Data Mining Assignment 2 Hotel Search

Yorick Mengelers[1][2557326], Ankur Anmol[2][2701807], and Qiao Ren[3][11828668]

[1] Vrije Universiteit, Amsterdam, the Netherlands y.mengelers@student.vu.nl
[2] Vrije Universiteit, Amsterdam, the Netherlands a.anmol@student.vu.nl
[3] Univeristy of Amsterdam, Amsterdam, the Netherlands qiao.ren@student.uva.nl
group number: DMT-2021 160

1 Introduction

The goal of this task is to predict the ranking of hotels, with a given user search. The more likely a hotel is booked, the higher it is ranked in the result. The data set is Kaggle Expedia Hotel data. Our pipeline is :

1. feature engineering
2. building up models:
   (a) regression models: linear regression, linear SVR, random forest regressor
   (b) classification models: SVM, random forest classifier, logistic regression
3. prediction and generating ranking result
4. evaluation

2 Business Understanding and Related Work

We found previous works in Kaggle Competition Personalized Expedia Hotel Searches ICDM, in Dec, 2013 [1]. The most interesting work is done by the 1st and 2nd placed team.

1st place in the competition Owen's team uses gradient boosting machine (GBM) as the predictor and NDCG as the loss function. Boosting is a method of converting weak learners (typically, a decision tree) to strong learner (a gradient boosted tree) [2]. GBM is based on boosting. GBM identifies the shortcoming of a weak learner by using gradient in the loss function. GBM allows one to optimise a user specified cost function, instead of a loss function that usually offers less control [2].

The approach of Owen's team contains preprocessing and feature engineering. Preprocessing contains 3 steps: 1)impute missing values with negative values 2) bound numerical values 3) down sample negative instances. In feature engineering, the team construct composite features, estimated the positions and uses EXP features. EXP features is a method which convert categorical features into numerical features. Each category in a categorical feature is replaced with an average of the target variable (booked/clicked) related, excluding the current observation. Estimated position is an EXP feature computed from property id, destination id, target month, and the position of the same hotel in the same destination in the previous and next search. They found out that : The most important features are: position, price, location desirability.

2nd place in the competition Jun Wang's team trained linear models and a non-linear model. Their non-linear model, which is LambdaMART, makes the best performance. So their final predictor is LambdaMART. LambdaMART is a combination of LambdaRank and MART (Multiple Additive Regression Trees) [3]. 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 [3].

The approach of this team is 3 steps: 1) missing values were filled with worst case scenario. The missing values of competitor descriptions were all set to zero. 2) Feature extraction: They extract users' historical data. They extract hotel quality to estimate the probability that each hotel is booked or clicked. They also extract non-monotonicity of feature utility, because some features have non-nomotonicity utility functions. 3) Feature normalization: they normalize hotel and competitor descriptions with respect to different indicators.

3 Feature Engineering and Data Processing

We applied following techniques to perform feature engineering. It fufills task 2 data understanding and task 3 data preparation.

3.1 Label Construction

The task was to predict which properties within one search id are most likely to be clicked or booked and rank them accordingly. The labels were set according to the following rule (after specifying this, the original boolean target values were dropped.):

- label = 5 if booked
- label = 1 if clicked but not booked
- label = 0 if not booked and not clicked

3.2 Removal of Outliers

It is necessary to detect and remove outliers from the dataset, because the data set was collected automatically without manual inspection. We use python functions to check whether or not there exist odd max, odd min or odd average values in the features. If outlier exist, we remove them. We did this in all the binary and numeric features. There are 8 numeric features: gross booking, distance to destination, price and the percentage price difference compared to competitors.

Price feature contains extremely high values. Fig. 1 shows the distribution of prices and price rate differences for hotels and the competitors. Fig. 2 shows that some of the percentage price difference compared to competitors for some instances are above 100. This is of course very unlikely and all instances with a

price difference of more than 50% were removed from the training set.

