Data Mining Techniques - Assignment 2 Scientific Report

Yannick Hogebrug (2625424), Dominique Hopman (2624150), and Jesse Schouten (2621562)

VU Amsterdam

1 Introduction

The dataset originates from Expedia, which is a company that focuses in search engines and websites in the traveling industry. The task is to predict what hotel a user is most likely to book. This could help Expedia to organize the search results in a suitable way, which can result in improved clicks and bookings of users on the website. The available dataset contains data about a search query of a user, which is basically a list of hotels that was seen in a session

Section 2 contains the data exploration, with the steps of pre-processing and Exploratory Data Analysis (EDA). Section 3 treats the models that were tried/used and also the metric that was used for evaluation. Finally, section 4 provides some conclusions about the obtained results and also gives some points for discussion.

1.1 Related work

This section contains some related work about the problem. The original dataset originates from the Kaggle competition of Expedia in 2013 [1]. On these Kaggle page, a lot of information can be found about approaches of competitors. For example, the approaches of the top 3 of the competition were made available. They described e.g. which features were important and what models they successfully used.

One of the models they all used was LambdaMART, which is a model developed by Chris Burges and his colleagues at Microsoft Research (2010). It combines LambdaRank and MART (Multiple Additive Regression Trees). They both make use of gradient boosted decision trees for on the one hand prediction tasks and on the other hand for solving a ranking task [2]. Also, a big advantage of the LambdaMART is that it is computationally efficient, which is a really important property for the big dataset.

A similar ranking problem is the Yahoo challenge from 2010. This challenge contained two datasets used internally at Yahoo! for learning the

web search ranking function. They also made primarily use of the LambdaMART algorithm [3].

Some inspiration for the importance of the different variables was drawn from the approaches of the winners. Some variables which were mentioned by them were e.g. the hotel quality estimation, price_usd (displayed price), prop_location_score2 (score outlining the desirability of the hotel's location) and estimated position. The latter one was used by the winner and is a feature that is only available in the testset[8]. In the next section we will come back to this on how we approximated this position.

2 Data exploration

In this section we will discover the dataset, using the information we discussed in the previous section. First, a brief explanation of the dataset is given. Secondly, some exploratory data analysis was done to explore the dataset and to determine the important factors in the data. Finally, we discuss the steps we took in pre-processing the data. All figures shown in this chapter are based on the data in the training set.

2.1 Data description

The dataset contains information about a search query of a user for a hotel, the hotel properties and for the training set, whether the user clicked on the hotel and booked it. For the training set there is data available of 54 different variables, for the test set there are only 50. Each line in the dataset represents a combination of a search query by a user with one specific hotel property that was shown as part of the search query.

2.2 Exploratory data analysis

There are 199.795 different search id's and 129.113 different property id's, which is the ID of a hotel. Each search id get on average a list of 25 hotels presented. A decent number of variables have a large number of missing values: visitor_hist_starrating, visitor_hist_adr_usd and srch_query_affinity_score have over 90% missing values, and for comp[x]_rate, comp[x]_Inv and comp[x]_rate_percentage_diff the number of missing values varies between 52% for comp5_inv and 98% for comp6_rate_percent_diff for example. So this is something that we payed attention to. As we are not going over all exact numbers for the missing values, we mention those later on in the report when relevant, as well as the ways we dealt with those missing values.

The left plot of Figure 1 shows when a user may or may not click on a hotel and the position of the hotel at the resulted list. The figure shows that users with a lower position are more likely to click on a hotel and also book a certain hotel. The latter is shown in the right boxplot of the figure. While looking at this figure, we see that the median for both the click bool as the booking bool lays around position 5. Later on, we will construct a separate model to estimate this position and use the results from this boxplot.

