Giovanni Briggs
4/24/15

Expedia Competition

This competition focuses on predicting whether or not a user will book a hotel based on a series of parameters. To solve this problem, I use a simple *repeatedcv* model. I did try to be very intentional about the variables that I used in the model. The main reason for that is *time*. While it would have been nice to run 10-fold repeatedcv with a very small shrinkage and use every variable at my disposal, my computer simply cannot handle it. The simplified tests that I did still took 15-30 minutes. For that reason, the Exploratory Data Analysis section in this report is lengthy.

Exploratory Data Analysis

The training data has over 50 columns. Some of these factors should be stronger indicators towards a user's behavior than others. The first problem then is determining which combination of factors will give us the best results.

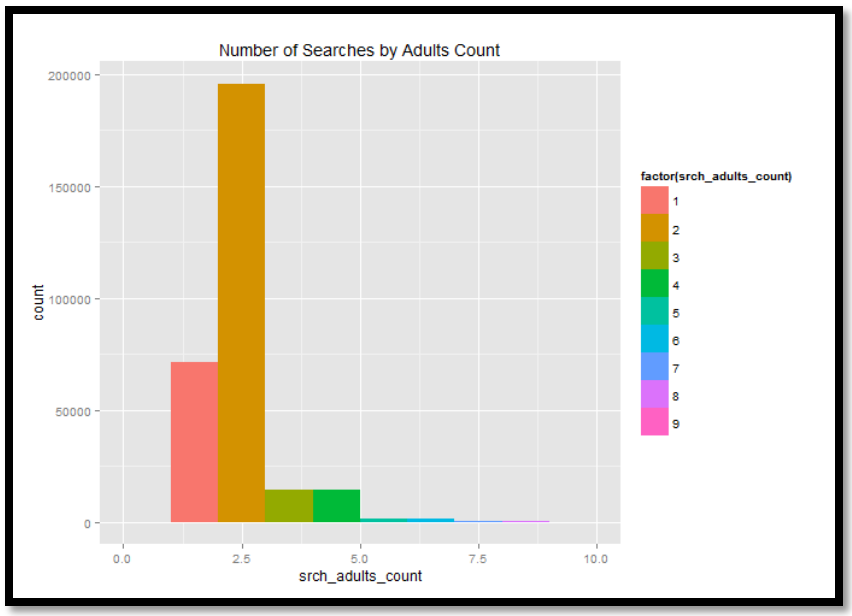
For starters, it helps to look at the general breakdown of the data. There are 300,621 searches in the training data set, but only 12,059 unique search ids. If we equate the unique search ids with users, then we can say that there are 12,059 users in the training set. There are also only 8,358 bookings in the training set, but we can also see that each user only books one hotel if they book at all. That means that 69.30923% of users book a hotel on Expedia.

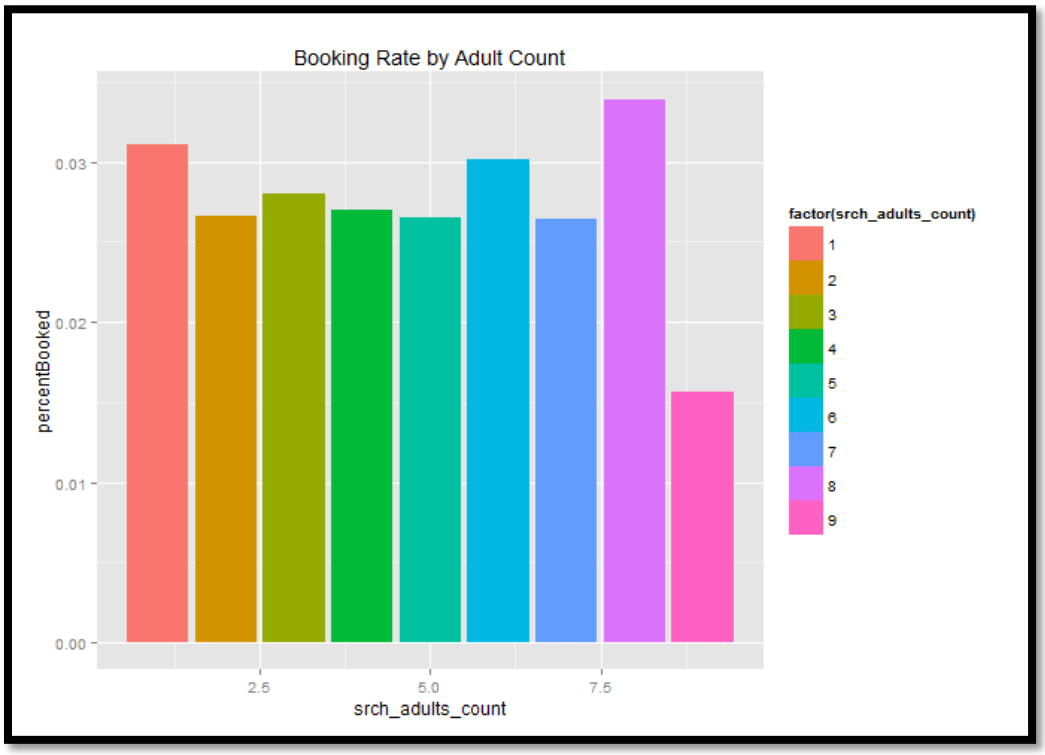
When I first saw these numbers, I wanted to do some sort of user-based collaborative filtering approach. If we can group the test and training sets into users, we can then search for similar users and then make predictions based on that similarity. I did not fully explore this idea though because the users in the test data did not have an established history that would allow me to make user based similarity comparisons.