The given price feature also shows some extreme values of more than 100 000 USD a night, raising the suspicions that these values are system errors. Fig. 2 shows that the extremely high values were mainly collected on two days, enhancing the measuring error suspicion. We searched for the most expensive hotel in Expedia and found that it was around 10 000 USD a night, so all instances above this value were removed from the test set.

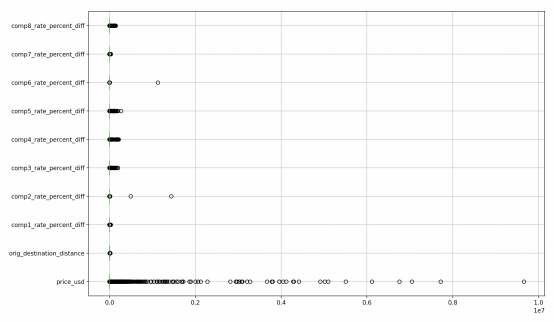


Fig. 1: The distribution of values for the hotel prices and the competitors price rate differences.

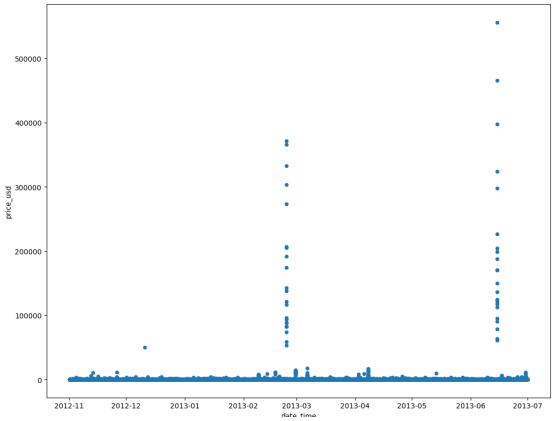


Fig. 2: the distribution of property price with respect to date time (year-month). the blue dots which are far away from most of the data points are the outliers

3.3 Fill In Missing Values

Fig. 3 shows all the features which contain missing values as a percentage of all the data in the train set. The features which contains large amount of missing data are related to the comparison with competitors. Intuitively, the "competitor rate percentage difference" is the most important feature. Unfortunately, the feature does not show whether the price difference is positive or negative. This makes it not informative, based on our inspection on the normal distribution of the feature. To create a new more informative feature, "relative price difference" was created. The new feature "relative price difference" is computed by multiplying the "comp _rate_percentage_diff" with the "comp_rate". (Then the two original columns were dropped.)

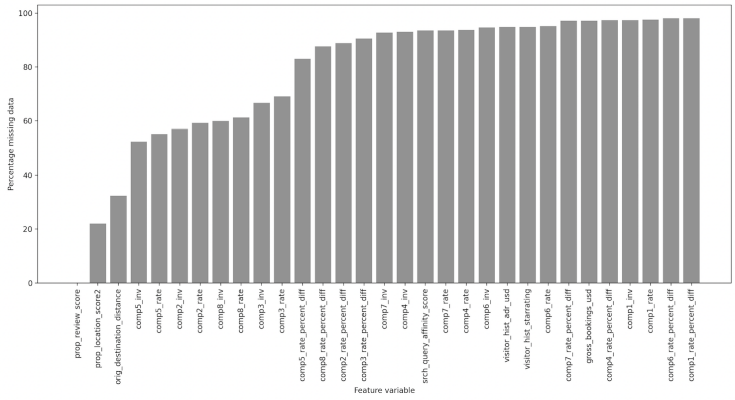


Fig. 3: the percentage of missing data in the features in the train set

To see if the price compared to competitors has an influence on user booking behaviour, the percentage relative price difference of competitor 5 was plotted against the booking and clicking chance (Fig. 4). Competitor 5 was chosen because it contains the least amount of missing values, which makes it the most informative. Fig. 4 shows that the clicking and booking behaviour depends on the relative price difference compared to competitors. So it is important to keep this feature. Also, Fig. 4 reveals that at zero percent price difference, the respective booking and clicking percentages were around 2.9% and 4.4% respectively. The overall percentage booked or clicked for all samples was 2.79% and 4.47% respectively. Based on this information, we imputed all missing competitor values with zero. The origin to destination is an important feature (data was not shown) and missing values were imputed by the overall mean. There are very few missing values in the feature: prop_review _score. Earlier researches showed that the lack of having reviews has a big influence on the booking chance. To mimic this lack of review scores, all the missing values were imputed with zero.

