Expedia Hotel Recommendations

Mengxin Qian, Anshika Srivastava, Zainab Danish, Rui Li

1. Introduction

The dataset for this project was taken from Kaggle and logs customer behaviour of Expedia users over time. Each record includes various features which are listed below. The details of these features can be found in the appendix (Appendix 1.1)

- Date-time
- Site Name
- Point of Sale Continent
- User Location Country, Region and City
- Origin Destination Distance
- User ID
- Source of connection (Is_mobile)
- Is_package
- Check-in and check-out dates (srch_ci, srch_co)
- Age-wise guest count (srch_adults_cnt, srch_children_cnt)
- Number of rooms (srch_rm_cnt)
- Search Destination (encoded: srch_destination_id)
- Type of destination (encoded: srch_destination_type_id)
- Booking (binary: is_booking)
- Hotel Continent, Country and Market
- Cnt (number of similar events in the same user session)
- Response: Hotel Cluster

Project Objective : This project aims to predict what hotel cluster a user will book based on attributes of the search the user is conducting on Expedia.

Size: 37 million rows
Predictors: 22 feature
Classification response: 100 Hotel clusters(categorical response)
Accuracy measure: MAPk

Expedia has in-house algorithms to form hotel clusters, where similar hotels for a search (based on historical price, customer star ratings, geographical locations relative to city center, etc) are grouped together. According to the description, there are 100 clusters in total and the algorithm to generate these clusters is not provided. The performance of the model will be scored using Mean Average Precision @ 5 (Refer to Appendix for more details), which means that we'll make 5 cluster predictions for each observation, and will be scored based on which point the correct prediction appears in our list. If the correct prediction comes earlier in the list, we get more points. For example, if the "correct"(actual)

cluster is **3**, then the predicted list of clusters in sequence of [4, 43, 60, **3**, 20] will give us lower score than those in sequence of [**3**, 4, 43, 60, 20].

2. Data

2.1 Data Description

The dataset included a total of 37 million rows spread over 3 years. It was too large to read in with our limited memory, therefore, we chose a 10% subset of the data from 2014. The data was subsetted by 10% proportion for each unique **search destination id** (our most important feature), so that it can represent as much as possible of the information from the original dataset. Same technique was used to split our training data and testing data.

2.2 Exploratory Data Analysis

We felt that a few features weren't helpful enough in their raw form and performed some feature engineering to make them more meaningful and useful. The following features were manipulated:

Raw Features	**Created Features**
check-in and check-out dates	Days before check-in date
check-in and check-out dates	Intended length of stay (in days)
date time	Day of the week, Month and Year of access
location variables	PCA features

Our next step was to perform some EDA to gain some basic intuition about the data. Listed below are some interesting facts.

1. There are a total of 22,447 search destinations.
2. The number of times a destination was searched for is 47 on average, with a large standard deviation of 483, implying that certain destinations were far more popular than others.
3. On average each search destination maps onto 5 different hotel clusters, visibly implying that the hotels have been clustered on factors other than just geographical location.
4. On average only 7.3% of the people who clicked, actually booked a hotel.
5. The peak booking day during the average month is the 15th, and the bookings tend to taper off towards the end of the month.