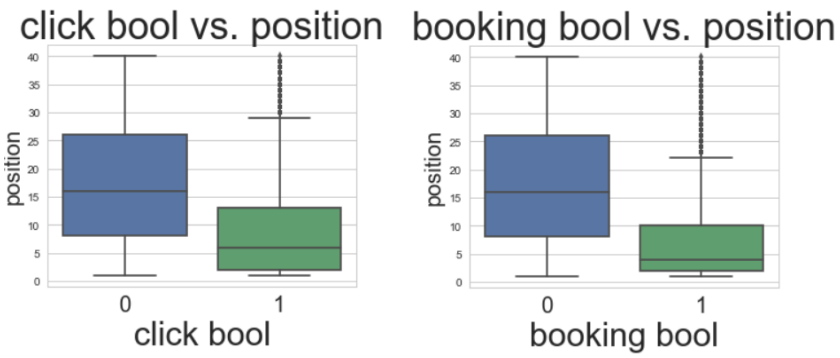

Fig. 1. Position when a user clicked/booked the hotel

Figure 2 shows a correlation plot with Pearson correlation coefficients between all features that are not clearly categorical, such as id's. Also ordinal features, like the position, are not included in the correlation plot. Pearson only works for two continuous interval/ratio features or one dichotomous feature and a interval/ratio feature, also known as point-biserial correlation (Tate, 1954)[5]. Because of the large dataset, we assume normality as well for all continuous variables in the plot despite the fact some variables have a large number of missing values. We are mainly interested in correlation between the features and gross_booking_usd, booking_bool, click_bool and position. Though, position can be seen as an ordinal feature, and therefore the Pearson correlation coefficient is not appropriate. For click_bool and booking_bool, we only see some weak correlations. We see that prop_location_score2 being the furthest away from 0 with a value of 0.07 for both click_bool as booking_bool. For gross_booking_usd we see only one variable with a reasonable correlation, with a coefficient of 0.44, which is srch_length_of_stay. For position, we calculated Spearman coefficients, which resulted in only weak correlations with -0.22 for prop_location_score2 being the value most far away from 0.

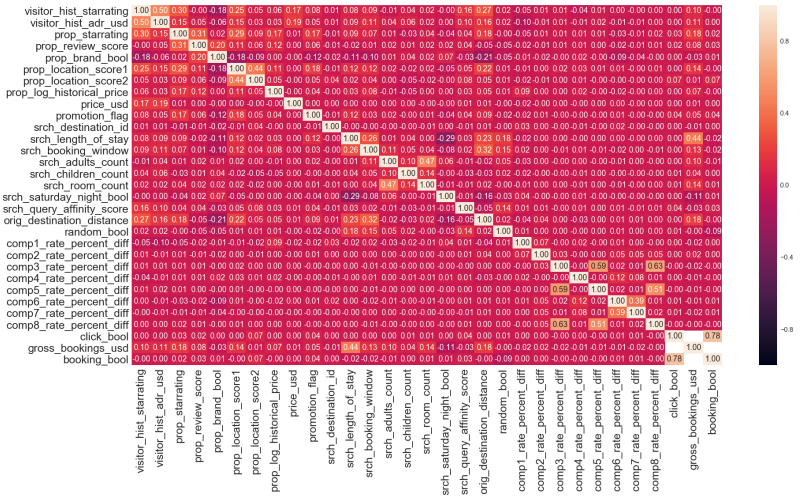

Fig. 2. Correlation plot for continuous variables

Based on the context, we expect price to be an important feature in some way or another: for example the relative price compared to other hotels will most likely influence whether users book or not, as well as the absolute price of a hotel as the general population won't have thousands of dollars to spend on a hotel. Figure 3 shows the probability of booking and clicking for '*price_usd*' intervals of equal length. We see a steady decrease in clicking and booking probabilities from hotels over 200 dollars. Second place winners Wang and Kalousis (2013)[6] described this as a monotonic utility, from which we can expect the variable to have some predictive power.

Figure 4 shows the distribution of the prop_location_score1 and prop_location_score2 with respect to the target variables. We see *prop_location_score*2 differs quite a lot for the different outcomes in the target variable, indicating a high score increases booking and clicking at the hotel. This might be an indication that the variable has some predictive power. *prop_location_score*1, from which we show no figure in this report, turned out to have very similar distributions across both groups, indicating less predictive power. We thought this was a pity, as *prop_location_score*1 has no missing values, while *prop_location_score*2 has approximately 22% missing values in both the training and test set.

