Predicting Airbnb Prices with Random Forest Methods

MAX KARSOK

3 DECEMBER 2018

I. Introduction

The final iteration of the model designed to predict Airbnb prices in New York City produced a root mean squared error (RMSE) of 54.43702. There were two unique characteristics to this model that resulted in comparatively accurate predictions and differentiated it from peer models:

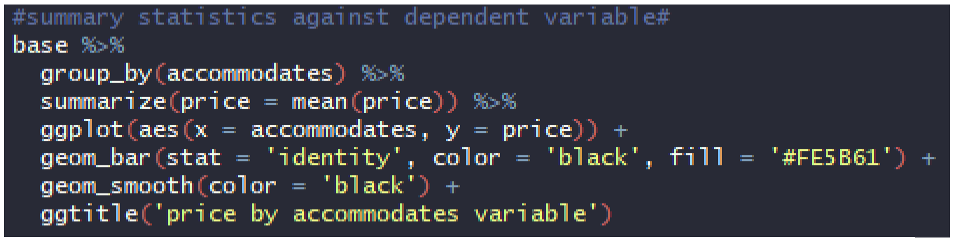
- Augmenting the provided data set with parameters from other data sources and generated parameters from the existing data set.
- Model comparisons resulting in the selection of specific random forest tuning parameters over a linear regression, linear support vector machine, or other decision tree methods.

Towards the close of the Airbnb Kaggle competition, this model ranked in the top 40% of 400 submissions. This work will summarize the data normalization and augmentation of the data set to prepare for modeling, the feature selection process, the comparison of evaluated models, the end results of the model's final iteration, and future considerations for continued model development or the model hypothetically integrating to an Airbnb production environment.

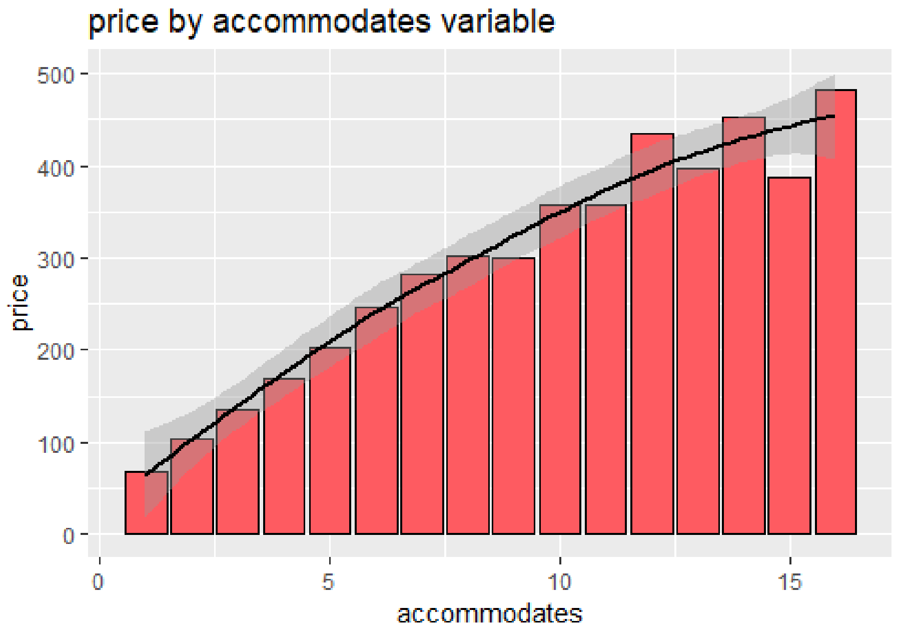
II. Exploring the Data

With price of the listing as the dependent variable in this model, the primary purposes of exploratory analysis were to understand (a) the completeness of the dataset and (2) potentially valuable features for the model as it relates to predicting price (using intuition alone). The data exploration process was conducted primarily through R packages **dplyr** (for simple aggregations) and **ggplot** (visualizations). While significant data exploration was conducted ahead of modeling, two examples will be provided for brevity purposes:

