Expedia Hotel Recommendation

Anandan Balaji

7 June 2016

Contents

1 Introduction

The objective of **Expedia Hotel Recommendation** is to predict the **hotel cluster** for the user. The hotel clusters are numbered based on many parameters like distance for the city center, amenities like swimming pool, gym etc. They are in the range 1 to 100.

2 Deep Dive into Dataset

2.1 Data Files

The following are the data files provided. They can be accessed from the following *kaggle location* (www.kaggle.com/c/expedia-hotel-recommendations/data).

- **train.csv** - the training dataset
- **test.csv** - the test dataset
- **destinations.csv** - hotel search latent attributes
- **sample_submission.csv** - the sample submission file in the correct format

2.2 Important Fields in the dataset

The following are the important fields which are available in the **training** dataset.

- **date_time** - TimeStamp
- `user location info`
  - **posa_continent** - ID of the continent
  - **user_location_contry** - ID of the country where the customer is located
  - **user_location_region** - ID of the region where the customer is located
  - **user_location_city** - ID of the city where the customer is located
- **orig_destination_distance** - Physical distance between the hotel and customer at the time of search
- `stay information`
  - **srch_ci** - Checkin date
  - **srch_co** - Checkout date
  - **srch_adults_cnt** - Number of adults
  - **srch_childrens_cnt** - Number of childrens
  - **srch_rm_cnt** - Number of rooms requested in the search
- `destination hotel info`
  - **srch_destination_id** - ID of the destination hotel
  - **hotel_continent** - Hotel continent
  - **hotel_country** - Hotel Country
- **is_booking** - 1 if a booking, 0 if a click.
- **cnt** - Number of similar events in the context of same user session.
- **hotel_cluster** - ID of the hotel cluster

Also the **destinations.csv** has the following information.

- **srch_destination_id** - ID of the destination hotel
- **d1-d149** - latent description of search regions

2.3 Format of Submission File

For every user event, we need to predict a space-delimited list of the hotel clusters they booked. we may submit up to 5 predictions for each user event. The file should contain a header and have the following format:

id,hotel_cluster

0,99 3 1 75 20

1,2 50 30 23 9

etc. . .

2.4 Limitations of the Dataset

1. The `user location info` (continent/country/city) and `destination hotel location` (continent/country) are **integer values**. There is no mapping available about the integer values to appripriate cities.
2. The `destination data file` has the information about the hotels in terms of **150 attributes** and they are of **nunerical** value. There is no mapping available for this one as well.

3 Exploring the data

3.1 Sampling the training dataset

The **training** dataset has 37M records with 4GB in size. Because of the huge dataset, we can't load the complete training datset in a normal computer. We need to have a machine with atleast **16GB RAM** size.

However, following are some ways to deal with the big data set for exploration and analysis.

- **sample the training data** - Use CATools package to create a smaller dataset by sampling as below. Note that the sampling will help in the early explorations. But for **final analysis and prediction, we have to run the complete training and test dataset**.

```
# for splitting the training data
library(caTools)

system.time(train_dt <- fread("train.csv", header = TRUE))

#to make it reproducible
set.seed(123)

## specify the column name
split = sample.split(train_dt$hotel_cluster, SplitRatio = 0.75)

new_train_dt = subset(train_dt, split == TRUE)
new_test_dt  = subset(train_dt, split == FALSE)

write.csv(new_train_dt, file = "new_train.csv", row.names = FALSE, quote = FALSE)
write.csv(new_test_dt, file = "new_test.csv", row.names = TRUE, quote = FALSE)
```

- **Use fread()** - The fread() from `data.table` package is faster than the read.csv() function.

Note: For this report, will use sampled training dataset, which has 25 thousand observations.

3.2 Examining the Date info

Let's take a look at the date info in the **training** dataset.

```
train_dt$year <- as.numeric(format(as.Date(train_dt$date_time, "%Y-%m-%d"), "%Y"))
unique(train_dt$year)
```

```
## [1] 2014 2013
```

Examine the records in the **test** data.

