Analysis of Prices for Airbnb Listings in Major US Cities

Rahul Malhotra

12/22/2020

Abstract

The focus of the project is to design models which will be used to predict and interpret the price of various Airbnb listings in major US cities using several factors. Among the factors, the most important ones were determined while others were omitted due to a lack in predicting capability. These results are useful for Airbnb hosts looking to create a new listing, as they can prioritize their time and investments towards aspects of a listing that will yield them the most return on their investment. To conduct the analysis, a dataset containing 74,111 listings was used, which included 28 predictors and the response, which is the logarithm of a listing's price. Prior to any analysis, some predictors were omitted, such as those consisting of text, categorical variables with several (hundreds) classes, and listings with missing data. The final dataset consisted of 47,787 listings and 15 predictors. Then, analysis was done which included several different models from the three approaches: 1) linear, 2) non-linear, and 3) tree based. To evaluate the models, the test mean squared error (MSE) was reported and it was found that the random forest model performed the best.

Table of Contents

List of tables

Table 1: Summary of Full Linear Model (page 12)

```
##
## Call:
## lm(formula = log_price ~ ., data = traindata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.574 -0.260 -0.010  0.247  2.882
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -7.34e+01   1.99e+00  -36.89  < 2e-16 ***
## room_typePrivate room            -6.08e-01   5.06e-03 -120.16  < 2e-16 ***
## room_typeShared room             -1.13e+00   1.40e-02  -80.48  < 2e-16 ***
## accommodates                      8.06e-02   1.92e-03   41.87  < 2e-16 ***
## bathrooms                         1.29e-01   4.55e-03   28.29  < 2e-16 ***
## bed_typeCouch                     1.02e-01   4.92e-02    2.07   0.0385 *
## bed_typeFuton                     1.68e-02   3.51e-02    0.48   0.6322
## bed_typePull-out Sofa             1.06e-01   3.64e-02    2.92   0.0035 **
## bed_typeReal Bed                  8.90e-02   2.88e-02    3.09   0.0020 **
## cancellation_policymoderate       1.40e-02   6.29e-03    2.23   0.0261 *
## cancellation_policystrict         4.50e-02   5.90e-03    7.63  2.5e-14 ***
## cancellation_policysuper_strict_30  3.44e-01   5.17e-02    6.65  3.0e-11 ***
## cancellation_policysuper_strict_60  7.08e-01   1.66e-01    4.26  2.0e-05 ***
## cleaning_feeTRUE                 -1.07e-02   5.74e-03   -1.86   0.0622 .
## cityChicago                      -1.76e+01   3.77e-01  -46.64  < 2e-16 ***
## cityDC                           -6.07e+00   1.75e-01  -34.57  < 2e-16 ***
## cityLA                           -4.88e+01   1.12e+00  -43.75  < 2e-16 ***
## cityNYC                          -2.89e+00   8.43e-02  -34.33  < 2e-16 ***
## citySF                           -5.30e+01   1.18e+00  -44.83  < 2e-16 ***
## host_has_profile_picTRUE         -1.38e-01   5.35e-02   -2.59   0.0097 **
## host_identity_verifiedTRUE        1.26e-02   4.84e-03    2.60   0.0093 **
## host_response_rate               -1.16e-03   1.59e-04   -7.30  2.9e-13 ***
## instant_bookableTRUE             -3.61e-02   4.67e-03   -7.72  1.2e-14 ***
## latitude                          7.78e-02   3.01e-02    2.58   0.0098 **
## longitude                        -1.04e+00   2.27e-02  -46.02  < 2e-16 ***
## number_of_reviews                -2.96e-04   4.94e-05   -6.01  1.9e-09 ***
## review_scores_rating              6.58e-03   2.93e-04   22.43  < 2e-16 ***
## bedrooms                          1.53e-01   3.90e-03   39.12  < 2e-16 ***
## beds                             -4.86e-02   2.93e-03  -16.57  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.406 on 38200 degrees of freedom
## Multiple R-squared:  0.641,  Adjusted R-squared:  0.64
## F-statistic: 2.43e+03 on 28 and 38200 DF,  p-value: <2e-16
```