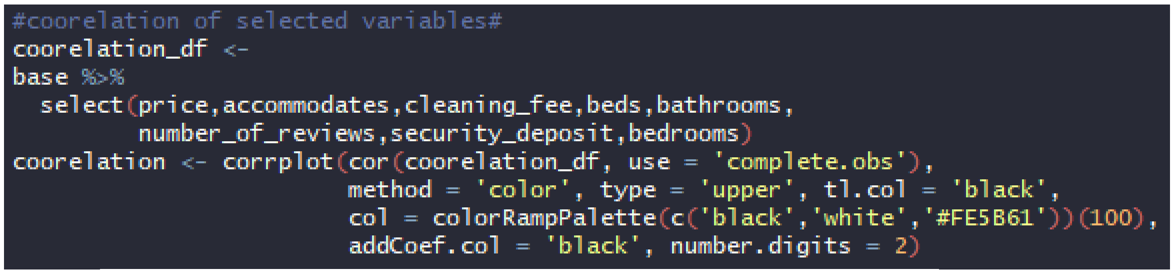
The first example leverages a common data process used in the exploration stage, which included aggregating the data in some fashion and then using an appropriate visualization to check for completeness and potential patterns. In the below example, average prices are calculated against the number of people the listing accommodates, confirming an intuitive guess I had thought while reviewing the dataset. This type of analysis not only helped assess accommodates as a viable variable, but also that it was a completely populated one. This sequence was conducted for multiple variables in the base dataset:



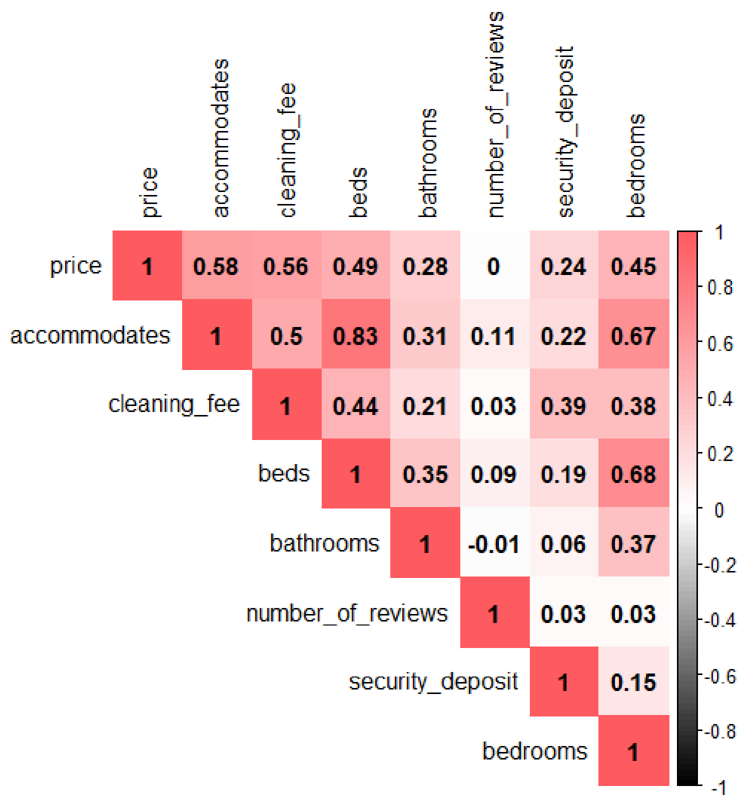
```
#summary statistics against dependent variable#
base %>%
  group_by(accommodates) %>%
  summarize(price = mean(price)) %>%
  ggplot(aes(x = accommodates, y = price)) +
  geom_bar(stat = 'identity', color = 'black', fill = '#FE5B61') +
  geom_smooth(color = 'black') +
  ggtitle('price by accommodates variable')
```



A second example examines correlations of some variables against the dependent variable and against each other. This exploration helped identify potentially significant attributes as well as multi-collinearity. While the final model was a random forest that accounts for multi-collinearity, this process was helpful during the exploratory stage to eliminate redundant attributes for increased processing efficiencies. Both these elements (correlation to price and multi-collinearity) were examined more scientifically once the modeling part of the process began, however this quick sample of some data points helped grasp the dataset:



```
#coorelation of selected variables#
coorelation_df <-
base %>%
  select(price,accommodates,cleaning_fee,beds,bathrooms,
         number_of_reviews,security_deposit,bedrooms)
coorelation <- corrplot(cor(coorelation_df, use = 'complete.obs'),
                        method = 'color', type = 'upper', tl.col = 'black',
                        col = colorRampPalette(c('black','white','#FE5B61'))(100),
                        addCoef.col = 'black', number.digits = 2)
```