3. Methods

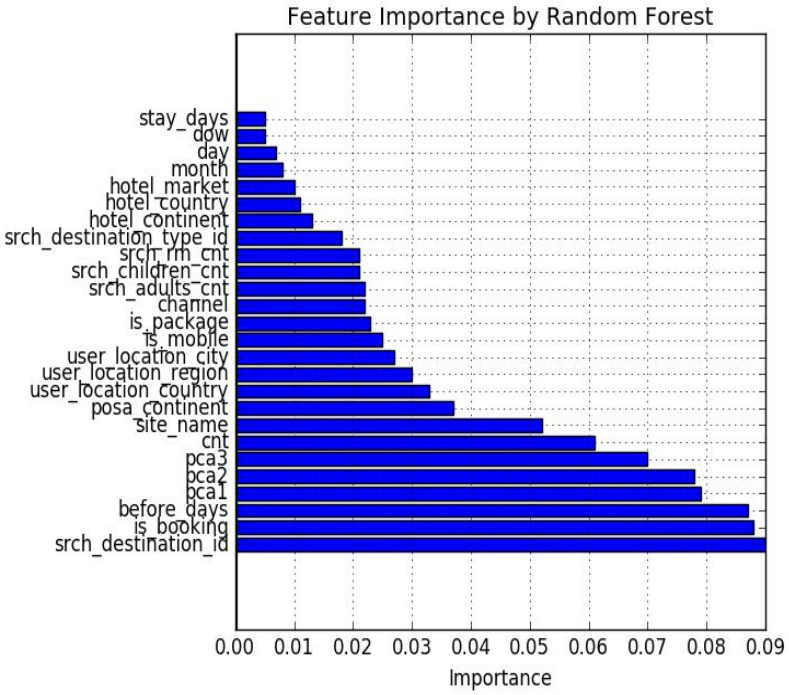
3.1 Feature Selection and Generation

The first step in applying machine learning is to generate features. We had the option of generating features using both training data, and what was available in destinations data. Kaggle didn't tell us exactly what each latent feature in destinations data was. Therefore, we thought it was safe to assume that it was some combinations of destination characteristics, like name, description, and more. Those features were converted to numbers, so they were anonymized.
We first used PCA to reduce the dimensions of the latent features to 3. In this case, we preserved most of the variance in destinations and at the same time, didn't lose a lot of information. Based on our EDA of training dataset, we also generated search year, month, day, day of week from booking date feature from raw training data. We further generated features including how long the customer stayed, and how long the customer searched for the hotel before actual check-in time. Finally, we joined the 3 destination PCA features with the features in training dataset using search destination id.

After generating all the potential features, we used random forest to select the features(**Fig.1**). We selected the top 7 features from here to include in our model.The features are listed as follows:

Fig. 1. Feature Importance by Random Forest

- Search Destination ID
- is_Booking
- Before_days
- PCA1
- PCA2
- PCA3
- Cnt



3.2 Modeling using ML Techniques

Now we have all the features for our model. Since there was no linear relationship between predictor and response variables, we decided to use decision tree approaches including random forest, boosting and bagging. Interesting to note here is that we needed to predict the top 5 clusters for each observation, as compared to 1 cluster as we did in our previous projects. We solved this problem through converting the response variable, which is hotel clusters with 100 unique categories, into 100 binary classifiers. We then

fitted the model using all classifiers as target variables, and extracted the probabilities from the classifier that the observation is in the unique hotel cluster. Thus, for each observation, we had 100 probabilities, one for each hotel cluster. We found the 5 largest probabilities for each observation, and assigned the corresponding 5 clusters as predictions.

Finally, we fitted the data using different machine learning approaches, and we computed accuracy rate using mapk package which is a developed package for Kaggle competition on test dataset. The Machine Learning approaches we used are listed below:

- LDA
- QDA
- Logistic Regression
- AdaBoost
- Gradient Boosting
- Decision Tree
- Random Forest
- Bagging

Before we delve deeper into each approach, it is pertinent to note that each approach was run using either the seven features listed above or only two features namely is_Booking and srch_destination_id. Classification accuracy was computed using the MAPK package from Kaggle.

(a) Logistic Regression:

We started out by taking the naivest of all approaches: Logistic Regression. In logistic regression, we model the conditional distribution of the response Y, given the predictors X. With over hundred hotel clusters to classify the observations into, we expected the approach to yield bad results, but we needed a threshold value just to see how bad things could be. Just as we expected, the 7-feature logistic regression model yielded a classification accuracy of 7.6% and the 2-feature logistic regression model yielded a classification accuracy of 4.7%. We needed a technique with which we could model the distribution of each of the predictor variables separately for each of the classes.

(b) LDA:

While it was naive to assume that the decision boundaries drawn would be linear, we decided to go ahead and try out Linear Discriminant Analysis,for the reason stated above. With a total of 100 clusters and 99 decision boundaries, the model performed just as badly as we expected it to. The classification accuracy for this model was a mere 8.6% with seven predictors. Contrary to our expectations, the accuracy fell even lower (5.2%) when we tried running the algorithm with two predictors only. The most obvious shortcoming of this approach was the linearity of the decision boundaries. We therefore, decided to proceed further with Quadratic Discriminant Analysis.

(c) QDA:

The main benefit that QDA offers over LDA is in the way that decision boundaries are drawn. It does not stick to the naive assumption that decision boundaries are linear. The drawing of quadratic decision boundaries makes the regions more restrictive and accurate. Thus, our main expectation from QDA was an improvement in accuracy rate. Quite surprisingly, QDA performed even worse than LDA in this scenario. The 7-feature model resulted in a classification accuracy of 8.1% and the 2-feature model had a classification accuracy of 4.6%. This made us realize that the problem was much bigger than just the way the decision boundaries were drawn. With over a hundred regions to classify the observations into, the algorithm was bound to perform this way.