Table 2: Summary of Best Subset (Using Backwards Search) Model (page 12)

```
##
## Call:
## lm(formula = log_price ~ room_type + accommodates + bathrooms +
##     cancellation_policy + city + host_response_rate + instant_bookable +
##     latitude + longitude + number_of_reviews + review_scores_rating +
##     bedrooms + beds, data = traindata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.579 -0.260 -0.009  0.247  2.884
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       -7.34e+01   1.99e+00  -36.90  < 2e-16 ***
## room_typePrivate room             -6.09e-01   5.02e-03 -121.30  < 2e-16 ***
## room_typeShared room              -1.13e+00   1.36e-02  -83.43  < 2e-16 ***
## accommodates                       8.06e-02   1.92e-03   41.92  < 2e-16 ***
## bathrooms                          1.29e-01   4.55e-03   28.35  < 2e-16 ***
## cancellation_policymoderate        1.26e-02   6.21e-03    2.03   0.0423 *
## cancellation_policystrict          4.38e-02   5.77e-03    7.60  3.0e-14 ***
## cancellation_policysuper_strict_30 3.46e-01   5.17e-02    6.68  2.3e-11 ***
## cancellation_policysuper_strict_60 7.07e-01   1.66e-01    4.26  2.1e-05 ***
## cityChicago                       -1.76e+01   3.77e-01  -46.60  < 2e-16 ***
## cityDC                            -6.05e+00   1.75e-01  -34.53  < 2e-16 ***
## cityLA                            -4.88e+01   1.12e+00  -43.72  < 2e-16 ***
## cityNYC                           -2.89e+00   8.43e-02  -34.28  < 2e-16 ***
## citySF                            -5.29e+01   1.18e+00  -44.80  < 2e-16 ***
## host_response_rate                -1.17e-03   1.59e-04   -7.37  1.8e-13 ***
## instant_bookableTRUE              -3.70e-02   4.65e-03   -7.95  1.9e-15 ***
## latitude                           7.83e-02   3.01e-02    2.60   0.0094 **
## longitude                         -1.04e+00   2.27e-02  -45.99  < 2e-16 ***
## number_of_reviews                 -2.81e-04   4.90e-05   -5.74  9.8e-09 ***
## review_scores_rating               6.60e-03   2.93e-04   22.51  < 2e-16 ***
## bedrooms                           1.52e-01   3.90e-03   39.05  < 2e-16 ***
## beds                              -4.83e-02   2.93e-03  -16.49  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.406 on 38207 degrees of freedom
## Multiple R-squared:  0.64,   Adjusted R-squared:  0.64
## F-statistic: 3.24e+03 on 21 and 38207 DF,  p-value: <2e-16
```

Table 3: Summary of Ridge Regression Model (page 13)

(Intercept)		room_typePrivate room	
3.68042475		-0.58563710	
room_typeShared room		accommodates	
-1.07276570		0.07048091	
bathrooms		bed_typeCouch	
0.12940301		0.04954284	
bed_typeFuton		bed_typePull-out Sofa	
-0.00925592		0.07413371	
bed_typeReal Bed		cancellation_policymoderate	
0.07124274		0.01312612	
cancellation_policystrict		cancellation_policysuper_strict_30	
0.05442376		0.35029493	
cancellation_policysuper_strict_60		cleaning_feeTRUE	
0.69126385		0.01169062	
cityChicago		cityDC	
-0.28444530		-0.07980646	
cityLA		cityNYC	
-0.08862764		0.00570177	
citySF		host_has_profile_picTRUE	
0.29077746		-0.14744232	
host_identity_verifiedTRUE		host_response_rate	
0.01597623		-0.00125690	
instant_bookableTRUE		latitude	
-0.04542092		0.00923781	
longitude		number_of_reviews	
-0.00017420		-0.00019308	
review_scores_rating		bedrooms	
0.00658592		0.14118436	
beds			
-0.02512160			

Table 4: Summary of Lasso Model (page 13)

(Intercept)		room_typePrivate room	
4.09375377		-0.61081823	
room_typeShared room		accommodates	
-1.11845030		0.05909550	
bathrooms		cancellation_policystrict	
0.11476929		0.02824090	
cancellation_policysuper_strict_30		cityChicago	
0.10981840		-0.23787206	
cityDC		cityLA	
-0.05665595		-0.12244678	
cityNYC		citySF	
0.00134337		0.26212197	
host_response_rate		instant_bookableTRUE	
-0.00061221		-0.03191063	
review_scores_rating		bedrooms	
0.00538320		0.13832166	
beds			
-0.00158508			

Table 5: Comparing Test MSE of All Linear Models (page 13)

	Test MSE	# of Coefficients
Full Model	0.16779	29
Backwards Search	0.16798	21
Ridge	0.17536	29
Lasso	0.17776	18

Table 6: Summary of Natural Cubic Spline (page 14)