Instead, I chose to look at each search individually. With 8,358 bookings, that means that 2.78% of searches result in a booking. The next problem is coming up with a metric to evaluate how strong of an indicator a factor is. I chose to use what I am calling *booking rate*. The *booking rate* is defined as:

$$bookingRate = \frac{\#ofBookings}{totalSearches}$$

Let's take the *srch_adult_count* column as an example. This column lists the total number of adults that the reservation is being made for. If we look at the total number of searches broken down by this count, we see that an overwhelming number of searches are for 2 adults. Since most searches involve 2 adults, it is unsurprising then that most bookings are made for 2 adults. What is more interesting is the booking rate – the total number of bookings made for 2 adults divided by the total number of searches for 2 adults.





The booking rate for that group falls shy of 3%. What we also see is how the booking rate is fairly similar across all groups. The standard error of the mean for this group is 0.00168817, which

means that the booking rate across this variable is fairly similar. This indicates that there is not very much differentiation between each of the *srch_adults_count* groups. *Table 1* shows the Standard Error of the Mean for the booking rate across multiple columns.

Column Name	Standard Error of the Mean for Booking Rate
site_id	0.001303036
prop_country_id	0.003236365
prop_review_score	0.003112549
prop_starrating	0.003475328
price_usd	0.002609757
srch_length_of_stay	0.001370446
srch_adults_count	0.00168817
srch_children_count	0.00290000
promotion_flag	0.006914509

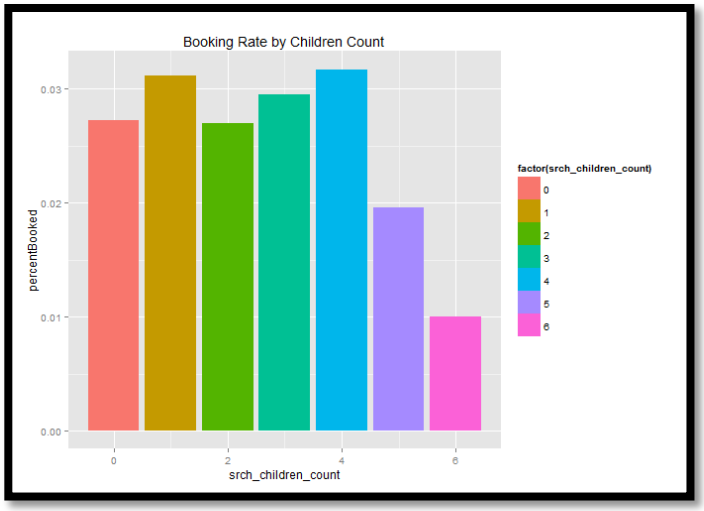
Table 1

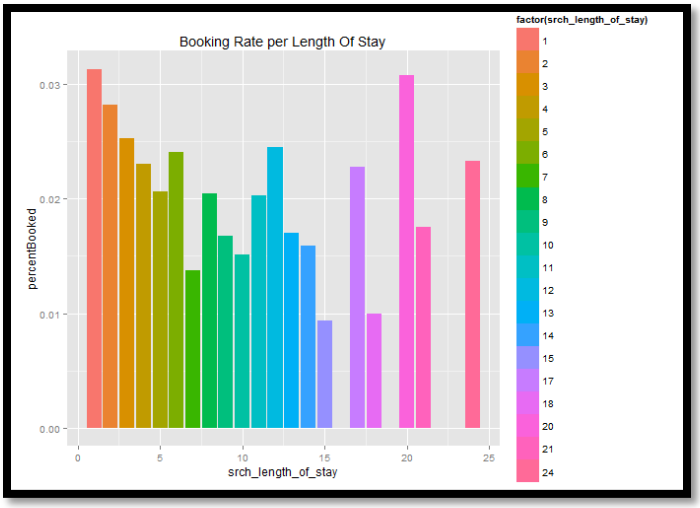
The larger the standard error of the mean, the more variation in the booking rate. This indicates that the variable is more likely to have a larger impact in a user's behavior than other variables. The *site_id* and *srch_length_of_stay* variables seem to have very low impact; whereas *prop_country_id* and *promotion_flag* appear to have a more significant impact on user behavior.

