Helen Chen
March 9, 2019
Professor Levitt
Data Construction

Problem Set 3

Data Cleaning

Before being able to fit models for OLS or Random Forest, it is important to ensure that the data being worked with is properly preprocessed, such that it makes sense in the context of the problem that we are looking to solve. I first did this through the following steps:

1. Determining which variables to keep

The original dataset given with the pset contains 88 columns of data. After looking through the column names and glancing at data values, I cut down the number of columns to 57 by dropping columns such as listing_url which are unique to each listing without adding any new or valuable information as well as columns like country which are the same for every listing in the data.

> id summary host_id host_name host_since host_location host_response_time host_response_rate host_acceptance_rate sex race age host_neighbourhood host_listings_count square_feet host_total_listings_count host_verifications host_has_profile_pic neighbourhood city state is_location_exact property_type room_type accommodates bathrooms bedrooms beds bed_type amenities price weekly_price monthly_price security_deposit cleaning_fee guests_included extra_people minimum_nights maximum_nights calendar_updated number_of_reviews first_review last_review review_scores_rating cancellation_policy house_rules reviews_per_month host_is_superhost host_identity_verified require_guest_profile_picture require_guest_phone_verification instant_bookable group_cancellation_policy first_review_day first_review_month first_review_year indicator

2. Cleaning Data

Among other things, I cleaned summary, house_rules, and amenities by counting number of words in each; I created indicator variables for categorical variables like sex, age, and race; I fixed capitalization errors for states and destrung several other variables.

3. Generating dummy variables for missing categorical data

For each of the categorical variables with missing data that I cleaned, I also created dummy variables for each to include in regression.

Cleaning Data Code

```
replace summary = "" if summary == "N/A"
gen summary_words = wordcount(summary)

replace host_response_time = "" if host_response_time == "N/A"
replace host_response_time = "NA" if host_response_time == ""
tab host_response_time, generate(host_response_time)

gen miss_host_response_time = 0
replace miss_host_response_time = 1 if host_response_time=="NA"

replace sex = "" if sex == "Unknown"
replace sex = "NA" if sex == ""
tab sex, gen(sex)
gen miss_sex = 0
replace miss_sex = 1 if sex=="NA"

replace age = "" if age == "Unknown"
replace age = "NA" if age == ""
tab age, gen(age)
gen miss_age = 0
replace miss_age = 1 if age=="NA"

replace race = "" if race == "Unknown"
replace race = "NA" if race == ""
tab race, gen(race)
gen miss_race = 0
replace miss_race = 1 if race=="NA"

replace host_total_listings_count = "" if host_total_listings_count == "NA"
destring host_total_listings_count, replace force
egen mean_host_total_listings = mean(host_total_listings_count)
replace host_total_listings_count = mean_host_total_listings if missing(host_total_listings)
drop mean_host_total_listings

tab cancellation_policy, gen(cancellation_policy)

replace house_rules = "" if house_rules == "N/A"
replace house_rules = "" if house_rules == "NA"
gen house_rules_words = wordcount(house_rules)

generate cat_host_has_profile_pic = 0
replace cat_host_has_profile_pic = 1 if host_has_profile_pic == "t"
label define cat_host_has_profile_pic_label 0 "f" 1 "t"
label values cat_host_has_profile_pic cat_host_has_profile_pic_label
drop host_has_profile_pic
rename cat_host_has_profile_pic host_has_profile_pic

replace state = proper(state)
replace state = "Ny" if state == "New York"
tab state, generate(state)

generate cat_is_location_exact = 0
replace cat_is_location_exact = 1 if is_location_exact == "t"
label define cat_is_location_exact_label 0 "f" 1 "t"
label values cat_is_location_exact cat_is_location_exact_label
drop is_location_exact
rename cat_is_location_exact is_location_exact

tab property_type, gen(property_type)


tab room_type, gen(room_type)
```

```
replace bedrooms = . if bedrooms < 0
egen mean_bedrooms = mean(bedrooms)
replace bedrooms = mean_bedrooms if missing(bedrooms)
drop mean_bedrooms

tab bed_type, gen(bed_type)

replace square_feet = "" if square_feet == "NA"
destring square_feet, replace force
egen mean_sq_ft = mean(square_feet)
replace square_feet = mean_sq_ft if missing(square_feet)
drop mean_sq_ft

egen mean_weekly_price = mean(weekly_price)
replace weekly_price = mean_weekly_price if missing(weekly_price)
drop mean_weekly_price

egen mean_monthly_price = mean(monthly_price)
replace monthly_price = mean_monthly_price if missing(monthly_price)
drop mean_monthly_price

egen mean_security_deposit = mean(security_deposit)
replace security_deposit = mean_security_deposit if missing(security_deposit)
drop mean_security_deposit

replace cleaning_fee = . if cleaning_fee < 0
egen mean_cleaning_fee = mean(cleaning_fee)
replace cleaning_fee = mean_cleaning_fee if missing(cleaning_fee)
drop mean_cleaning_fee

gen host_verifications_words = wordcount(host_verifications)
replace host_verifications_words = 0 if host_verifications == "[]"
drop host_verifications
rename host_verifications_words host_verifications

gen no_amenities = length(amenities) - length(subinstr(amenities, ",", "", .))

egen mean_first_review_year  = mean(first_review_year)
replace first_review_year = mean_first_review_year if missing(first_review_year)
drop mean_first_review_year
```