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-58.3537129	6.9748426	-8.36631	6.1441e-17
ns(accommodates, 3)1	0.4898953	0.0209085	23.43039	1.4899e-120
ns(accommodates, 3)2	1.0537529	0.0242701	43.41776	0.0000e+00
ns(accommodates, 3)3	0.9534154	0.0323334	29.48705	5.5978e-189
ns(bathrooms, 3)1	0.4249768	0.0280623	15.14405	1.1724e-51
ns(bathrooms, 3)2	0.8347432	0.0762509	10.94732	7.5297e-28
ns(bathrooms, 3)3	0.9206363	0.0652628	14.10660	4.4898e-45
ns(host_response_rate, knots = c(25, 50, 75))1	-0.0786087	0.0340796	-2.30662	2.1081e-02
ns(host_response_rate, knots = c(25, 50, 75))2	-0.0607350	0.0214553	-2.83077	4.6461e-03
ns(host_response_rate, knots = c(25, 50, 75))3	-0.1429822	0.0601203	-2.37827	1.7399e-02
ns(host_response_rate, knots = c(25, 50, 75))4	-0.0844335	0.0261102	-3.23374	1.2228e-03
ns(number_of_reviews, 3)1	-0.0253152	0.0223929	-1.13050	2.5827e-01
ns(number_of_reviews, 3)2	-0.0864924	0.0407759	-2.12117	3.3914e-02
ns(number_of_reviews, 3)3	-0.0426846	0.0875431	-0.48758	6.2585e-01
ns(latitude, 3)1	3.3071434	0.7205352	4.58984	4.4499e-06
ns(latitude, 3)2	3.7722446	0.6078584	6.20579	5.4979e-10
ns(latitude, 3)3	7.2018865	0.5384857	13.37433	1.0550e-40
ns(longitude, 3)1	128.1299897	7.6952920	16.65044	4.9748e-62
ns(longitude, 3)2	37.9578070	6.4871468	5.85123	4.9192e-09
ns(longitude, 3)3	78.7854318	7.5557664	10.42719	2.0156e-25
ns(review_scores_rating, knots = c(25, 50, 75))1	-0.0464150	0.0714856	-0.64929	5.1615e-01
ns(review_scores_rating, knots = c(25, 50, 75))2	-0.3385333	0.0486332	-6.96095	3.4344e-12
ns(review_scores_rating, knots = c(25, 50, 75))3	-0.6534395	0.1410253	-4.63349	3.6073e-06
ns(review_scores_rating, knots = c(25, 50, 75))4	0.1544079	0.0664968	2.32203	2.0236e-02
ns(bedrooms, 3)1	0.7168513	0.0273946	26.16758	1.3034e-149
ns(bedrooms, 3)2	1.0703859	0.0457353	23.40396	2.7461e-120
ns(bedrooms, 3)3	1.4128572	0.0946236	14.93134	2.8583e-50
ns(beds, 3)1	-0.1805314	0.0316145	-5.71041	1.1354e-08
ns(beds, 3)2	-0.9082023	0.0938611	-9.67603	4.0422e-22
ns(beds, 3)3	-1.2008392	0.0931106	-12.89691	5.6251e-38
room_typePrivate room	-0.5351861	0.0056367	-94.94666	0.0000e+00
room_typeShared room	-1.0255443	0.0140809	-72.83256	0.0000e+00
bed_typeCouch	0.1001616	0.0472468	2.11997	3.4015e-02
bed_typeFuton	0.0087909	0.0336823	0.26099	7.9410e-01
bed_typePull-out Sofa	0.0770797	0.0349441	2.20580	2.7404e-02
bed_typeReal Bed	0.0628103	0.0276877	2.26852	2.3303e-02
cancellation_policymoderate	0.0178894	0.0060811	2.94179	3.2652e-03
cancellation_policystrict	0.0477628	0.0057179	8.35323	6.8631e-17
cancellation_policysuper_strict_30	0.3656837	0.0497288	7.35356	1.9686e-13
cancellation_policysuper_strict_60	0.5604844	0.1596307	3.51113	4.4672e-04
cleaning_feeTRUE	-0.0135469	0.0055300	-2.44972	1.4301e-02
cityChicago	-13.1317705	1.6966482	-7.73983	1.0199e-14
cityDC	-13.8914079	0.4294941	-32.34366	1.9661e-226
cityLA	69.7241340	7.0962574	9.82548	9.3067e-23
cityNYC	-8.7176318	0.2408981	-36.18804	5.5993e-282
citySF	63.9631423	7.0028021	9.13394	6.9204e-20
host_has_profile_picTRUE	-0.1418863	0.0514188	-2.75942	5.7931e-03
host_identity_verifiedTRUE	0.0169476	0.0046736	3.62625	2.8794e-04
instant_bookableTRUE	-0.0311392	0.0045094	-6.90545	5.0826e-12

Table 7: Summary of Smoothing Spline (page 14)

```
## Anova for Parametric Effects
##                            Df Sum Sq Mean Sq  F value  Pr(>F)
## s(accommodates)             1   5399    5399 34088.22 < 2e-16 ***
## s(bathrooms)                1     80      80   504.66 < 2e-16 ***
## s(host_response_rate)       1     10      10    60.23 8.6e-15 ***
## s(number_of_reviews)        1      8       8    48.00 4.3e-12 ***
## s(latitude)                 1     47      47   296.93 < 2e-16 ***
## s(longitude)                1    289     289  1827.52 < 2e-16 ***
## s(review_scores_rating)     1    183     183  1154.20 < 2e-16 ***
## s(bedrooms)                 1    157     157   988.54 < 2e-16 ***
## s(beds)                     1     63      63   395.21 < 2e-16 ***
## room_type                   2   3231    1615 10198.80 < 2e-16 ***
## bed_type                    4     19       5    29.52 < 2e-16 ***
## cancellation_policy         4     60      15    95.12 < 2e-16 ***
## cleaning_fee                1      3       3    20.93 4.8e-06 ***
## city                        5   7808    1562  9858.43 < 2e-16 ***
## host_has_profile_pic        1      1       1     6.22   0.013 *
## host_identity_verified      1      3       3    20.44 6.2e-06 ***
## instant_bookable            1      9       9    58.69 1.9e-14 ***
## Residuals               38173   6046       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## Anova for Nonparametric Effects
##                          Npar Df Npar F   Pr(F)
## (Intercept)
## s(accommodates)               3   93.3 < 2e-16 ***
## s(bathrooms)                  3   14.3 2.5e-09 ***
## s(host_response_rate)         3    1.6    0.19
## s(number_of_reviews)          3    9.1 5.0e-06 ***
## s(latitude)                   3  157.5 < 2e-16 ***
## s(longitude)                  3   27.8 < 2e-16 ***
## s(review_scores_rating)       3  148.4 < 2e-16 ***
## s(bedrooms)                   3   44.3 < 2e-16 ***
## s(beds)                       3   24.9 4.4e-16 ***
## room_type
## bed_type
## cancellation_policy
## cleaning_fee
## city
## host_has_profile_pic
## host_identity_verified
## instant_bookable
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Table 8: Comparing Test MSE of Generalized Additive Models (page 14)