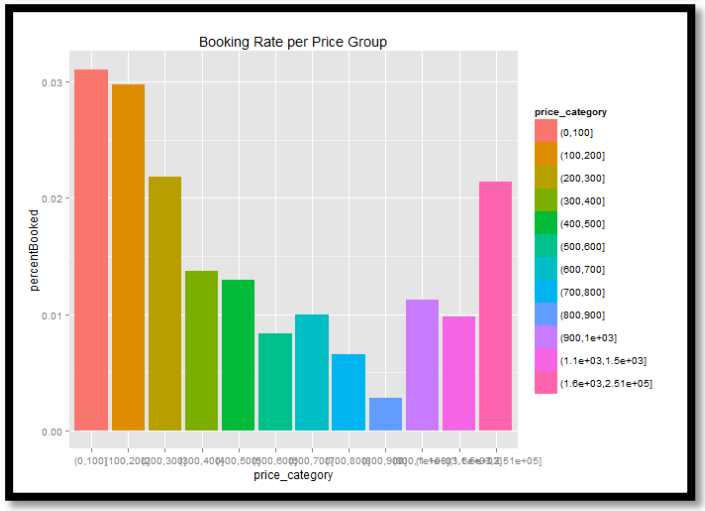
The *price_usd* column was actually divided into categories before checking the booking rate. Each category was a range of 100 and was achieved by the following line:

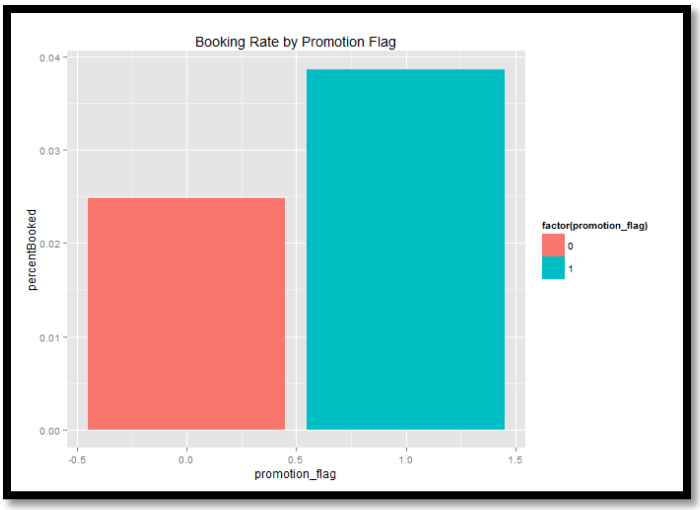
```
>>> df.train$price_category <- cut(df.train$price,  c(0,100,200,300,400,500,600,700,800,900,
        1000,1100,1500,1600,max(df.train$price)))
```

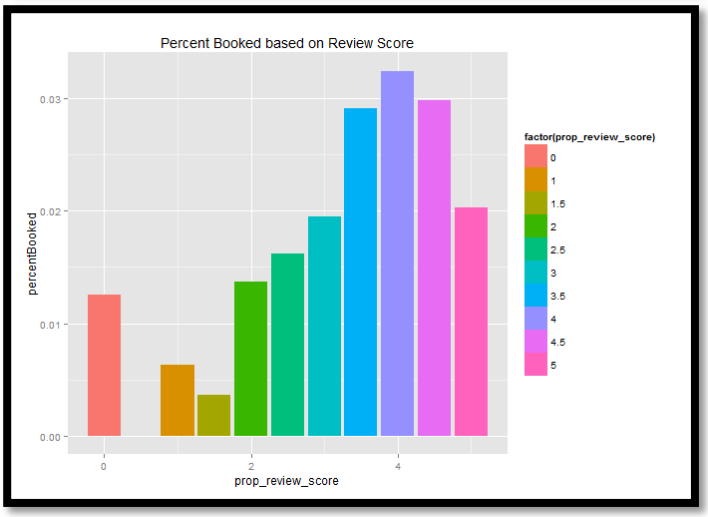
The group of graphs below shows the booking rate across the different variables.