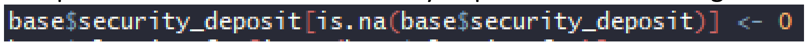


	price	accommodates	cleaning_fee	beds	bathrooms	number_of_reviews	security_deposit	bedrooms
price	1	0.58	0.56	0.49	0.28	0	0.24	0.45
accommodates		1	0.5	0.83	0.31	0.11	0.22	0.67
cleaning_fee			1	0.44	0.21	0.03	0.39	0.38
beds				1	0.35	0.09	0.19	0.68
bathrooms					1	-0.01	0.06	0.37
number_of_reviews						1	0.03	0.03
security_deposit							1	0.15
bedrooms								1

III. Preparing the Data for Analysis

Data preparation occurred in three phases during this process: (1) the modification of existing data, (2) integration of non-provided and publicly available data points, and (3) feature selection against the normalized dataset.

1. Modification of existing data (data cleansing): The first set of feature modifications were made on existing data points; this was the most significant effort in developing this model. Some of these data cleansing tactics managed NA values in different capacities; below, null values in the security deposit attribute were replaced with $0 on the assumption that there was no security deposit for these listings:



```
base$security_deposit[is.na(base$security_deposit)] <- 0
```

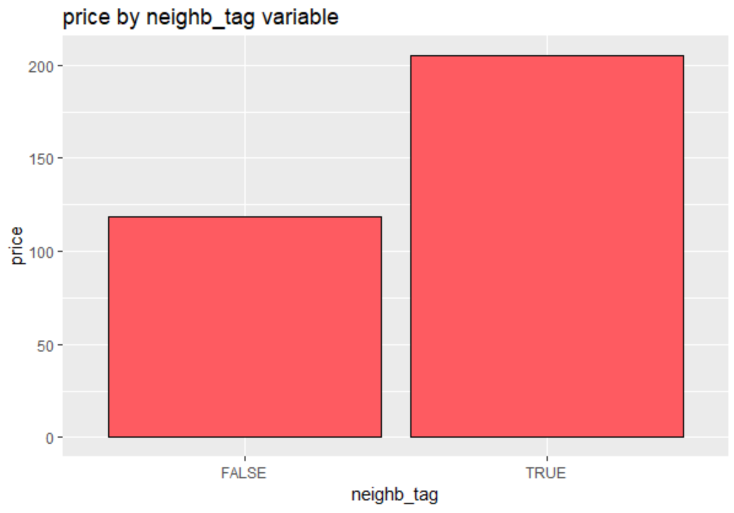
In other cases, replaces a numeric field with the mean of that column was the best normalization approach, especially if there were limited amounts of empty values. Here, the response rate variable is being normalized, and it is likely that if there is no response rate value, it is simply a data error and not response rate = 0 (this would mean the host never responds to inquiries). Using the mean for this value and others was more logical if this attribute was to ever be used in a model:

```
base$host_response_rate[is.na(base$host_response_rate)] <-
  round(mean(base$host_response_rate, na.rm = TRUE))
```

The neighborhood values proved to be significant in predicting price, however the variables were cumbersome and difficult to efficiently roll into a model. To account for this, a separate linear regression was computed using only the neighborhood_cleansed variable (lower level than the 5 boroughs but still cleaned and aggregated) to determine which neighborhoods were statistically significant in predicting price. These neighborhoods were then grouped together to form one variable that tagged each listing if it was located in a statistically significant neighborhood:

```
base$neighb_tag <- grepl("Tribeca|Flatiron District|SoHo|NoHo|Midtown|West Village|
                          Chelsea|Theater District|Greenwich Village|Kips Bay
                          `Hell's Kitchen`|Civic Center|Unionport|Murray Hill|Gramercy|
                          Carroll Gardens|Nolita|Financial District|
                          Battery Park City|Upper West Side",
                          base$neighbourhood_cleansed)
```

Neighborhoods with this variable = TRUE saw a $50+ increase in average price compared to those with the variable = FALSE (i.e. in or not in these selected neighborhoods):



A final concept leveraged to normalize the base dataset and create new, potentially significant attributes was leveraging very basic text aggregations like above. In the below example, the ‘amenities’ field is examined and it is clear that amenities of the listing are separated by a ‘,’ character in the field (Ex: “TV, Wifi, Hot Water”). While advanced sentiment to understand the impact of each of these amenities was out of the scope of the project, a simple aggregation to count the number of times a pipe character was included indicates the number of amenities (+1) the listing has provided. This proved to be an influential variable included in the final model:

```
base$amenities_cnt <- str_count(base$amenities, ',') + 1
```