1.1 OLS Regression

1. After cleaning all of my data, I first ran a cross validated regression including all 107 predictors many of which were indicator variables for things like property_type and other categorical variables. I used crossfold in stata with 8 folds.

```
crossfold regress review_scores_rating host_response_rate host_acceptance_rate host_listings_count
host_total_listings_count accommodates bathrooms bedrooms beds price weekly_price monthly_price
security_deposit cleaning_fee guests_included extra_people number_of_reviews reviews_per_month
host_is_superhost host_identity_verified require_guest_profile_picture require_guest_phone_verification
instant_bookable group_cancellation_policy indicator summary_words host_response_time1
```

host_response_time2 host_response_time3 host_response_time4 miss_host_response_time sex1 sex2 sex3 sex4 sex5 miss_sex age1 age2 age3 age4 miss_age race1 race2 race3 race4 race5 miss_race cancellation_policy1 cancellation_policy2 cancellation_policy3 cancellation_policy4 cancellation_policy5 cancellation_policy6 cancellation_policy7 house_rules_words host_has_profile_pic state1 state2 state3 state4 state5 state6 state7 state8 is_location_exact property_type1 property_type2 property_type3 property_type4 property_type5 property_type6 property_type7 property_type8 property_type9 property_type10 property_type11 property_type12 property_type13 property_type14 property_type15 property_type16 property_type17 property_type18 property_type19 property_type20 property_type21 property_type22 property_type23 property_type24 property_type25 property_type26 property_type27 property_type28 property_type29 property_type30 property_type31 room_type1 room_type2 room_type3 bed_type1 bed_type2 bed_type3 bed_type4 bed_type5 host_verifications no_amenities first_review_year if indicator==1 & review_scores_rating != ., k(8) loud r2

Doing this gave me 8 R^2 values which averaged to: **0.10384**

(.1008378 + .1090492 + .1041337 + .1158984 + .0904227 + .116003 +.1036043 + .0907772) / 8 = **0.1038407875.**

However, since not all of the predictors were significant, I then went back to the regression and looked the p-values of the possible predictors for each of the partitions. From there, I looked for predictors that were statistically across the most partitions, using my judgement to remove predictors that did not make sense logically and add others back when I believed they were important to the model. I used crossfold in stata with 8 folds to see how the predictors I was playing with affected R^2. I found that host_is_superhost, price, accommodates, and strict cancellation_policy were four of the most significant predictors, which makes sense given that these are all important factors that affect which airbnbs people book, and thus their overall experience. After keeping just those that were the most significant, I ended up with a regression with 42 predictors as my final model.