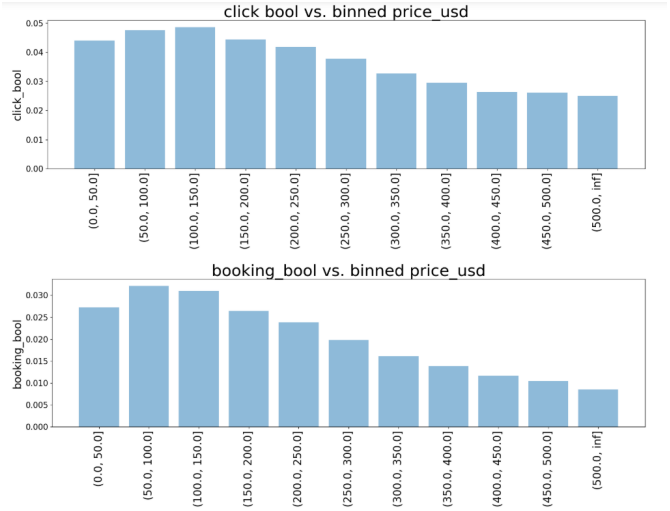

Fig. 3. click and book probabilities for price usd bins

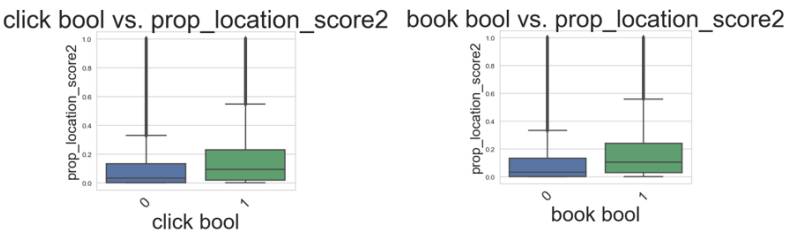

Fig. 4. Prop_location_score2 for booking_bool and click_bool

2.3 Data pre-processing

As discussed in the data description the raw dataset consists of a row for each hotel property and search query. There can be multiple rows per search query because a list of hotels is presented to the user.

Based on intuition, data analysis and literature research, we added some features to the dataset, namely:

- *Relevance_grades* (dependent variable): gives a 5 when a user booked a room at this hotel, 1 when a user clicked on the hotel to see more information and 0 otherwise. This new feature combines the two target variables of booking bool and clicking bool to one variable.
- *Avg_price_propID*: average price per hotel.

- *Std_price_propID*: standard deviation in the price per hotel.
- *Avg_price_srchID*: average price per search query.
- *Avg_price_srch_destinationID*: average price of the destination where the hotel search was performed.
- *price_usd_position*: rank of the price for the search id.
- *price/prop_starrating*: price divided by the star rating of the hotel.
- *compx_rate * compx_rate_perc_diff* (x from 1-8): variable which expresses the price differences between Expedia and their competitors in either a positive or negative percentage.

The missing values of the different features are filled with -1 by default, so the machine learning algorithms know that something is going on with these values. Only price/prop_starrating and prop_review_score are filled with 0 because either it indicated a worse case scenario for price/prop_starrating or because it made sense for prop_review_score as there where not that many (0.14%) and because it yields the same value as having no reviews at all.

3 Models

In this section we will describe the models we used and how we evaluated them, using the NDCG_5-score. The available training data was split in 70/30% srch_id's train/validation for model evaluation.

3.1 Evaluation metric

For this ranking problem, NDCG-5 is used as the evaluation metric. NDCG is a normalization of the Discounted Cumulative Gain (DCG) measure [7]. NDCG has two advantages compared to many other measures. First, NDCG allows each retrieved document has graded relevance while most traditional ranking measures only allow binary relevance. That is, each document is viewed as either relevant or not relevant by previous ranking measures, while there can be degrees of relevancy for documents in NDCG. Second, NDCG involves a discount function over the rank while many other measures uniformly weight all positions. This feature is particularly important for search engines as users care much more about top ranked documents than others. The formula for the DCG is:

$$DCG_5 = rel_1 + \sum_{1}^{5} \frac{rel_i}{log_2(i+1)}$$

and for NDCG:

$$NDCG_5 = \frac{DCG_5}{IDCG_5}$$

with $IDCG_5$ the DCG of the ideal ordering.

