Problem Set 3

Joel Whittier

10 March 2019

Part I
OLS and Random Forest go toe-to-toe

1 OLS

For these types of predictive projects, I try to start by asking myself for the most powerful features I could possibly hope for, then try my best to replicate them. If I could magically quantify anything, I would quantify the perceived value of each property prior to the stay, and the perceived quality post stay. I would then take the difference between the two and use that to determine the review.

This first led me towards things I should dis-include: public information. People will generally give themselves reasonable expectations based on the listing. If I go into a listing knowing that I'm sleeping on an air mattress, I'm not going to give it a negative review for sleeping on an air mattress. If it was that much of a problem for me I would have never booked the room in the first place. What I am really looking for is things that could happen on the stay that would make it good or bad, which I probably wouldn't realize going in.

Don't get me wrong, I factor in prices in all my models. But it's partially because prices can reveal traits of an owner. For example, if an owner charges high hidden fees (I would guess a lot of people don't look at the cleaning fee), that could leave me with a sour taste. There's also always going to be biases against low cost properties, because not everyone is a rational economist who recognizes that you are only paying for a bed and nothing else at a 50 dollar property.

Most of the categorical and outside variables attempt to create a caricature of unexpected aspects of a stay, along with the owners.

Characterizing the owners was the easier part. For example, a host with a better response rate probably takes the job much more seriously than one with a lower. Another example for the owners is the race of the owner, as it is well

documented (mainly through studies on tipping) that whites tend to always receive better reviews.

Unexpected aspects was much harder, but definitely the biggest difference maker on this assignment. One of my most powerful variables was detecting overcrowded set ups: ones where there are way too many guests allowed for how many rooms the place actually had. This would certainly lead to parties bringing more guests than were comfortable and ruining their stay. Another was transit descriptions. It's notoriously hard for tourists to navigate any city they visit, and while most think they can just wing it, a little help can certainly avoid disasters that would be reflected in the reviews. And finally, some locations are prone to tourists having much less fun than they expect. So many visit places like New York and don't realize what a city of that size actually feels like; it can be overwhelming, and their human side will absolutely influence the reviews.

And finally, it was probably most important to figure out what to disclude. Anything with a few observations would absolutely be over-fitting, so things like square footage were rather worthless to this exercise. Even if it's not assessed on this basis, I always avoided variables with signs clearly in the wrong direction, even if it immensely "improved" the regression. Overall, this assignment is a serious test of self control. It's irresponsible to just include anything that makes the R-squared higher, because at some point it would certainly become overfitting.

R-2 Prediction

I predict my R-2 will be about .107; with my 95% confidence interval being (0.099,0.115).

In order to find my expected R-Squared, I used k-folding to generate 5 predicted R-Squared, then took the average of them. This basically means I used a random number generator to create 5 subsamples in my data. Each R-squared of this would be called a predicted R-squared. From there, I averaged the predicted R-squared to get about .107. This happened to be almost exactly the same as my R-squared of the original regression, which is typically a good sign of fitting correctly.

I was initially concerned about the range of my predicted R-squared, as it spanned from .094 to .114. However, I realized this sort of range is pretty standard to any model ran on this data set; from the most under-fit to the most correctly fit, each had a .25 range to mean value ratio. So I chose to accept these result, along with the fact that my coefficients all had high t values.

I will be the first to admit this problem set was incredibly difficult and I could absolutely be overestimating. However, I truly believe my verification process was the best I could build given my model, and that the results produced proved my answer reasonable.

2 Random Forest

This models selection process was generally similar to OLS. The biggest difference between the two was a much smaller concern for over-fitting, as over-fitting

is much harder to pull off in a random forest. Obviously, I avoided doing things such as coding in latitude and longitudinal locations, as that could act as a hard classification system. But because of validation methods, as long as I truly partition a cross validation sample, it is incredibly difficult to present an over-fit model. I also removed some of the transformations I made in part 1 that would only act as noise in a Random Forest. Because it has the ability to cluster, it is not nearly as sensitive to extreme values, which takes away a major reason to apply log transformations on features. There were also a few features that were strong in the random forest but not the OLS model, such as the ratio of beds per room. I'm assuming most of these features are used internally to classify over regress, which is why it does not help OLS, but is usable in the random forest.