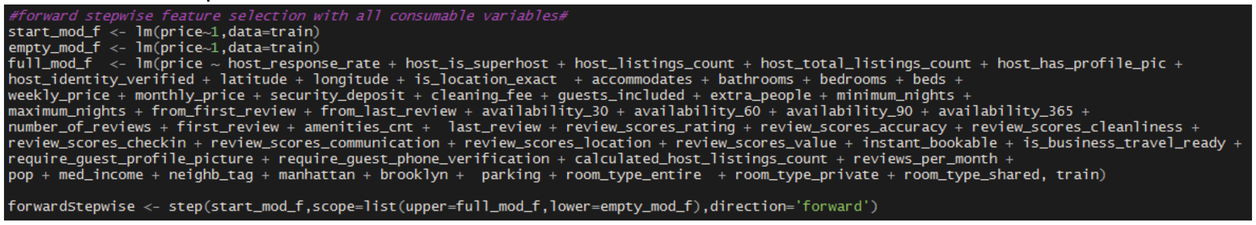
2. Integration of Census Data: The second major feature addition to the model was the incorporation of non-provided census bureau data. The total population and median household income for each zip code in New York City was added to each listing through a publicly available data source here: https://factfinder.census.gov/. Testing all available census attributes was not in the scope of the project, so intuition and for model interpretability, only zip code population and median household income were included. Income proved to be a very significant attribute in the model, while population only moderately significant:

```
zip_data <- read.csv('ny_zip_LOD.csv')
base <- merge(x = base, y = zip_data, by = 'zipcode', all.x = TRUE)
base$med_income[is.na(base$med_income)] <- round(mean(base$med_income, na.rm = TRUE))
base$pop[is.na(base$pop)] <- round(mean(base$pop, na.rm = TRUE))
```

For reference, the .csv referenced in the submission code titles ‘ny_zip_LOD.csv’ will be included so that results can be re-produced.

3. Feature Selection:

To identify which of the 53 prepared features were best suited for analysis, a forward stepwise selection was executed. This process brings variables into the model one at a time based on their significance to the dependent variable. This process selected 39 variables to be included in any modeling done on the dataset. Forward stepwise selection was used and the feature selection process (as opposed to lasso, backward stepwise, etc.) for two reasons: (1) interpretability of the results for a non-technical user, and (2) the ability to eliminate variables with high collinearity. By analyzing the VIR for the parameters chosen from different feature selection methods, the forward stepwise was selected. The stepwise code is shown below:



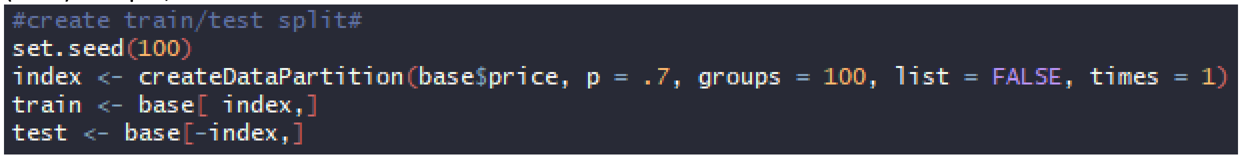
```
#forward stepwise feature selection with all consumable variables#
start_mod_f <- lm(price~1,data=train)
empty_mod_f <- lm(price~1,data=train)
full_mod_f  <- lm(price ~ host_response_rate + host_is_superhost + host_listings_count + host_total_listings_count + host_has_profile_pic +
host_identity_verified + latitude + longitude + is_location_exact  + accommodates + bathrooms + bedrooms + beds +
weekly_price + monthly_price + security_deposit + cleaning_fee + guests_included + extra_people + minimum_nights +
maximum_nights + from_first_review + from_last_review + availability_30 + availability_60 + availability_90 + availability_365 +
number_of_reviews + first_review + amenities_cnt +  last_review + review_scores_rating + review_scores_accuracy + review_scores_cleanliness +
review_scores_checkin + review_scores_communication + review_scores_location + review_scores_value + instant_bookable + is_business_travel_ready +
require_guest_profile_picture + require_guest_phone_verification + calculated_host_listings_count + reviews_per_month +
pop + med_income + neighb_tag + manhattan + brooklyn +  parking + room_type_entire  + room_type_private + room_type_shared, train)

forwardStepwise <- step(start_mod_f,scope=list(upper=full_mod_f,lower=empty_mod_f),direction='forward')
```