regress review_scores_rating accommodates age1 age3 bed_type4 bed_type5 cancellation_policy5 cleaning_fee first_review_year host_is_superhost host_listings_count host_response_rate host_response_time1 host_response_time2 host_response_time3 host_verifications house_rules_words instant_bookable no_amenities number_of_reviews price property_type1 property_type2 property_type11 property_type16 property_type20 property_type21 race1 race2 race3 race5 reviews_per_month require_guest_phone_verification room_type3 sex2 sex3 state1 state3 state4 state6 state7 state8 summary_words if indicator==1 & review_scores_rating != .

2. The R^2 value I guess my performance will be is **0.10354**.

This comes from cross validated data using crossfold in stata with 8 folds.

```
crossfold regress review_scores_rating accommodates age1 age3 bed_type4 bed_type5
cancellation_policy5 cleaning_fee first_review_year host_is_superhost host_listings_count
host_response_rate host_response_time1  host_response_time2  host_response_time3 host_verifications
house_rules_words instant_bookable no_amenities number_of_reviews price property_type1 property_type2
property_type11 property_type16 property_type20 property_type21 race1 race2 race3 race5
reviews_per_month require_guest_phone_verification room_type3 sex2 sex3 state1 state3 state4 state6
state7 state8 summary_words if indicator==1 &  review_scores_rating != ., k(8) loud r2
```

Running the above code gave me 8 R^2 values which averaged to: **0.10354** which is a good estimate of how well my model will do out of sample.

(.1088822 + .1051417 + .1025733 + .1024969 + .1108554 +.1130715 + .0988244 + .0865081) / 8 = **0.1035441875**

1.2 Random Forest

1. For random forest, I included most of the variables that I cleaned in my original regression; manually excluding some of the ones I felt like did not make sense after looking through the data in previous parts of the problem. In order to cross validate for random forest, I partitioned the training data into 4 partitions. For each partition, I excluded it from the random forest model, and then used the random forest model generated to predict values for the partition. I then regressed these predicted values with the true values for the training set to predict a reasonable R^2 for out of sample predictions. I also tried several different iterations, depths, and lsizes to come up with the model:

```
randomforest review_scores_rating host_response_rate host_acceptance_rate host_listings_count
accommodates bathrooms bedrooms beds price weekly_price monthly_price security_deposit cleaning_fee
guests_included extra_people number_of_reviews reviews_per_month host_is_superhost
host_identity_verified require_guest_profile_picture require_guest_phone_verification instant_bookable
group_cancellation_policy summary_words host_response_time1 host_response_time2
host_response_time3 host_response_time4 miss_host_response_time sex1 sex2 sex3 sex4 sex5 miss_sex
age1 age2 age3 age4 miss_age race1 race2 race3 race4 race5 miss_race cancellation_policy1
cancellation_policy2 cancellation_policy3 cancellation_policy4 cancellation_policy5 cancellation_policy6
```

cancellation_policy7 house_rules_words host_has_profile_pic state1 state2 state3 state4 state5 state6 state7 state8 is_location_exact property_type1 property_type2 property_type11 property_type16 property_type20 property_type21 property_type27 room_type1 room_type2 room_type3 bed_type1 bed_type2 bed_type3 bed_type4 bed_type5 host_verifications no_amenities first_review_year if indicator==1 & partitions != 1 & review_scores_rating != ., type(reg) iter(11) depth(11) lsize(20)

2. The R^2 value I guess my performance will be is **0.1190**.

This value was found through regressing values predicted for each partition in the training data on their actual values through the method mentioned in the previous part of this problem.

```
. regress review_scores_rating pred if review_scores_rating !=.
```