```
##                  Test MSE
## Cubic Spline      0.15377
## Smoothing Spline  0.16044
```

Table 9: Summary of Regression Tree (page 15)

```
##
## Regression tree:
## tree(formula = log_price ~ ., data = traindata)
## Variables actually used in tree construction:
## [1] "room_type" "longitude" "bathrooms" "city"
## Number of terminal nodes:  8
## Residual mean deviance:  0.19 = 7270 / 38200
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.7200 -0.2870 -0.0161  0.0000  0.2580  3.0600
```

```
## [1] "Regression Tree Test MSE: 0.18587"
```

Table 10: Bagging Results (page 15)

```
##      Test MSE # of Trees
## [1,]  0.24874          1
## [2,]  0.16256          5
## [3,]  0.15079         10
## [4,]  0.14590         20
```

Table 11: Random Forest Results (page 15)

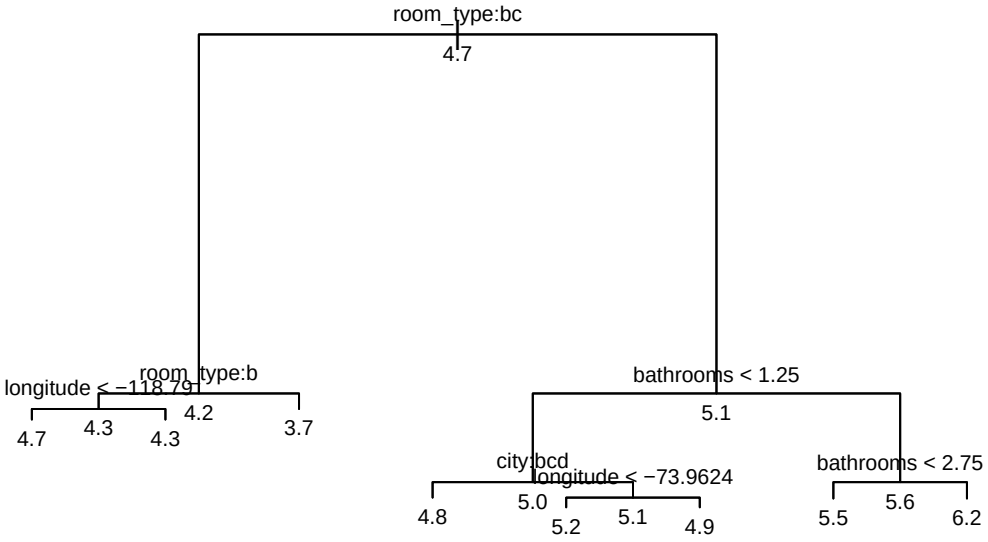
```
##      Test MSE # of Trees
## [1,]  0.22254          1
## [2,]  0.12060         10
## [3,]  0.11404         30
## [4,]  0.11307         50
```

Table 12: Boosting Results (page 16)

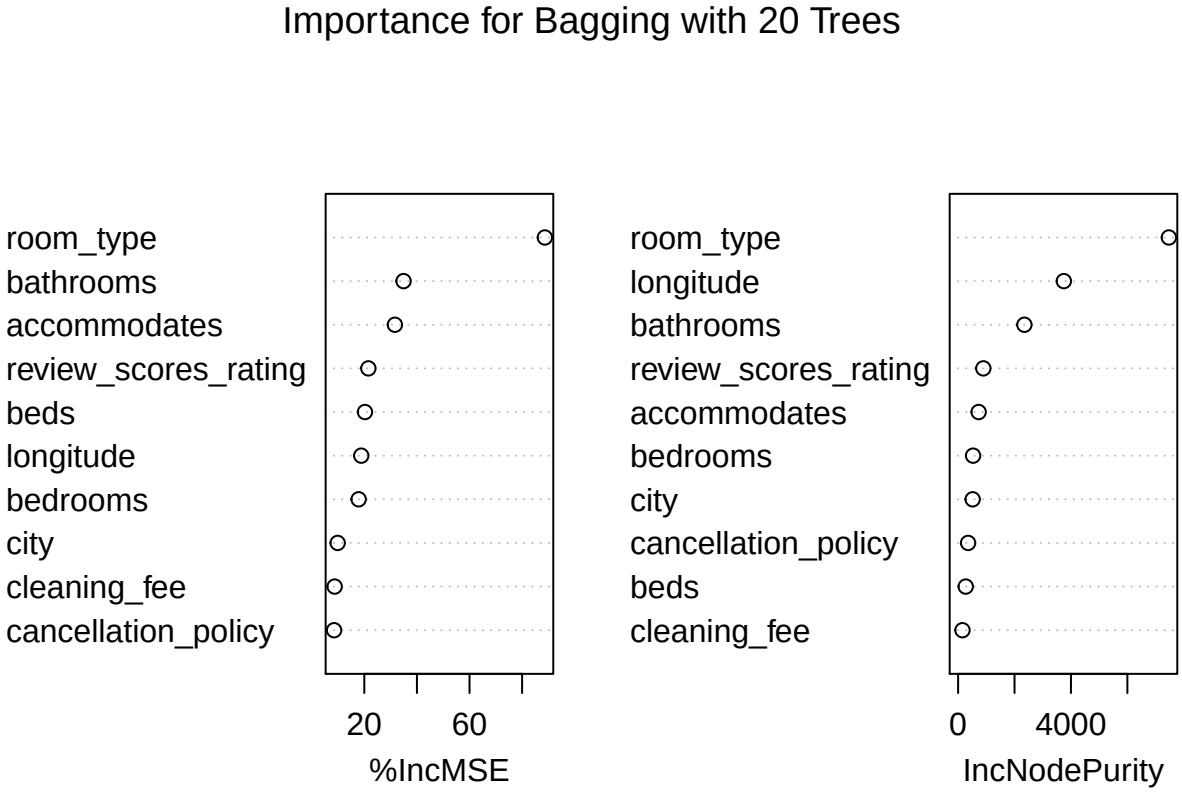
```
##                       100 Trees 500 Trees 1,000 Trees 2,000 Trees 3,000 Trees 4,000 Trees
## Interaction Depth = 1   0.16813   0.13832     0.13269     0.12855     0.12678     0.12579
## Interaction Depth = 2   0.14070   0.11840     0.11345     0.11046     0.10956     0.10892
## Interaction Depth = 3   0.13328   0.11269     0.10992     0.10831     0.10781     0.10767
## Interaction Depth = 4   0.12782   0.11024     0.10752     0.10671     0.10651     0.10708
```

List of graphs

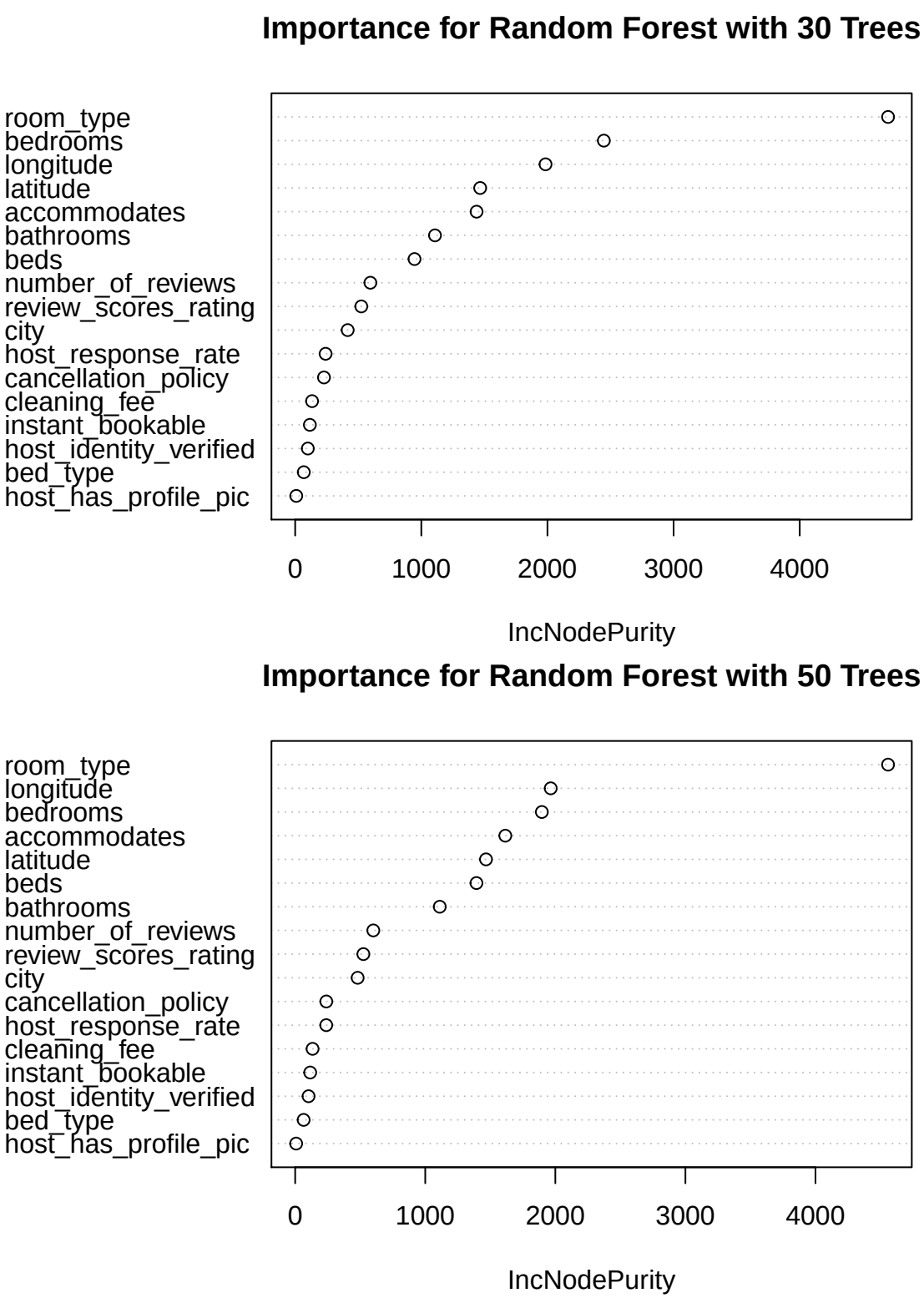
Graph 1: Regression Tree (page 15)



Graph 2: Importance of Variables in Bagging Model (page 15)



Graph 3: Importance of Variables in Random Forest Model (page 15)



Introduction

Founded in 2008, Airbnb is a company which hosts over six million worldwide listings for rental properties. Its main service is providing customers with a marketplace for a short-term property, usually a small home, to stay in during vacation. However, Airbnb itself does not own any of the properties. Instead, it allows hosts to list their properties on its platform in exchange for a commission. Airbnb's service has been on the rise in the last decade as an alternative to its main competition: hotels. Vacationers often find the utilties, such as a kitchen, and privacy of homes to be reasons for selecting Airbnb over hotels for their place to stay. In addition, hosts also compete with other listings on the platform. Thus, hosts must offer desirable listings to attract customers to choose theirs, rather than hotels or other listings.

By identifying which of the many factors involved within a listing contribute the most to its price, hosts can focus on them, both when creating a new listing or modifying a current one. As a result, they can maximize the price for the listing and in turn generate more profit. Within a listing, there are many different aspects that go beyond just what the property itself has. Some of these include the number and types of rooms, different policies and fees, and information about the listing and host such as reviws and ratings. A model that can identify the significance of these aspects towards a listing's price allows hosts to gauge which are worth investing time and money towards.

Linear Models

Linear models estimate coefficients for the intercept and each category for all chosen predictors, such that the residual sum of squares (RSS) is minimized. Although linear models do not account for non-linearity or interactions between features, if the true model is linear, they can serve as the best fit. The full model, containing all predictors, and simplified models, using different reduction methods, were built to assess how a linear approach does with predicting the price of listings.

Simple Linear Regression