(d) Adaboost

Adaboost is a recently developed machine learning technique offering various advantages over the classification algorithms discussed so far. It works by using 'weak learners' - algorithms that are only slightly better than random guessing - to fit repeatedly modified versions of data. The algorithm learns from each successive iteration and focuses on the edge cases. It assigns greater weight to the misclassified observations and goes on iterating till either the classification rate is 100% or it has reached the maximum number of iterations. The number of weak learners is a very important parameter for Adaboost and we found the right parameter after running through several iterations.
We set the n_estimators to 50 and kept the learning rate at one.

Our accuracy rate did improve by over 200%, and went up to 19.3 for the 7 features. Important to note here is that adaboost serves the purpose of reducing dimensions and works best with high dimensional data. Given only two features, Adaboost did improve the accuracy rate by roughly 200% from the naive algorithms, it didn't however, perform as well as we had hoped it would.

(e) Gradient Boosting

Gradient boosting is an extension of Adaboost and works best in scenarios where the variables are of various types. Since we have that kind of heterogeneity in our data, we expected that Gradient Boosting would do well. It performed roughly the same as Adaboost in the 7-feature situation and better than Adaboost in the 2-feature situation. Gradient Boosting works with several trees, so we specified that maximum depth as 3 and learning_rate at 0.1 since it gave us the best classification estimates.

(f) Decision Tree

We decided to use Classification Trees as our final Machine Learning approach in hopes of achieving some improvement on our low accuracy rate. Decision Trees were our first attempt at achieving greater accuracy. Since decision trees follow the greedy algorithm at every split they make, the MSE (classification error in this case) is minimized at every step. We had the depth parameter at our disposal and we tweaked it as needed to maximize classification accuracy. This tweaking cost us a lot of time since the dataset was large. We finally settled on a max_depth of 17. The results of this algorithm were quite surprising. While the decision tree failed to beat the boosting methods for the 7-feature set, it outperformed boosting by roughly 200% on the 2-feature subset bringing our classification accuracy all the way up to 25.9%.

(g) Random Forest and Bagging

A step further from decision trees was to make use of bagging and random forests. Both approaches use B bootstrapped datasets and build several trees.

In bagging, all the features are considered when making a split. In random forests, instead, the split that is picked is the best split among a random subset of the features. This does result in an increase in bias of the forest. However, it is more than compensated by the reduction in variance. With this algorithm, we could tweak the number of trees in the forest. Generally, the greater the number of trees, the better the classification accuracy. However, it also takes longer to compute and might become computationally inefficient after a certain point. We therefore, decided to go with a total of 26 trees. Our results for both bagging and random forests were very similar to those of decision trees whereas we were expecting some improvement. However, we attribute that to the problem of decreasing returns to scale. After a certain point, the classification rate stops improving and that is what we believe happened in our scenario.

Having tried our luck with Machine Learning techniques we decided to conduct some research into alternative means of achieving better classification accuracy. We stumbled upon what we like to call the 'Aggregation Approach' on Kaggle and decided to employ it in order to improve our classification accuracy.

3.3 The Aggregation Approach

This approach relies on the count of historical booking and clicking history. Since the training data is huge, this approach tries to exploit it by predicting the clusters based on frequency the particular cluster was booked for same search criteria. That is, the greater the number of times a hotel cluster say 'H' is booked for say Destination id 'D', the greater probability it has of being selected in the future too. The relevance of clusters is decided by their booking and clicking frequency on the historical data.

The relevance of clicks in determining the popularity of clusters is decided using K-fold cross validation. Click weight acts as a hyper-parameter and we try to tune it in order to get best predictions.

Since the srch_destination_id is one of the best features in predicting hotel cluster and it has 22,447 distinct values, it has a good potential to predict relevant cluster.

Experiments: The hotel clusters were predicted using several combinations of top categorical features. The results are appended in the table below:

Click Weight **Predictor Combinations**	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**	**0.30**	**Average**
srch_destination_id	0.301	0.304	0.305	0.306	0.306	0.308	0.305
srch_destination_id,site_name	0.29	0.293	0.295	0.296	0.298	0.30	0.295
srch_destination_id,site_name,before_days	0.288	0.292	0.294	0.296	0.296	0.296	0.292
srch_destination_id','site_name','month'	0.281	0.282	0.282	0.283	0.285	0.287	0.284
srch_destination_id,site_name,before_days,month	0.281	0.286	0.288	0.291	0.292	0.293	0.287

***** MAPK on validation data(20% of training data)***

As can be seen in the table, srch_destination_id helps in most accurate predictions. Aggregating over other combinations has lower accuracy as the other features seem to only be adding noise to the data.