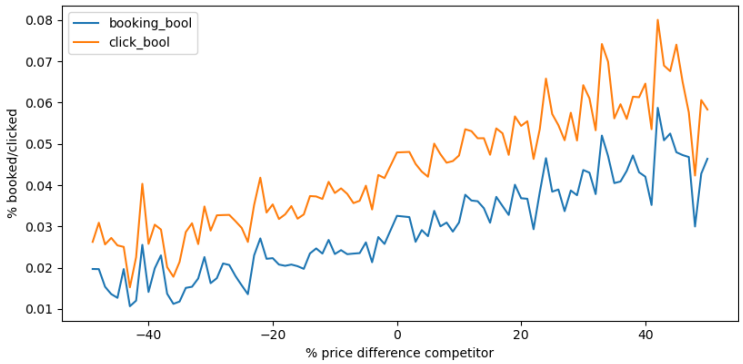


Fig. 4: the price difference compared with competitor 5. Horizontal axis shows how much the price of competitor 5 is higher than this property. 0 means they have the same price. Vertical axis shows the percentage of bookings and clickings. When the competitor 5 has higher price, our property have more bookings and clickings.

3.4 Date Conversion

As mentioned earlier, the feature "relative price difference (in percentage) compared to competitors" was added to the dataset. In the original datasets, the feature date _ time was in the form of an object and could not be used as an input for the machine learning task. Using the pandas function to _ datetime this feature was converted to three numeric features: year, day and month.

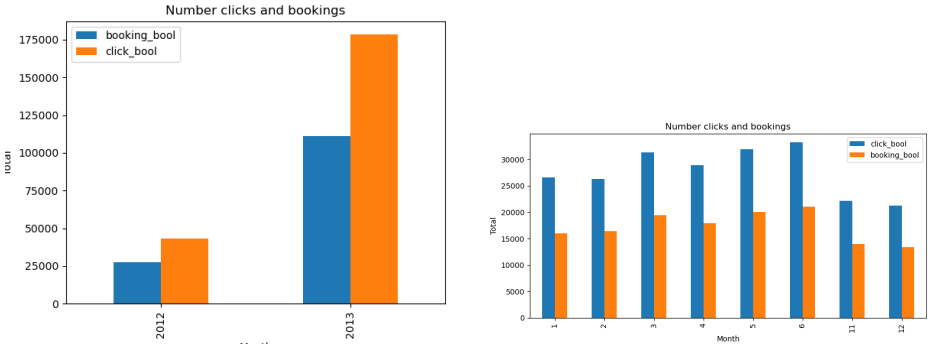
The collection of the data was not equally distributed over the years 2012 (two months) and 2013 (6 months). This leads to a big class imbalance for this feature year with respect to the target value (Fig. 5a) and the column year was dropped. Fig. 5b shows that the added feature "month" did contain useful information. Because the total amount of booking and clicks differ per month.

3.5 Data Normalization

The price was normalized by dividing it by the mean price of the country, in order to make this feature more informative. This is base on the description in the kaggle competition: for the price_usd feature, it says "note that different countries have different conventions regarding displaying taxes and fees and the value may be per night or for the whole stay" .

