Airbnb Price Prediction Model Report

Phoebe (Chi-Hsin) Chen

November 30, 2020

Abstract
In this report, I describe my insights and process of arriving at the best machine learning model on predicting prices of properties listed on Airbnb. After constructing more than 20 models, I found that an XGBoost model with 100 nrounds and 3 nfolds yielded the lowest RMSE of 56.13970.

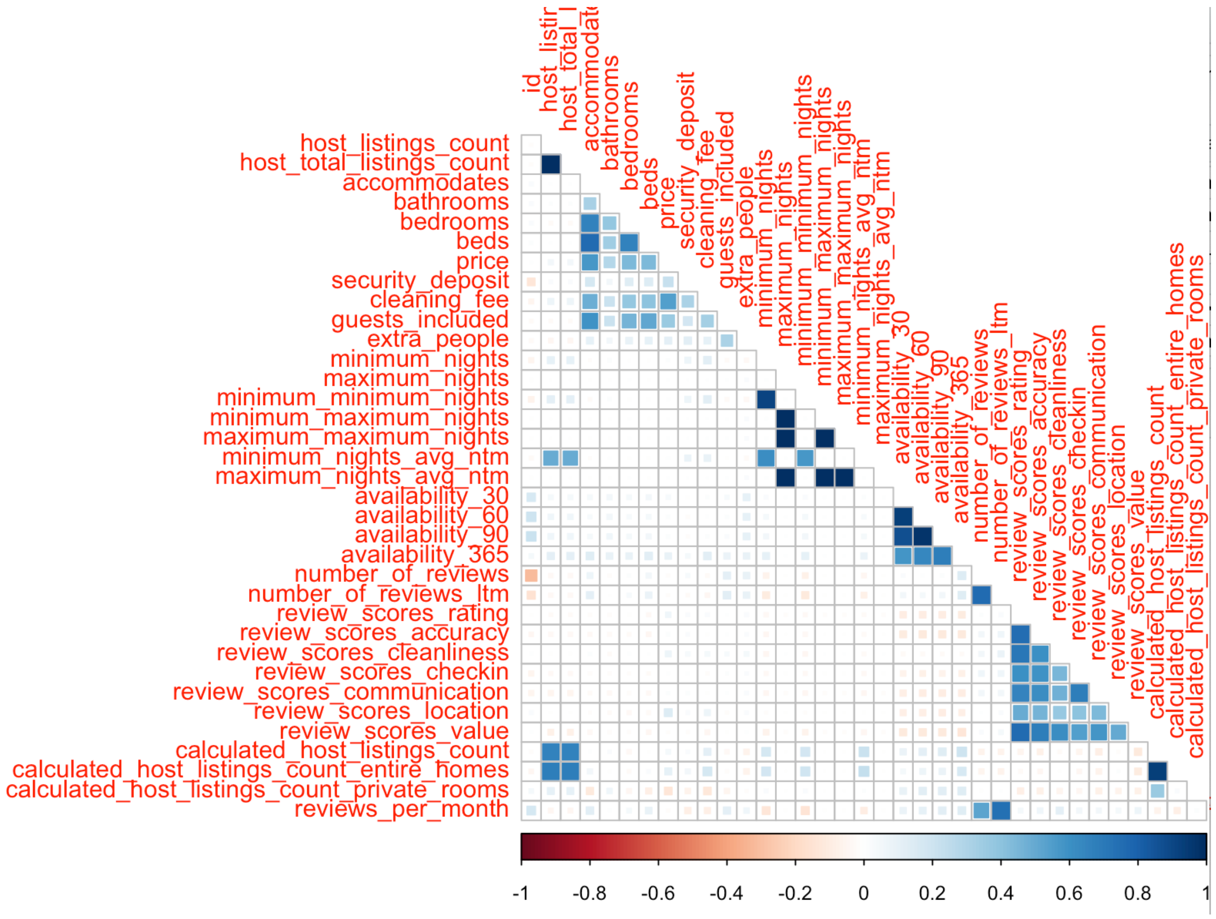
1 Initial Exploration

After receiving the analysisData and scoringData, it was apparent that these datasets were much larger and much more incomplete than the "perfect" datasets we previously used in class. But not only was there a lot of missing data, there was also a lot of obviously irrelevant data. For example, since the objective of the project was to predict the price of these properties, it was intuitively clear that certain information such as the host's name would be irrelevant. Furthermore, at a quick glance I could tell that some information such as country, country_code, and state would be irrelevant as they were identical for all properties in the analysisData.

Thus, it was obvious that a lot of data cleaning was needed before creating the actual prediction model. However, this was one of the most challenging parts of the project for me, as given the "perfect" datasets we used in class, we did not do a lot of data cleaning.

2 Methods and Feature Selection

In my final XGBoost model, I used five packages. I first used the dplyr package to combine the analysisData and the scoringData. This was done for efficiency so that the two datasets could be cleaned together, and to ensure that the techniques used to clean the two datasets were the same. Next, I used the caret package, specifically the preProcess and predict functions to replace all missing numerical values with the median value. This would ensure that there are no numerical NA values in the training and test sets to obtain a more accurate prediction model. I also used the corrplot package to identify any highly correlated predictors that needed to be removed (see correlation plot below). For example, minimum_minimum_nights was highly correlated with minimum_nights, availability_30 was highly correlated with availability_60 etc. During this step I removed 8 columns including the id column (which obviously would not have been a good predictor of price). Lastly, I used the vtreat and xgboost packages to create my XGBoost model.