Source	SS	df	MS	Number of obs	=	39,719
				F(1, 39717)	=	5365.01
Model	319675.426	1	319675.426	Prob > F	=	0.0000
Residual	2366546.28	39,717	59.5852225	R-squared	=	0.1190
				Adj R-squared	=	0.1190
Total	2686221.71	39,718	67.6323508	Root MSE	=	7.7191

review_sco~g	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
pred	1.182267	.016141	73.25	0.000	1.15063	1.213903
_cons	-17.01575	1.507513	-11.29	0.000	-19.97051	-14.06098

3. Since the R^2 value for random forest is higher that the R^2 obtained from OLS, I would predict that random forest would do better in out of sample predictions.

2.1 OLS

1. The thought process in what makes a good OLS model is very similar to question 1, thus I used my model from the previous regression as a starting point since I already know that this model does fairly well out of sample. I dropped bed_type5 and room_type3 which correspond to pull out bed and shared room, since these traits are less applicable to higher priced airbnbs. Another reason why I dropped some variables is because I did not want to overfit the model with data

from lower prices. Because we also know that the out of sample predictions we are trying to make have higher prices than any of the ones in our training set, I also limited the regression of the training data to values with prices higher than the median price. Since the prices of these airbnbs are closer to the out of sample data we are trying to it, it may do a better job with prediction.

My resulting model is as follows:

```
regress review_scores_rating accommodates age1 age3 bed_type5 cancellation_policy5 cleaning_fee
first_review_year host_is_superhost host_listings_count host_response_rate host_response_time1
host_response_time2  host_response_time3 host_verifications house_rules_words instant_bookable
no_amenities number_of_reviews price property_type1 property_type2 property_type11 property_type16
property_type20 property_type21 race1 race2 race3 race5 reviews_per_month
require_guest_phone_verification room_type1 sex2 sex3 state1 state3 state4 state6 state7 state8
summary_words if indicator==1 &  review_scores_rating != .  & price > med_price
```

2.2 Random Forest

1. One of the possible falliabilities of random forest is that it can be overfit to the training data. Thus in order to minimize this effect, I decreased the depth of the model, such that we are less likely to overfit to lower values than what we are trying to predict out of sample. I also removed bed_type and room_type because they seem less relevant to higher prices, also as a precaution to not overfitting.

```
randomforest review_scores_rating host_response_rate host_acceptance_rate host_listings_count
accommodates bathrooms bedrooms beds price weekly_price monthly_price security_deposit cleaning_fee
guests_included extra_people number_of_reviews reviews_per_month host_is_superhost
host_identity_verified require_guest_profile_picture require_guest_phone_verification instant_bookable
group_cancellation_policy summary_words host_response_time1 host_response_time2
host_response_time3 host_response_time4 miss_host_response_time sex1 sex2 sex3 sex4 sex5 miss_sex
age1 age2 age3 age4 miss_age race1 race2 race3 race4 race5 miss_race cancellation_policy1
cancellation_policy2 cancellation_policy3 cancellation_policy4 cancellation_policy5 cancellation_policy6
cancellation_policy7 house_rules_words host_has_profile_pic state1 state2 state3 state4 state5 state6 state7
state8 is_location_exact property_type1 property_type2 property_type11 property_type16 property_type20
property_type21  property_type27 room_type1 host_verifications no_amenities first_review_year if
```

```
indicator==1 & partitions != 1 & price > med_price & review_scores_rating != ., type(reg) iter(10) depth(7)
lsize(20)
```

2. I think that OLS will do better than RF on this question for out of sample predictions, due to the way the models are structured. OLS minimizes residuals, and since we know that ratings are clustered in the high 80s and 90s and is bounded by 100, a lot of the ratings for higher priced airbnbs will be similar to rating in the current data.

3. This question is harder to answer using random forest because of differences between random forest and OLS. In this case in which we must extrapolate ratings for higher prices, none of the training data that we feed to the random forest contains high prices, so it becomes difficult to train a random forest model to look at interactions between other variables and high prices.