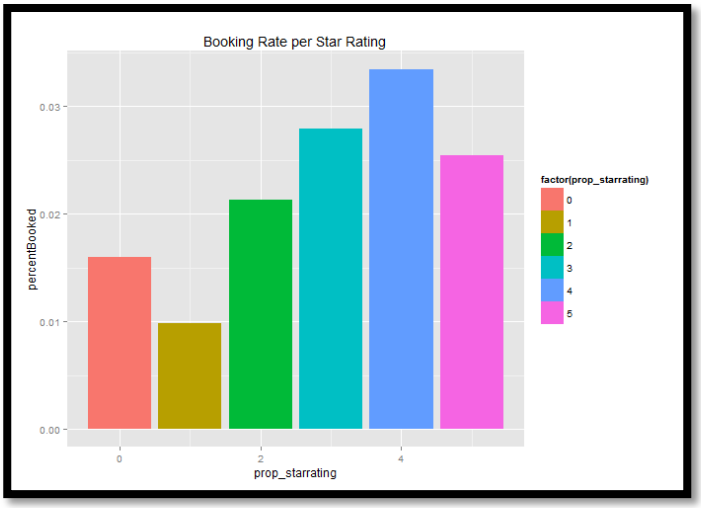

Booking Rate by Children Count
percentBooked
0.00
0.01
0.02
0.03
0
2
4
6
srch_children_count
factor(srch_children_count)
0
1
2
3
4
5
6


Booking Rate per Length Of Stay
percentBooked
0.00
0.01
0.02
0.03
0
5
10
15
20
25
srch_length_of_stay
factor(srch_length_of_stay)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
17
18
20
21
24


Booking Rate per Price Group
percentBooked
0.00
0.01
0.02
0.03
price_category
price_category
(0,100]
(100,200]
(200,300]
(300,400]
(400,500]
(500,600]
(600,700]
(700,800]
(800,900]
(900,1e+03]
(1.1e+03,1.5e+03]
(1.6e+03,2.51e+05]


Booking Rate by Promotion Flag
percentBooked
0.00
0.01
0.02
0.03
0.04
-0.5
0.0
0.5
1.0
1.5
promotion_flag
factor(promotion_flag)
0
1


Percent Booked based on Review Score
percentBooked
0.00
0.01
0.02
0.03
0
2
4
prop_review_score
factor(prop_review_score)
0
1
1.5
2
2.5
3
3.5
4
4.5
5


Booking Rate per Star Rating
percentBooked
0.00
0.01
0.02
0.03
0
2
4
prop_starrating
factor(prop_starrating)
0
1
2
3
4
5

Model

With the knowledge gained from the explorartory data analysis, I chose to define the formula for predicting the *booking_bool* as follows:

```
>>> formula <- booking_bool ~   site_id +  prop_country_id +  (prop_review_score * prop_starrating) +
price_usd +   srch_adults_count +  promotion_flag+  srch_children_count
```

An important note here is the relationship described between *prop_review_score* and *prop_starrating*. When thinking about how a user interacts with Expedia, we would expect them to take the two ratings into consideration together. This is the same behavior in movie reviews. The critics have one review and the general public another and those ratings are both published to give the user a better picture. Using * tells R to use cross factoring between the two variables. So what we actually have is:

```
>>> formula <- booking_bool ~   site_id +  prop_country_id +  (prop_review_score  :  prop_starrating) +
prop_review_score + prop_starrating +  price_usd +   srch_adults_count +  promotion_flag+
srch_children_count
```

This establishes a stronger tie between the *prop_review_score* and *prop_starrating* than simply doing a + between the variables would. Doing this showed significant improvement in the results when doing in-house testing (not uploading to Kaggle).

Another way to express this relationship would be to create a new variable called *prop_rating_average* where we take the mean of the two variables. Some of the *prop_review_score* and *prop_starrating* values are 0 though. This means that no rating information was available. Taking the average of the two variables when one of them was 0 would unfairly drag the average rating down by half. I did create functions to take those 0s and turn them into non-zero values, but the solution actually performed worse than just leaving the zeros both in-house and on Kaggle. I did not push the issue further because as we see from the booking rates of both variables, having a 0 rating actually had a higher rate than a rating of 1. Leaving the 0 seems to be more helpful than trying to impute it.

The final model used 10-fold repeatedcv with 150 trees, depths from 1 to 4, and shrinkages of 0.1, 0.05 and 0.001. The model was then fitted against the test data and the results placed in a csv file and then uploaded to Kaggle. It scored a 0.63362.