Other than that, for data cleaning, I cleaned up the zipcode variable to contain only 5 digits, obtain the 40 most popular zipcodes, and convert it from numerical to factor type. I also removed all numerical columns with variances close to zero. As for character columns, I converted all of them into factor type and then removed columns with more than 50 levels and those with more than 90% NAs. This was done because with so many levels/NA values, I thought these variables would not be good predictors.

3 Model Comparison

I tried many different types of models ranging from linear to my final model of XGBoost and also many different feature selections. Even after discovering that XGBoost was the best performing model, I tested with various nrounds and nfold predictors. Below is a table that summarizes some of my better-performing models. The model highlighted in yellow shows the best performing model.

Model Type	RMSE on Training Data	RMSE on Kaggle	Notes
Model 4: Regression Tree	61.98714	71.38797	only uses numerical predictors cp = 0.001
Model 4: Regression Tree	69.91185	72.14861	only uses numerical predictors cp = 0.0005
Model 9: cvBoosting	69.90322	67.43226	n.trees = 50

Model 10: XGBoost	48.08502	56.68966	nrounds = 50
			nfold = 3

Model 10: XGBoost	46.03396	56.13970	nrounds = 100
			nfold = 3

Model 10: XGBoost	43.22265	56.22171	nrounds = 100
			nfold = 10

Model 10: XGBoost	45.50103	57.63900	nrounds = 200
			nfold = 3

Model 10: XGBoost	43.15294	56.60607	with best subset
			nrounds = 100
			nfold = 3

4 Discussion

From the importance matrix shown below, it is clear that the level “Entire home/apt” of room type, bathrooms, and cleaning_fee had the most impact on the accuracy of the model as these predictors had the highest gain. On the contrary, cancellation_policy, “Hotel room” room type, “super_strict_30” cancellation policy, and “Other” property type added the least gain to the model.

```
                                       Feature         Gain        Cover    Frequency   Importance
  1:        room_type_lev_x_Entire_home_slash_apt 2.609974e-01 1.219459e-02 0.0057125676 2.609974e-01
  2:                                    bathrooms 1.520524e-01 1.275507e-02 0.0366806975 1.520524e-01
  3:                                 cleaning_fee 1.061161e-01 6.279635e-02 0.0817799158 1.061161e-01
  4: neighbourhood_group_cleansed_lev_x_Manhattan 7.767513e-02 1.563599e-02 0.0099218280 7.767513e-02
  5:                                     bedrooms 6.899654e-02 3.601306e-02 0.0432952495 6.899654e-02
 ---
104:                property_type_lev_x_Guest_suite 1.809028e-05 2.329551e-03 0.0003006615 1.809028e-05
105:                property_type_lev_x_Earth_house 1.807287e-05 2.164284e-05 0.0003006615 1.807287e-05
106:     cancellation_policy_lev_x_super_strict_30 1.080528e-05 2.404649e-03 0.0003006615 1.080528e-05
107:                       room_type_lev_x_Hotel_room 9.771686e-06 2.451222e-03 0.0003006615 9.771686e-06
108:                      property_type_lev_x_Other 9.657368e-06 2.451098e-03 0.0003006615 9.657368e-06
```

Given the fact that many variables in the dataset given were character-type data that I converted into factor type, it made sense that the best-performing model would be able to accurately utilize these predictors. However, since XGBoost converts these variables into binary dummy variables, it was interesting to note that factor variables with more than 2 levels could be accurately utilized as a binary variable as well.

I also created another XGBoost model that included regsubsets function to find the best subset. However, I was unable to include non-numerical predictors in the regsubsets function and so this model yielded a RMSE higher than my model without this function.

5 Future Directions

If I were given more time, I would attempt to clean up the variables I had removed from the dataset. For example, I removed variables first_review and last_review because there were more than 50 levels. However, I do believe that if I were able to mutate these variables into categories such as “within last month”, “within last 3 months”, “within last year” etc. these variables could be a good predictor of price as well as people may be more willing to pay more for a property more people have stayed at and reviewed.

6 Appendix

At the beginning of this project, because I was initially unsure how to clean character type data, my first few models used only numerical predictors. I constructed many different models using only numerical predictors, ranging from linear models to lasso models. This yielded RMSE in the 70s on Kaggle.

Next, as I understood how to clean and use non-numerical predictors, I began to include them in many of models such as Random Forest, cvRanger, cvBoost etc. However, I realized that after removing bad-performing predictors, a lot of the columns I was cleaning did not end up as a predictor in my model. For example, in my cvBoosting model, I removed outliers of sqft_99 only to find that this predictor was later removed. After trying all these models using essentially the same feature selections, I found that XGBoost yielded the lowest RMSE.

Out of the 23 submissions I made, my more successful models included: Regression Tree, Bootstrap Aggregation, Tuned Random Forest, CVBoost, and my final model of XGBoost. Codes for these models are also included in this submission.