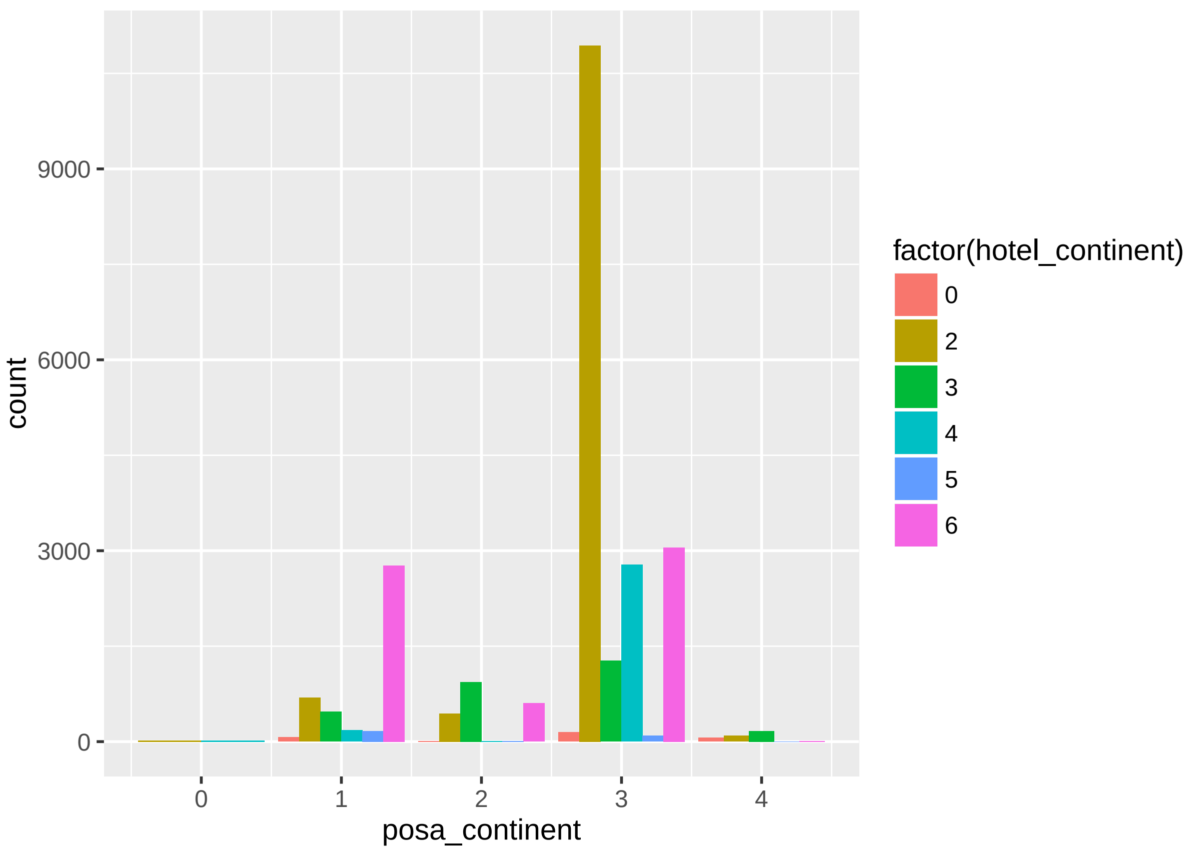
```
test_dt$year <- as.numeric(format(as.Date(test_dt$date_time, "%Y-%m-%d"), "%Y"))
unique(test_dt$year)
```

```
## [1] 2015
```

