-validation. These are the best average scores after every boosting round. We expect from our model to perform and generalize in unseen data according to these scores.

Table 4: Cross-validation for LGBMranker

Cross-validation	
	LGBMRanker
score_1	0.20
score_2	0.24
score_3	0.27
score_4	0.31
score_5	0.35

8 Results

As our final model, we used LGBMRanker with the hyperparameters as mentioned earlier and compared with XGBRanker. We have created Table 5 below showing the results from each training and test set that we got:

Table 5: Results on training and test set

Scores		
	LGBMRanker	XGBRanker
training_set	0.78	0.88
training_set_feat_remov	0.78	0.89
test_set	0.359	0.356

9 Discussion and Future Work

In this section, we will make a quick recap of our models and the algorithms we worked on and we will also discuss new solutions and how they can be implemented in the future.

1. We have constructed two models, both models are based on decision trees. We implemented two similar models, but they are using different implementations. They differ as one model makes use of LambdaRank and the other one of LambdaMart. Both models are used in ranking tasks with great results, something that was anticipated also on this ranking task of Expedia.

2. For improvement of our models we suggest of reconsideration of attributes used for training and evaluating the models. Creating new attributes based on "prop_id" and calculating mean and standard deviation of numerical values could be easily be tried in a future project. Changing the parameters maybe be helpful for small improvements.

Based on our experience, LTR algorithms are used less compared to that of classification and regression algorithms. Due to this, enough information is not provided online. We spent a sizeable amount of time understanding workable LTR algorithms and how to effectively work with them. As further research, new models can be constructed except from decision trees to evaluate the performance of different problems in the same task. Such models can be, for instance, Neural Networks, which can extract higher possible solutions of this problem as they are based on non-linearity between the features.

10 Conclusion

Our decision trees approach seemed to work well based on the results we obtained. Both the ranking algorithms perform almost similary, that made us thinking that focusing on improving just one of them would lead us to better results. In the Kaggle competition the best score was at 0.54%, that means that there is room for improvement on our models.

References

1. Gislainy Laise da Silva,Luiz Mendes Filho,Sérgio Marques Júnior *Analysis of the Perception of Accommodation Consumers on the Use of Online Travel Agencies (OTAs)*
2. Owen Zang *Personalized Expedia Hotel Searches – 1st place*
3. Jun Wang and Alexandros Kalousis *Personalized Expedia Hotel Searches – 2st place*
4. Hyun Kang *The prevention and handling of the missing data*
5. P Royston. *Multiple Imputation of Missing Values. The Stata Journal*
6. https://www.thrillist.com/news/nation/cost-of-5-star-hotels-around-world-2019-chart
7. @inproceedings,Christopher J. C. Burges *From RankNet to LambdaRank to LambdaMART: An Overview*, 2010
8. Hamed Valizadegan, Rong Jin, Ruofei Zhang, Jianchang Mao *Learning to Rank by Optimizing NDCG Measure*,January 2009
9. Guolin Ke,Qi Meng,Thomas Finley, Taifeng Wang,Wei Chen,Weidong Ma, Qiwei Ye,Tie-Yan Liu *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*
10. Tianqi Chen,Carlos Guestrin *XGBoost: A Scalable Tree Boosting System*