Initially, a simple linear regression model was created including all of the remaining 15 predictors, giving us a total of 29 coefficients (including the intercept), to predict the log price. Interestingly, we can see from Table 1 that only one predictor, "bed_type", was not significant at the 5% level. However, having all 15 predictors, some of which have several categories, is likely to lead to overfitting of the data so our next objective was to try and simplify the model.

Backward Search

The first attempt at trying to simplify the model was using best subset selection with a backward search. To determine how many coefficients we should estimate in our model, we plotted various criteria, such as adjusted R-squared, Mallow's C_p, and BIC against the number of predictors. The BIC was chosen as the deciding criterion as it penalizes models with more predictors and found that it was minimized when the model had 21 coefficients (including the intercept). From Table 2, we can see that all of the predictors were significant, which is expected when doing a backward search. While this gave a simpler method, the backward search is a greedy algorithm, meaning that once a predictor is removed, it cannot be added back. Thus, predictors that have been removed may perform better in later stages so we were hesitant to choose this as the best linear model.

Ridge

Before using another method to simplify the model by removing some of the predictors, we used ridge regression to shrink the coefficients we had in the simple linear regression model. The reasoning for this was that some of the predictors we have are likely to be correlated with one another since they are somewhat related and ridge regression is able to help deal with this. To choose the value of lambda, our tuning parameter, we created a grid of several values and used cross validation to find the best value. We then created our ridge model and compared these coefficients (Table 3) to the ones in the initial model (Table 1). We found that for the "city" variable, the coefficient in the initial model was shrunk by a factor of 60 to 900 times in the ridge, depending on the specific city. Also, we found that "latitude" and "longitude" were shrunk by a factor of 10 and 6000 times, respectively. These variables displayed a lot of shrinkage, which we believe makes sense as they all have to do with location and were thus likely correlated with one another.

Lasso

For our final linear method, we implemented the lasso regression which both shrinks and omits some of the predictors from the simple linear regression model. Just as in the ridge implementation, we selected a grid of values for the tuning parameter, lambda, and chose the best one using cross validation. Table 4 shows the summary of the resulting model, which consisted of 11 predictors with a total of 18 coefficients (including the intercept). It had removed 11 of the coefficients from the initial model (Table 1), two of which were "latitude" and "longitude", but kept the "city" variable, likely due to multicollinearity and the extreme shrinkage we saw in the ridge model.

Comparing Linear Models

Table 5 show the test MSE calculated in each method. We see that the simple linear regression model gave us the smallest error. However, this is to be expected since it contains all the predictors and is likely overfitting the data, which is resulting in the small error. When we conducted the backwards search, the test error was marginally higher but only slightly simplified and used a greedy approach. The test errors for the ridge and lasso models are a bit greater than that of the simple linear method, but again, this is to be expected as they simplify the model and attempt to address some of the problems with the simple linear method, such as overfitting and multicollinearity.

Among these four methods, we believe the most appropriate linear method would be the lasso model, as the test MSE is only about 6% greater than that of the simple linear model, but is much simpler in terms of the number of estimates (coefficients) it contains.