3.3 User's current location and the destination hotel location

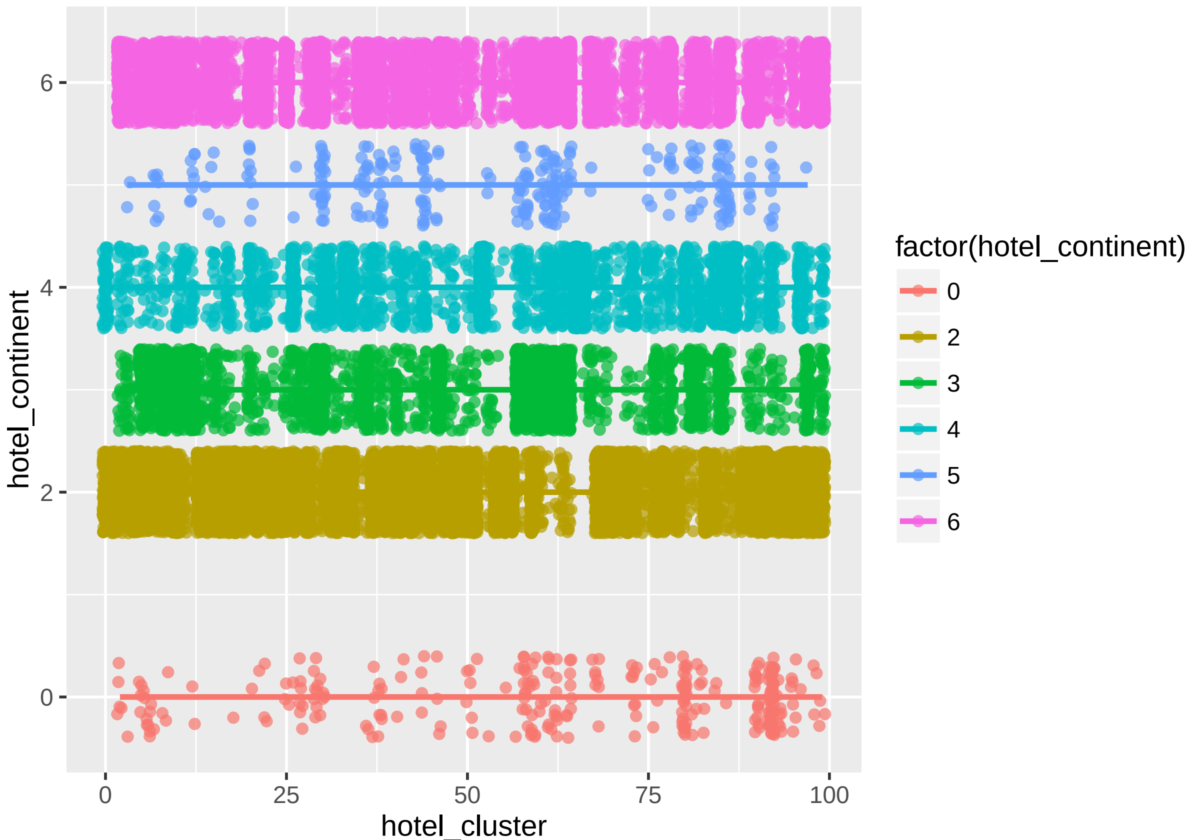
The posa_continent is user current location at the time of booking and the hotel_continent is the desitnation location.

```
ggplot(train_dt, aes(posa_continent, fill = factor(hotel_continent))) + geom_bar(position = "dodge")
```



3.4 Spread of hotel clusters in different continents.

```
ggplot(train_dt,
       aes(x = hotel_cluster, y = hotel_continent, col = factor(hotel_continent))) +       # to avoid o
  geom_jitter(alpha = 0.7) +
      geom_smooth(method = "lm", se = F)
```



4 Approach to solution

4.1 Correlation Info of hotel_cluster with rest of attributes.

```
cor(train_dt$hotel_cluster, train_dt$srch_destination_id)
```

```
## [1] 0.003811001
```

```
cor(train_dt$hotel_cluster, train_dt$posa_continent)
```

```
## [1] 0.0004647611
```

```
cor(train_dt$hotel_cluster, train_dt$user_location_country)
```

```
## [1] -0.0338143
```

```
cor(train_dt$hotel_cluster, train_dt$user_location_region)
```

```
## [1] 0.01410099
```

```r
cor(train_dt$hotel_cluster, train_dt$user_location_city)
```

```
## [1] -0.01355649
```

```r
cor(train_dt$hotel_cluster, train_dt$orig_destination_distance)
```

```
## [1] NA
```

```r
cor(train_dt$hotel_cluster, train_dt$user_id)
```

```
## [1] 0.01577343
```

```r
cor(train_dt$hotel_cluster, train_dt$is_package)
```

```
## [1] 0.05255554
```

```r
cor(train_dt$hotel_cluster, train_dt$srch_adults_cnt)
```

```
## [1] 0.0136383
```

```r
cor(train_dt$hotel_cluster, train_dt$srch_children_cnt)
```

```
## [1] 0.01279837
```

```r
cor(train_dt$hotel_cluster, train_dt$srch_rm_cnt)
```

```
## [1] -0.00102075
```

```r
cor(train_dt$hotel_cluster, train_dt$srch_destination_id)
```

```
## [1] 0.003811001
```

```r
cor(train_dt$hotel_cluster, train_dt$srch_destination_type_id)
```

```
## [1] -0.03006388
```

```r
cor(train_dt$hotel_cluster, train_dt$is_booking)
```

```
## [1] -0.02241071
```

```r
cor(train_dt$hotel_cluster, train_dt$cnt)
```

```
## [1] -0.002306554
```

```r
cor(train_dt$hotel_cluster, train_dt$hotel_continent)
```

```
## [1] 0.004342731
```

```r
cor(train_dt$hotel_cluster, train_dt$hotel_country)
```

```
## [1] -0.003691975
```

```r
cor(train_dt$hotel_cluster, train_dt$hotel_market)
```

```
## [1] 0.01885019
```