3.6 Suggested Feature Engineering

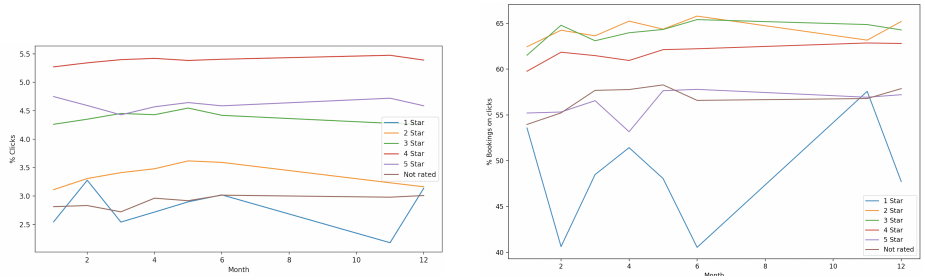
To boost model performance, some more feature engineering steps can be done. This section is about the feature engineering steps that were in the pipeline but



(a) by year (b) by month

Fig. 5: number of clicks and bookings

were not incorporated in the scripts due to time and computational cost saving. Fig. 6b and Fig. 6a show that the booking and clicking rate is dependent on the hotel star rating. Regarding the click rate, most people are likely to click on a high star hotel (4 and 5 stars). But when it comes to the actual booking, the hotels with a lower star rating (2 and 3 stars) are actually more likely to get booked. This causes the order of the star rating with regard to the actual task to vanish. By treating this feature as categorical, for example by one-hot encoding, we could boost model performance.

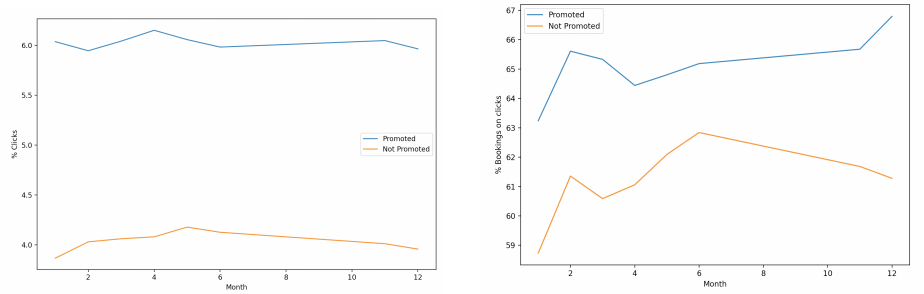


(a) Percentage of clicks (b) Percentage of bookings (after clicked)

Fig. 6: Percentage of clicks and bookings for hotels in various star rating category, with respect to month

Hotel bookings and clicks also relied on promotions. Fig. 7 shows hotels which were promoted received more clicks as well as were more likely to be clicked. The clicks probability increased almost by 50% and there was increase in booking percentage as well (5%). Also, the features: side_id, country_id(s), srch_destination_id and other numerically given categorical features should be

transformed using one-hot encoding turning them into new features. The major drawback is, due to a large number of unique values in most of the categorical features, computational costs would increase dramatically. This increases the running time even further with the large dataset.



(a) Percentage of clicks (b) Percentage of bookings (after clicked)

Fig. 7: Percentage of clicks and bookings for hotels w.r.t to whether they were promoted or not. The percentage of bookings-after-click means percentage of clicks which were converted into bookings

4 Method

4.1 Ranking based on Regression Models

Regression model is used to predict the label for each property. Labels are 0,1,5. We want our predicted values to be as close to the labels as possible. In order to rank the properties, the approach is to sort the predicted value from large to small, for any given search id. The properties with large predicted value are ranked into high position. We use two different approaches to build up a regression model:

- Approach 1 is property non specific, which means the model is trained on all the properties. Approach 1 uses random forest.
- Approach 2 is property specific, which means the model is trained on each specific properties. Approach 2 uses Linear Regression and Linear SVR.

Approach 1 The first approach was to train a Random Forest regression model using all the instances. RF is a supervised learning algorithm that uses an ensemble algorithm to combine multiple decision trees [6]. It builds a forest of uncorrelated decision trees using random sampling and feature selection. It combines tree correlation using a proximity matrix with tree pruning to make the final forest for the regression task [6].

Approach 2 This approach is to train a regressor on every property in the training set. All property-specific models were stored in a dictionary. For each search id, predictions were made for every property in the search list. The final ranking was based on the predicted value made by the regressor.

We come up with this approach because Whether a property is clicked or booked is dependent on how well a property and a search id are matched. To make the feature property a useful categorical feature, one approach is to encode it. This would mean a big gain in the number of features and associated with increasing model complexity.