Non-Linear/General Additive Models (GAM)

General additive model allows for a non-linear function for each variable with more accurate predictions, as well as nice interpretations and inferences. In addition, they allow for more flexibility in the model by increasing the number of knots while keeping the degree fixed, whereas in polynomial regression, the degree must be increased to introduce more flexibility. The degree is fixed in GAMs by making use of a piecewise cubic polynomial where the coefficients of the polynomial change based on where the knots are. The smoothness of the functions can be summarized by an effective degree of freedom. However, although an improvement to the linear models, this approach does not include and may miss important interactions between features. In addition, if the true model is linear, non-linear models will overfit the data. To assess how these models predict the price of listings, two different GAMs were built. In both models, step functions were used for the qualitative variables and for the quantitative variables, the natural cubic spline and smoothing spline were used and compared.

Natural Cubic Spline

The first GAM that was built utilized the natural cubic spline. For the quantitative variables, the default degrees of freedom used was 3. For certain features, such as "host_response_rate" and "review_scores_rating", knots were sepcified at each quartile based on where the function of the feature may vary rapidly, rather than specifying the degrees of freedom. From Table 6, we can see see that the most significant variables in the model are those that deal with location, notably "longitude" and "city". Also, we can see that the two categories "LA" and "SF" have the largest coefficients which may indicate that the most expensive Airbnb listings tend to be in California. This may also explain why "longitude" has a larger coefficient than "latitude" since the state of California is more vertical than it is horizontal.

Smoothing Spline

The second GAM utilized the smoothing spline which is a slightly modified version of the natural cubic spline. It can be thought of as a natural cubic spline with knots at every unique value of x_i, where x_i is the value of the input features for the ith listing. However, since there will be many more knots, and as a result many more degrees of freedom, compared to the natural cubic spline, a tuning parameter λ is used to control the roughness of the smoothing spline and hence, the effective degree of freedom. Table 7 shows the summary of the smoothing spline model's effects. Again, we can see that "longitude" and "city" are among the most significant variables, in addition to "accommodates", "bathrooms", "review_scores_rating", and "room_type".

Comparing GAMs

Table 8 shows the test error for each of the two splines used in the GAMs. As expected, both perform better than any of the linear models because they account for non-linearity in the relationship between the price and predictors, thus, allowing for more flexibility. However, the added flexibility does come with the risk of overfitting, especially if the true model is in fact linear. Comparing the two GAMs against one another, we see that the natural cubic spline performs slightly better than the smoothing spline, with a 4% smaller test MSE. Thus, since the smoothing spline has more degrees of freedom, the most approporiate GAM would be the one using a natural cubic spline as it has a smaller test MSE and does not overfit as much as the smoothing spline.

Tree Based Models