4.2 Linear Model

```r
model = lm(hotel_cluster ~ srch_destination_id + srch_destination_type_id + is_booking + cnt + orig_des
summary(model)
```

```
##
## Call:
## lm(formula = hotel_cluster ~ srch_destination_id + srch_destination_type_id +
##     is_booking + cnt + orig_destination_distance + user_location_country +
##     user_location_region + is_mobile + is_package + hotel_continent +
##     hotel_country + hotel_market, data = train_dt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.424 -24.706  -0.099  23.181  56.510
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                5.066e+01  1.067e+00  47.496  < 2e-16 ***
## srch_destination_id        3.104e-05  2.385e-05   1.302 0.193029
## srch_destination_type_id  -3.446e-01  1.201e-01  -2.870 0.004104 **
## is_booking                -3.015e+00  8.361e-01  -3.606 0.000312 ***
## cnt                       -2.526e-01  2.030e-01  -1.244 0.213385
## orig_destination_distance  9.926e-05  1.270e-04   0.782 0.434327
## user_location_country     -2.748e-02  5.336e-03  -5.151 2.63e-07 ***
## user_location_region       2.158e-03  1.983e-03   1.088 0.276570
## is_mobile                 -2.471e-01  7.117e-01  -0.347 0.728487
## is_package                 3.029e+00  5.665e-01   5.347 9.09e-08 ***
## hotel_continent            5.806e-01  1.771e-01   3.278 0.001048 **
## hotel_country             -9.066e-04  5.071e-03  -0.179 0.858109
## hotel_market              -2.199e-04  4.948e-04  -0.445 0.656679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.68 on 15298 degrees of freedom
##   (9688 observations deleted due to missingness)
## Multiple R-squared:  0.007947,	Adjusted R-squared:  0.007169
## F-statistic: 10.21 on 12 and 15298 DF,  p-value: < 2.2e-16
```

```r
#to avoid multi colinearity, try different model
model2 = lm(hotel_cluster ~ srch_destination_id + cnt + orig_destination_distance + user_location_regio
summary(model2)
```

```
##
## Call:
## lm(formula = hotel_cluster ~ srch_destination_id + cnt + orig_destination_distance +
##     user_location_region + hotel_country + hotel_market, data = train_dt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.470 -24.783   0.041  22.469  49.411
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                5.018e+01  8.613e-01  58.258  < 2e-16 ***
## srch_destination_id       -8.595e-06  2.142e-05  -0.401  0.68818
## cnt                       -4.868e-02  2.012e-01  -0.242  0.80879
## orig_destination_distance  3.087e-04  1.158e-04   2.666  0.00769 **
## user_location_region       6.423e-04  1.964e-03   0.327  0.74367
## hotel_country             -1.542e-03  4.864e-03  -0.317  0.75115
## hotel_market              -3.658e-04  4.948e-04  -0.739  0.45974
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.78 on 15304 degrees of freedom
##   (9688 observations deleted due to missingness)
## Multiple R-squared:  0.0005773,  Adjusted R-squared:  0.0001855
## F-statistic: 1.473 on 6 and 15304 DF,  p-value: 0.1828
```

4.3 Non Linear Model

```r
library(rpart)
frmla = hotel_cluster ~ srch_destination_id + user_location_region + orig_destination_distance
ctrl = rpart.control(minSplit=5, minbucket = 50)

expediaTreeModel = rpart(frmla, data = train_dt, method = "class", control=ctrl )

# get the cp - complexity factor
printcp(expediaTreeModel)
```

```
##
## Classification tree:
## rpart(formula = frmla, data = train_dt, method = "class", control = ctrl)
##
## Variables actually used in tree construction:
## character(0)
##
## Root node error: 24341/24999 = 0.97368
##
## n= 24999
```

```
##
##          CP nsplit rel error xerror xstd
## 1 0.0091204      0         1      0    0
```

4.4 Rule based approach

From the linear model results, the R-squared value is negligible. So, the dependent variables are *NOT* correlating well. Using the non linear model - CART (Classification and Regression Tree), which also illustrates that the dependent variables are not correlating well. In this case, these basic Machine Learning algorithms are not much of help for this problem.

