Smart Host Data Science Assignment

Michael Hirsch

May 12, 2015

1. Which listing in the comp set has the highest price? What is the correlation between price and a binary measure of occupancy?

   The highest price at a single snapshot belong to listing: 14829 with a price of: $4000.00

   The correlation between price and a binary measure of occupancy is: -0.606154688488

2. Describe an approach that would recommend a price for listing 14790 with the available data. What assumptions does this model have and how does it generate recommended price? What are the strengths and weaknesses of your proposed model?

   The approach goes as follows. We group the original dataset, excluding listing 14790 by listing_id and date, taking the mean of the prices across all of the snapshots. It appears that price is set to 0 once the listing has been booked, so days in which there was no price, since the listing was booked before the snapshotting began, I used the probable close price column. Then, we create a multi linear model, training it on a training set consisting of all listings *except* 14790. We use day of week and date as our predictor

   The model makes the following assumptions:

   - The compset is generated using location as a factor. Since we are using date as a predictor variable, we will be assuming that price fluctuates with events on particular dates.
   - Days for which price data is missing is filled by probable close price.

   The model is trained on the other listings and predicts prices based on the date and day of the week of the listing. This model is strong in that it's predictions follow the fluctuations in the training set, and it predicts higher price values during events (these are Austin listings!) and on the weekends. It's weakness is that it does not fully incorporate the booking the status of the training listings. This, however, is a limitation of the data, for we cannot be sure that a listing became unavailable because it was booked, or if it was simply taken off by the owner. If given more time, smarter analysis could refer to the ultimately booked column to decide what action had been taken.

3. Apply your model to the available data. What are the recommended prices for each night? Include any code that you use to complete this task.

   ** files attached in email **

4. Based on your analysis, do you have any suggestions about improving the data quality in the database? What data changes would you recommend that would help you implement a better pricing model for this listing? For example, can you recommend data quality improvements, or additional variables that you would like that would help inform your pricing model?

   There are a few things I would request in an updated data model:

- More thorough information on what action had been taken on the listing. This would allow me to omit entries for which the owner had made a particular night unavailable from my model.
- Price changes from snapshot to snapshot. This could be calculated, but (as I explain in the next question) I think it is more efficient to have this stored.
- Request for price to remain once listing had been booked. This is essential for the model as using probable close price values could lead to inaccurate results.
- It is a bit unclear what z-score is actually representing. Does this represent where the current price lies in relation to the average price within the compset or a running average? Also, is there another similarity measure that is defining a distribution of prices, or is it just within the compset?

5. How would you store this data set in a way that uses less storage space, without losing information? What considerations would you have for analyzing with data in this format? If money was no object and we were tracking 3 million listings, what would be the optimal system to use?

   I would recommend the following changes to data storage:

   - Store only snapshots in which the data has changed. Most of the prices actually aren't changing that much, and we have a lot of redundant information here. Also, adding the percent change in the price when the price value is changed will accomplish what I believe the redundancy is trying to do.
   - I'm not entirely clear on how this csv was presented, but we could store listings with their compsets as a separate objects, and individual listings' snapshot information as their own objects.

   If data were stored this way, we would need to compute the intermediary snapshots when doing out analysis. Since we would not be storing redundant information, we would need to create entries for the days in which the price remained fixed. We could simply take the range of dates between the two snapshots and set the price equal to the price before the change. Also, when generating datasets like the one included in the assignment, we would first need to refer to the compset objects, then get the appropriate snapshots from their locations.

   For storing a large amount of listings, we could go with redis, a very flexible and popular NoSql database. There are many benefits to going with redis, most notably it's speed and versatility in storage. This could allow us to give near real time pricing information when adding or removing items from a compset.