3.2 Failed efforts

From the data analysis we found *prop_location_score*2 to be an important variable, as opposed to *prop_location_score*1 which did not seem to be a very good variable. Basically, *prop_location_score*2 had around 20% missing values, while *prop_location_score*1 did not contain missing values. We tried imputing the missing values with a linear regression, seeing the missing values as the target value for the regression problem, rather than imputing the missing values naively with the value -1. First note we deliberately chose this value outside of its natural domain to let the model know something is up with these values. A number of variables were included based on linear correlation, *prop_location_score*1 being the strongest correlated with a Pearson correlation coefficient of approximately 0.4 as shown in Figure 2. Using the imputation with linear regression made the performance of the model a lot worse than imputing with -1 values. An explanation might be that linear regression did not really seem a very appropriate model, given the plots we saw. Though, we still expected it to be better then the simple approach. Other models were not considered, as a fast and easy approach was desired.

We tried an encoding scheme popularized by the long time Kaggle number 1 Owen Zhang, who also won this competition. The scheme is called Leave-One-Out Encoding, which is basically taking the mean of the target value for (high cardinality) categoric variables, while the target variable is left out of the mean while imputing the values in the training set. It is highlighted by Zhang in [8]. As prop_country_id, visitor_country_id and site_id were such categorical variables with insignificant importance to the model, we tried adding the leave-one-out encoded. This did not add any value to the model, and due to the fact it cost a bit of time to apply such an encoding scheme, we kept on using the original variables.

3.3 Different methods

Our very first submission was a Random Forest (RF) fitted on the '*booking_bool*' as the target variable, considering it as a classification problem. We used a few selected variables based on limited data analysis and intuition, namely 'price_usd, prop_log_historical_price' and all compx_rate_perc_diff. So basically price and price position compared to competitors. We used the predicted probabilities resulting from the model to rank the items in the query and rank the items. For the second 'improved' baseline we used a RF with the actual 'relevance scores' (0 if not booked or clicked, 1 if clicked and 5 if booked) as target variable, considering it as a regression problem. We know this is not entirely correct, but it improved the score a bit, though it did not get us past 0.3. We used the same features as meant before with in addition the variable *prop_location_score*2. We droppen this approach in our final model.

For the final model, the hyperparameters were selected by applying a random search with 5-fold cross validation with the package 'RandomizedSearchCV' in python.

After iterating a given number of times, the 'best' performing model on the validation set was used to make final predictions for the testset. The RF algorithm has multiple hyperparameters to tune. Probst, Wright & Boulesteix (2018) described some hyperparameters which could be of influence for the prediction performance [4], like number of trees in the forest, number of features to consider when looking for the best split and minimum number of samples required to be at a leaf node. The 'best' model used as values for these parameters 100, number of features/3 and 5 respectively. This gave us a submission score of 0.2767.

From that moment, we ended up using LambdaMART, which is considered to be the current state of the art ranking algorithm. As we already mentioned in section 1, it was also frequently used in the past kaggle competition.

3.3.1 Estimating the Position We mentioned the 'position' variable before, which we concluded to be an important variable most likely. While running the LambdaMART on the validation set, we saw that the position variable really improved the score. Because this variable is not present in the testset, we decided to fit a separate model for estimating the position only on instances with rand_ = 0, as we assumed by definition that the positions of the instances with rand_bool = 1 are random. We also did this with the help of LambdaMART. For choosing the rank parameter, we used the information of the boxplot of Figure 1. Because the median of both boxplots lays around 5 for the users that did click/book and also this median has zero overlap with the interquartile distance of the other boxplot in the figure, we set this parameter to 5. This led to a $NDCG_5 - score$ of 0.6516 with 2000 training rounds. Here, we used a subset of the variables in the final model. These features will be discussed further on in this section. After fitting this model, we predicted the position for the trainingset and testset, so we could use this variable for the final model.