Tree based models offer a different and unique set of techniques to approach regression problems. Instead of considering the absolute quantity of a predictor, the predictor is segmented into a number of regions. Each of these regions contain a particular range of values for a predictor and can be thought of as nodes on a decision tree. In other words, we can think of a decision tree as taking a quantitiative variable and giving it different classes based on ranges of values. Then, each region of the predictor will have its own contribution to the response. In order to decide how to build the decision tree, a similar goal as in the linear approach is used where our target is to minimize the RSS. The first type of tree model that was built was a baseline approach which involved a simple regression tree. The following models that were built then used the decision tree as a building block to achieve a more refined tree based approach.

Regression Tree

Our initial tree based approach involved fitting a regression tree. The output indicated that only 4 of the predictors were usedin the tree construction. We also checked to see whether pruning the tree improved its performance. Comparing both trees, we found that a lower test MSE was obtained when using the unpruned tree. From the summary of the tree (Table 9), we can see that the four predictors included in the tree are "room_type", "longitude", "bathrooms", and "city", again, highlighting the importance of the location variables. The graph of the tree (Graph 1) gives a visualization of the tree and how the splits of the predictors are done.

Bagging

The first advanced tree based approach we tried was bagging. This involves creating multiple trees where each tree contains a different sample of the training set. In each node of the tree, different predictors are tried as "split candidates" to see which will minimize the RSS. Normally, we would try every predictor in our data, however since our dataset has too many predictors, we decided to use just the 10 most significant ones in order to reduce the running time of this approach. One benefit of bagging is that it is able to reduce variance and overfitting, even as the number of trees increase. In fact, as the number of trees gets larger, the error comes down, almost converging to a minimum. Table 10 shows the test MSE obtained using bagging for different number of trees. We can see that as get to around 20 trees, the decrease in the test MSE is fairly minimal and will not decrease much more as we increase the number of trees. Looking at the importance plots (Graph 2), we see variables "longtiude", "room_type", and "bathroom" appear as some of the most important variables, similar to what we saw in the GAMs.

Random Forest

The random forest approach is a slight advancement to the bagging method as it decorrelates the trees in the case of one very strong predictor. In fact, bagging is a special case of a random forest, specifically when we set the number of predictors to consider at each node to be the number of predictors in our dataset. Typically, we set the number of predictors to consider at each node, m, as $m \approx \sqrt{p}$, where p is the total number of predictors. Then from these m predictors, one is chosen as the actual predictor to split on, based on which minimizes the RSS. For our dataset, we set $m = 4$ as there are 17 predictors. Table 11 shows the test MSE obtained using the random forest approach and for different number of trees. We see the test MSE is lower than in any of the previous methods and beyond about 30 trees, the decrease in test MSE is minimal. The number of trees used in the random forest model is important to consider. Although the model will improve as the number of trees increase, it will overfit the data more as well. In Graph 3, the importance of each predictor is plotted when using 30 and 50 trees. In bot, we can see that "room_type" has the most importance by a large margin. In addition, "longitude", "bathrooms", "latitude", and "accommodates" also appear among the predictors with the most importance.

Boosting