The advanced machine learning algorithms like random forest, Xgboost etc. may be suitable for this problem. Also note that they may be better computing resources with huge RAM - 16GB to 32GB.

Alternatively, we can predict the hotel cluster by formulating a list of rules as below

4.4.1 Rule 1: Identify the often used hotel cluster for a destination

The idea is to identify the often used hotel cluster for a given destination. In our case, we have to identify the top five hotel cluster.

Also the training dataset has `is_booking` flag, ie., `is_booking` is 1 for the confirmed booking and `is_booking` is 0 for the unconfrimed booking. So, while identifying top five, we can provide more weightage to the confirmed booking.

```
# get the training data
train_dt <- fread("train.25000.csv", header = TRUE)
test_dt <- fread("test.1000.csv", header = TRUE)

## compute_weightage
## based on is_booking, give weightage to the hotel cluster
## if is_booking == 1 then weightage = 1
## else weightage = 0.15
compute_weightage <- function(booking_flag) {
  sum(booking_flag) * 0.85 + length(booking_flag) * 0.15
}

# collect the srch_destination_id based on weightage of hotel_cluster
# use data.table notation of doing J expr BY group
dest_id_n_hotel_cluster_grp_count = train_dt[, compute_weightage(is_booking), by = list(srch_destinatio

# get the top five
# inputs : hotel_cluster and weightage
get_top_five <- function(hc, n) {

  # get the ordered list interms of freq.
  # so sort them in decreasing order
  hc_ordered <- hc[order(n, decreasing = TRUE)]
  #print(hc_ordered)

  # we need 5, if no match, we may get 0.
  result <- min(5, length(hc_ordered))

  ## return the result with hc separated by spaces
```

```r
  paste(hc_ordered[1:result],  collapse = " ")
}

## again use the J expr and BY group of data table
## The package data.table creates the column with name V1
dest_id_n_top_5_hotel_cluster = dest_id_n_hotel_cluster_grp_count[ ,get_top_five(hotel_cluster, V1), by=

## Now, merge the test and top 5 clusters based on dest id.
## recommend the predicted hotel cluster
# use merge: specify both tables x & y
#          : specify the column name for merge
#          : in case of no match, add the row from x and put NA in the respective column.
recommended_hc <- merge(test_dt, dest_id_n_top_5_hotel_cluster, by = "srch_destination_id", all.x = TRUE

## extract the id and hotel clusters
result_dt <- recommended_hc[order(id), list(id, V1)]

## set the col names
setnames(result_dt, c("id", "hotel_cluster"))
```

4.4.2 Rule 2: Predict based on destination distance

Give importance for the booking when the distance between the customer and destination hotel (`orig_destination_distance`) matches.

```r
# create temporary data frame with the needed fields
t1_dt = train_dt[, list(orig_destination_distance, hotel_cluster, is_booking)]

# group them based on the is_booking flag with appropriate weights
t2_dt = t1_dt[, compute_weightage(is_booking), by = list(orig_destination_distance, hotel_cluster)]

# get the top five based on this rule
t3_dt  = t2_dt[ ,get_top_five(hotel_cluster, V1), by=orig_destination_distance]

# ignore if dest distance is NA
t4_dt = t3_dt[complete.cases(t3_dt),]

# merge test and training dataset.
merge_dt <- merge(test_dt, t4_dt, by = "orig_destination_distance", all.x = TRUE)

# extract the id and hotel clusters
result3_dt <- merge_dt[order(id), list(id, V1)]

## set the col names
setnames(result3_dt, c("id", "hotel_cluster"))
```

4.4.3 Combine the results

Combine the results and identify the unique five by providing the precedence to the above rules. Below is the snapshot of the results.

```
id,hotel_cluster
0,5 37 55 11 8
1,5
2,0 31 96 91 59
3,1 45 79 24 54
4,42 28 59 91 2
5,91 42 16 48 33
6,95 21 91 2 33
7,95 91 18 68 98
8,1 45 79 24 54
...
```

The **complete R code** is available in the following github location (www.github.com/abalaji-blr/CapstoneProject/tree/master/Source/expedia_script.R).

5 Further Exploration and Study

The advanced machine learning algorithms like random forest, XGBoost etc. need to be evaluated to see whether they help in resolving this problem.