Unfortunately not all properties in the testing set were in the training set. Properties which were not in the model dictionary were given an value of zero. Because for every unique property, a model needs to be trained, models had to be simple and fast to train. Three algorithms are intuitively most suitable: LinearRegression, Linear support vector machine and decision trees [6]. Other algorithms will probably yield an higher score. Concerning the computational costs, Linear Regression and Linear Svm were chosen for the task. To check if model performance could be boosted, using complexer models random-forest was included in the validation scheme. To reduce computational cost, the hyperparameter of the grid search of three property models were used for validating all model.

The only well performing regressor was random forest. Unfortunately none of our computer had a large enough working memory to store all trained property specific models and last minute change of approach was needed

We decided to implement two computation less costly methods. Because Rf had the highest performance on the data set we desired to stick which it. One was to train an random forest regression model on a subset of the train data. Using the same label as mentioned earlier. As before the property were ranked according to there regression output.

4.2 Ranking based on Classification Models

Classification model aims at predicting the target class and its corresponding confidence score. There are 3 classes: 0,1,5. For each property, the classifier gives not only which class it belongs to but also how confident it is. The ranking result is generated based on two principles: 1) The property with high valued class is ranked into high position. 2) Within each class, the property with high confidence is ranked into high position.

We attempted to make 3 different classification models: 1)SVM 2)Random Forest 3) Logistic regression.

SVM constructs a hyperplane or set of hyperplanes in a high dimensional space, which can be used for classification [7]. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction [6]. Logistic regression converts value to probabilities using logistic function and categorize data into groups based on the probabilities [8].

4.3 Validation

In order to objectively pick the best model for the task cross-validation was used to assess model performance. The training set was split three times (Kfold) into a test and train set and each split was used to calculate the model performance, this is the outer CV loop. Because we chose to go for regression the mean square error was chosen for scoring the models. To find the best hyperparameters Grid search cross-validation was used. The size of the data set made training and evaluation very computationally expensive. To reduce computational cost the training dataset an subsample of 10 000 randomly pitched user id's the data was used for model evaluation. The subsampling was done using the function random subsample.

4.4 Evaluation Metric: NDCG

We use NDCG as the evaluation metric. NDCG (Normalized Discounted Cumulative Gain) measures the quality of ranking result. Because ndcg is widely used in the previous work and in the literatures [9]. NDCG is a normalized sum of the graded relevance values of all results in a search result list [9], as shown in Equation eq1. The parameter p means only the top p items are taking into account.

$$nDCG@p = \frac{DCG@p}{IDCG@p}, DCG@p = \sum_{i=1}^{p} \frac{2^{relevance\_i} - 1}{\log_2(i+1)} \tag{1}$$

5 Result and Discussion

5.1 Ranking based on regression models

We trained the three regression models: SVM, Linear regression and Random Forest. Regression models were evaluated based on their mean squared error. Mean error measures the difference between predicted values and the labels. Mean squared error, is the squared error of the value a model predicts compared to the actual result. Table 1 shows that only random forest was able to predict the outcome. Linear regression and svm were not suited for the task. Unfortunately none of our computer had a large enough working memory to store all trained property specific models and last minute change of approach was needed.

Table 1: negative mean error of 3 regression models

Model	negative mean error
SVM	-173.410 (22270.113)
Linear regression	-90214629668938.281 (21268660984313836.000)
RandomForest	-1.011 (3.223)

To evaluate the models, We take a part of the data from the train set and set this data as self-defined test set. Each regression model made a prediction on the self-defined test set. We coded the ndcg by ourselves. Then we computed the ndcg for each model (Table 2).

Table 2: ndcg of 3 regression models

Model	nDCG@5
LinearSVR	0.134
linear regression	0.138
Random forest Regression	

5.2 Ranking based on classification models

We implemented SVC, random forest and logistic regression for classification. Due to time constraint, we do not have the accuracy of logistic regression. But we have optimized SVC and random forest.