Boosting is a similar tree based method to bagging and random forest but differs slightly in the construction of its trees. In the prior two methods, mulitple trees are built independently and then they are averaged. In boosting, the trees are dependent as they are built sequentially, meaning that each new tree is grown using information from the previously built trees. The model will slowly learn and update the tree based on where previous trees performed poorly. Table 12 shows the test MSE obtained with varying numbers of trees and the level of interaction depth. Although we expect the test MSE to decrease as the number of trees and interaction depth increase, overfitting can become an issue. We can see that as we get past about 1,000 trees the test MSE does not decrease by much. However, at this number of trees and beyond, we see the lowest test MSE among all of the models, especially for the boosting models with a higher level of interaction depth in its trees.

Comparing Tree Based Models

Of the tree based models, the regression tree is the simplest and has the highest test MSE when more trees are used in the other models. Of the refined tree approaches, we see bagging gets much better results than the regression tree and does not suffer from overfitting, unlike the random forest and boosting models. However, these two models do get a lower test MSE than any of the other models. Thus, if the number of trees is not an issue, boosting would be the best tree based model, followed by random forest. If it were an issue and we would want to prevent overfitting the data, then the bagging model would be best.

Conclusion

After having utilized all three approaches to this problem, we compared each, highlighting their advantages and disadvantages. For the linear methods, the main advantage is their simplicity, especially when implementing a lasso regression to both shrink and omit some predictors. The linear models are simpler than those produced by the non-linear methods, in terms of the number of coefficients to estimate, however, they fail to capture any nonlinearity. Thus, the linear approach, specifically lasso, is only best if the true model itself is linear.

When the true model is not linear, then the general additive models are able to generate models which may be closer to the truth, while also reducing the test error in comparison to the linear methods. In addition, they are much more computationally efficient compared to the tree based methods. However, they can result in worse overfitting compared to the linear models and still do not consider interactions between predictors.

The tree based models, specifically the boosting method, gave us the smallest test MSE among all models. However, we must note that this may be due to overfitting, as a large number of trees can result in too many splits of the data. Also, the tree based methods can be computationally inefficient and take a long time to generate, depending on the number of trees and variables that are chosen for each split.

In conclusion, the boosting approach offers the lowest test MSE for predicting the prices of Airbnb listings, but at a greater likelihood of overfitting the training data. Thus, it may be approproiate to decrease the number of trees when using this model, or implement an overfitting reduction technique. However, if the true model is simpler and linear, then the lasso model may be the best model to use for this problem.

References

Hastie, Tibshirani, and Friedman. The Elements of Statistical Learning. Springer, 2009, 2ed. Book version: https://web.stanford.edu/~hastie/ElemStatLearn/

Witten, Hastie, James, Tibshirani. An Introduction to Statistical Learning: With Applications in R. Springer, 2013, Corrected 8th printing. Book version: http://faculty.marshall.usc.edu/gareth-james/

Dataset

Deloitte. (2017). AirBnB listings in major US cities. Version 1. Retrieved April 15, 2020 from https://www.kaggle.com/rudymizrahi/airbnb-listings-in-major-us-cities-deloitte-ml