****Aggregation approach was tested with maximum 4 categorical features*

4. Results

4.1 Results

While we spoke in detail about all of the approaches, the results are clearly summarized in the table below:

Table 4.1 Performance of Machine Learning Algorithms

Type	Algorithm	Accuracy		Hyperparameters
		Feature# =7	Feature# = 2	
Decision Boundary +regression	**LDA**	0.08679	0.05159	
	QDA	0.08171	0.04558	
	Logistic Regression	0.07659	0.04748	
Boosting	**AdaBoost**	0.19283	0.11066	learning_rate=0.1, n_estimators=50
	Gradient Boosting	0.18879	0.16578	learning_rate=0.1, max_depth=3
Trees	**Decision Tree**	0.16148	**0.25906**	max_depth=17
	Random Forest	0.18613	**0.25899**	n_estimators=26
	Bagging	0.18701	**0.25895**	n_estimators=26
Aggregation	**Frequency based**	**0.27*** **	**0.31**	**click_relevance=0.30**

4.3 Feature and model evaluation conclusion:

(i) When using selected important features(with latent variables):

-- Boosting methods performs the best; Trees are slightly worse; LDA,QDA, Regression are the worst.

(ii) When dealing with only two parameters:

-- Only Trees boosted up by 39%~ 63%, having the best performance

-- other two classes of classification methods have lowered their performances up to 50%

5. Conclusion:

1. Simple approaches like aggregation might perform better than classic Machine learning approaches in case of huge size data or similar problems

a. Aggregation approach gave an MAPK of 0.31 whereas traditional machine learning approach gave an MAPK of 0.26.
b. The aggregation approach is computationally cheaper compared to the tree algorithms as it took less than 20 minutes to do the predictions on the test data where training and test data summed to 27 million observations.

2. Parameter tuning is expensive, each model takes 1~3 hours for the 0.1 portion subset data, eg. random forest took 2:48 to tune the estimators from 3 to 30. Our machine could not iterate through as many potential parameters as possible.

6. Recommendation:

1. Explore Collaborative filtering for making hotel cluster predictions for the user. It will involve collecting preferences from many similar users , where users can be grouped as similar users,based on their location and each cluster can be assigned a ranking using this collaboration.

2. Perform aggregations over similar users i.e. use frequency of the booking made by similar users in predicting their preference for given srch_destination_id.

3. Exploit the potential of this huge a data by hosting and running it on multiple-CPU server.

7. Appendix

Appendix 1.1: Feature Details

Column name	Description	Data type
date_time	Timestamp	string
site_name	ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)	int
posa_continent	ID of continent associated with site_name	int
user_location_country	The ID of the country the customer is located	int
user_location_region	The ID of the region the customer is located	int

user_location_city	The ID of the city the customer is located	int
orig_destination_distance	Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated	double
user_id	ID of user	int
is_mobile	1 when a user connected from a mobile device, 0 otherwise	tinyint
is_package	1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise	int
channel	ID of a marketing channel	int
srch_ci	Check-in date	string
srch_co	Checkout date	string
srch_adults_cnt	The number of adults specified in the hotel room	int
srch_children_cnt	The number of (extra occupancy) children specified in the hotel room	int
srch_rm_cnt	The number of hotel rooms specified in the search	int
srch_destination_id	ID of the destination where the hotel search was performed	int
srch_destination_type_id	Type of destination	int
hotel_continent	Hotel continent	int
hotel_country	Hotel country	int
hotel_market	Hotel market	int
is_booking	1 if a booking, 0 if a click	tinyint
cnt	Number of similar events in the context of the same user session	bigint
hotel_cluster	ID of a hotel cluster	int

destinations.csv

Column name	Description	Data type

srch_destination_id	ID of the destination where the hotel search was performed	int
d1-d149	latent description of search regions	double

MAPK Calculations:**

Mean Average Precision @ 5 (MAP@5):

$$MAP@5 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{min(5,n)} P(k)$$

where |U| is the number of user events, P(k) is the precision at cutoff k, n is the number of predicted hotel clusters.

**Source : https://www.kaggle.com/c/expedia-hotel-recommendations/details/evaluation*