3.4 Final model

The LambdaMART algorithm ended up as the final model. Based on exploration, the max_depth parameter seemed to be the most important parameter. While tuning this parameter, we found out that it quickly tended to overfit, while raising the value of this parameter. In the final model, we used all features, except the compx_inv variables, with x from 1 up to 8. These features have a lot of missing values and we saw little

impact after imputing the missing values with an extreme value. So, we decided to exclude these variables from the model. The final value for the max_depth parameter which gave the best results equals 5.

We also tried to exclude certain variables from the model, but this made the results worse. So, we decided to include all the variables, except the ones mentioned above, in the model.

3.5 Results

As we described earlier, we started with applying the RF algorithm. Then, we moved on using the LambdaMART algorithm, which is specialized in ranking. During this part, we kept on trying to add new features, which was mainly based on the approaches of the top 3 of the available kaggle competition. In the last part, we focused on optimizing the hyperparameters and trained as long as possible, while being aware of overfitting. The table below shows the exact results of the different algorithms.

	$NDCG_5$
First Baseline (RF, classification)	0.1741
Improved Baseline (RF, regression)	0.2767
Final model (lambdaMART)	0.3833

Table 1. $NDCG_5$ for each model throughout project

4 Conclusion and discussion

4.1 Conclusion

From the data analysis we can conclude that price, position and prop_location_score2 are important features. Based on that conclusion we added some new features that worked fine. Different models were tried and evaluated to set-up a final approach. First, LambdaMart was used to estimate the position and later on used in the final model.

4.2 Discussion

There are a number of ways the model could be improved for better accuracy. Our approach was basically to throw all variables that seemed relevant into the same model. We saw that it was quite common for people in the competition to group features logically, fit different models and combine those models for the eventual prediction. Winner of this competition, Owen Zhang (2013), can be taken as an example as he divided

the feature into five groups, fitting 26 Gradient Boosting Machine models using four model types [8]. As concluded earlier, removing variables yielded worse results, so this was no option. Another thing we could have done, is to take an average of the final model and other models (such as the baseline model), as an average across different model can yield better results. Also, Zhang (2015) concluded that downsampling negative instances resulted in better predictions (and less training time) as well, which was something we didn't try [8].

4.2.1 What we have learned This assignment was our first time working with a ranking algorithm, which was definitely really interesting. Besides that, it was also the first time stacking different models. We estimated the position with a model and used this as a feature in the final model. We learned a lot from this. Furthermore, we used one-hot encoding a lot beforehand, but during this assignment we also spent some time reading and applying Leave One Out Encoding, which is a pretty new technique. Although it did not help quite much in our algorithm, it is really something that can be useful in other situations. Also, the LambdaMART algorithm did not seem to be very commonly used, as we could find no examples of people implementing this algorithm in Python. So, we basically only worked with the documentation of the XGBoost library, which was also interesting to do, as usually blog posts or something similar provide a lot of insight to how a library can and should be used in Python.

References

1. Personalize expedia hotel searches - icdm (2013), https://www.kaggle.com/c/expedia-personalized-sort
2. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
3. O, Chapelle, Y.C.: Yahoo! learning to rank challenge overview (2011)
4. Probst, P., Boulesteix, A.L., Wright, M.: Hyperparameters and tuning strategies for random forest (04 2018)
5. Tate, R.F.: Correlation between a discrete and a continuous variable. point-biserial correlation. The Annals of mathematical statistics **25**(3), 603–607 (1954)
6. Wang, K.: A. personalizing expedia hotel search - 2nd place (2013), https://www.dropbox.com/sh/5kedakjizgrog0y/_LE_DFCA7J/ICDM_2013?preview=4_jun_wang.pdf
7. Yining Wang, Liwei Wang, Y.L.D.H.T.Y.L.W.C.: A theoretical analysis of ndcg ranking measures (2013)
8. Zhang, O.: Winning data science competitions (2015), https://www.slideshare.net/ShangxuanZhang/winning-data-science-competitions-presented-by-owen-zhang