R-2 Prediction

I believe my R-2 will be 0.149.

I determined this by creating a validation with 10% of the data. This helps prevent over-fitting, as a validation sample is a proxy of the true out of sample. When regressing the scores against the predicted scores, the R-2 was .140. This is also about in line with the theoretical relationship between OLS and random forest for regression. This problem require a lot of psuedo-classification: it's really easy to tell when a property is really good or really bad, but near impossible to model exactly how due to the confounding factors. By simply being able to identify really good properties, and not having to worry about distribution, we can construct a much higher R-squared.

Again, true out-of-sample on a problem like this one will usually be a little lower than the cross validation. However, we also have to account for the fact that we include the cross validation data in the final production, which should slightly boost this number. For this reason, I think the .149 I received for my cross-validation set is the most responsible estimate I could make.

Part II

What Happens When Your Data Is Missing?

My first thought was to take the most expensive houses from the dataset and use that as the represenative sample. However, I decided to change this slightly for a couple of reasons.

First of all, price is incredibly dependent on location. When we are looking at the absolute most expensive listings on airbnb, we are looking at ones that certainly meet a standard of luxury in some of the most expensive cities in the world. If we just took the most expensive overall, we would get some mid tier apartments in New York in our data set, and lose out on some luxurious points in smaller cities, which are probably also very representative of this set. For

this reason, I wanted a set that contained mostly luxurious big city properties, but also midsize city luxury, as some mansions in smaller areas certainly appear in the most expensive list.

So, I had to group by geographic region, and take the top 10% in each region. Because big cities had more properties, it would capture more of them, but also represent the small town beauties. And luckily for me, our good friends at the post office have already sorted these regions for me! The first digit in a zip-code actually represents the state belonging to one of 10 geographical regions. I'm not exactly sure how they came up with this grouping, but I'll trust the post office's system more than what I would come up with. So, I took the top 10% in each region, and began using that to train my model.

My model between this and part 1 were mostly similar. The biggest difference was removing some categorical variables, such as cancellation policy and old owners. I figured these were not nearly as relevant because cancellation must inherently be strict on these, and the owners must almost always be old. I also removed a variable about listed availability, because many of those properties reach a point where exclusivity is a bonus, but many do not, so it is impossible to accurately include this variable.

2.1 Random Forest

I used the same training data set as OLS.

Again, excessive variables are not nearly as dangerous in random forest. I took out some variables mentioned above that would clearly overfit the data. The biggest change I made was reducing the maximum depth of this tree. Because our out of sample data is not represented in our random forest, we want to force this model to be more generalized. If it is overly specific, as it expects something in sample, it will become an even worse predictor of the data.

Better Predictor

The better predictor will almost certainly be OLS. Random Forest is incredibly reliant on a data set being in sample, as is the case with all machine learning. There is no good way to "translate" models between two data sets like OLS, each sample you have to start brand new.

Why? This is harder with random forest because random forest inherently classifies. Even when it regresses, it is looking for specific combinations and patterns within the data set to build its model. There is no particular way to predict if these patterns will appear in a different data set of how they will register in other sets. Because forests are so specific to one set of patterns, it is nearly impossible to transfer it to a different one.

A good example of this is the classic dog or cat classification. If you take a machine learning model (like random forest) and feed it pictures of dogs and cats, it becomes incredibly good at recognizing which is which. The problem is, if you feed it squirrels out of sample, it will have no tools to recognize that it does not fit well with the patterns of the original sample. It is entirely possible the out of sample data set will be filled with metaphorical squirrels and create some insane residuals, which the OLS will be much more resiliant towards.