Hyperparameter optimization To find the optimal hyperparameters across validated grid search was used with a split of four. The grid search tries all combinations of a preset set of hyperparameters. By using the most suitable hyperparameters the feature space is tweaked till it generalizes the training set the best. The accuracy and optimized hyperparameters of the best model is shown in Table 3.

Table 3: accuracy and hyperparameters of the best model

Model	accuracy	hyperparameters
Best SVC for classification	0.833	'gamma': 'scale', 'kernel': 'linear'
Best Random forest for classification	0.667	'criterion': 'gini', '$n_estimators'$: 50

5.3 Final Model

The final model was trained on a small subset of the entire train set. It is tested on the entire testset. The function predict_proba was used to predict the probability of an correctly predicted property. It correctly predicted the booking boolean results in the highest kaggle score. To increase the weight (importance) of the booking probability, the clicking probability was squared. The final ranking was based on the result of the multiplication of both probabilities. As is shown in Table 3, the highest scoring model was a support vector machine with a

Linear kernel. For classification, a linear kernel was best to generalize the feature space. The big advantage of using a linear kernel is that it does not increase the dimensionality of the input features, whereas polynomial or radial kernels does. It largely reduces the computational power when training the models and make predictions. Because the number of models needed to be trained has no influence on the computational power needed only the number of samples matthew. We decided to go with the property based model which is trained using a support vector machine with a linear kernel, scaled gamma and an auto learning rate. The ranking was based on the multiplication of the squared clicking probabilities by the booking probability.

5.4 Improvement

In order to improve the model, there are couple of things that could be done. Firstly, about filling in missing values, it is better to fill in meaningful values. We filled in zeros for all the missing values. Zeros make sense for some features, but not for all the features. A better approach is to fill in the worst scenario for all the missing values. Which values to fill in depends on which feature it is. The assumption is that users do not want to click on the property which has a missing values. So we can fill in the worst scenario values.

Unfortunately due to multiple ram memory issues and the time to predict the whole testing set both approaches did not finish in time to get substituted to kaggle. When the result is in that will be submitted to kaggle, but this will be after handing in this report. Next time a little warning would be a useful warning for the time it takes to predict the whole test set.

6 What We Learned

In terms of the skills and knowledge, we have learned various models to do the ranking. Regression models, classification models and Learning To Rank models are all able to perform the ranking task. We learned how to build up these models and how to evaluate these models. We especially learned how to evaluate the ranking result. In feature engineering and data preprocessing, we learned that not all the features are relevant to the result. It is important to find out the most relevant features. This is based on inspecting the meaning of data and the distribution of data.

The main difficulty is that it takes a long time to do the training and prediction, which is out of our expectation. Because the number of data instances is large.

References

1. Personalized Expedia Hotel Searches https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab Dec,2013.

2. Harshdeep Singh: Understanding gradient boosting machines. https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab Nov, 2018.
3. Nikhil Dandekar. Intuitive explanation of learning to rank, ranknet, lambdarank and lambdamart https://medium.com/@nikhilbd/intuitive-explanation-of-learning-to-rank-and-ranknet-lambdarank-and-lambdamart-fe1e17fac418 Jan, 2016.
4. Kofoed,W.: Concepts In Predicting Machine Learning.A conceptual framework for approaching predictive modelling problems and case studies of competitions on Kaggle. Technical University of Denmark. Copenhagen (2013)
5. Breiman, L. Random Forests: Machine Learning **45**, 5–32 (2001)
6. Liaw,A., Wiener,L.: Classification and regression by random forest. R news **2**, 18–222 (2002)
7. Noble,W.: What is a support vector machine. Nature biotechnology. Nature Publishing Group **24**, 1565–1567 (20026)
8. Feng,J., Xu,H.: Robust logistic regression and classification. Advances in neural information processing systems **27**, 253–261 (2014)
9. Wang, Y.,Yuanzhi,Li.: A theoretical analysis of NDCG ranking measures. Proceedings of the 26th annual conference on learning theory **8**, 68–75 (2013)