For reference, the top variables selected by the forward stepwise are shown below:

Rank (generated by AIC in forward stepwise)
1 – *accommodates* (provided, not modified)
2 – *median household income* (user added via census)
3 – *cleaning fee* (provided, NA = $0)
4 – *Manhattan flag* (user created, logical variable if listing located in Manhattan
5 – *Room Type = Entire* (user created, logical variable if listing was the entire apartment)

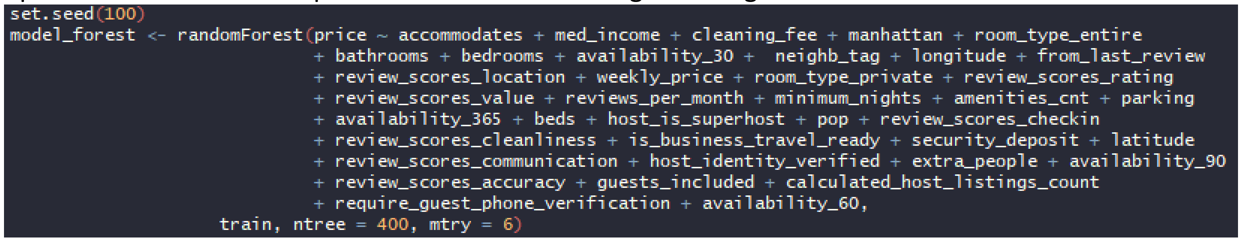
The final step before testing modeling techniques against the dataset was to split the base set into a train (70%) and test (30%) sample, done so below:



```
#create train/test split#
set.seed(100)
index <- createDataPartition(base$price, p = .7, groups = 100, list = FALSE, times = 1)
train <- base[ index,]
test <- base[-index,]
```

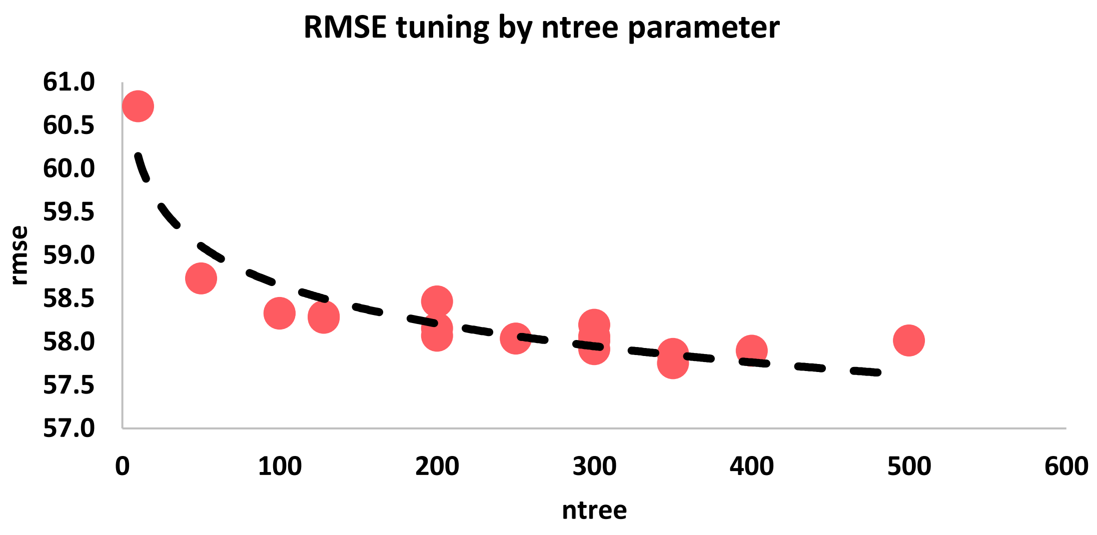
IV. Modeling Techniques

The random forest model chosen to predict the provided Airbnb listings creates 400 decision trees, evaluating up to 6 variables at each split in each tree. The code generating this model can be seen below:



```
set.seed(100)
model_forest <- randomForest(price ~ accommodates + med_income + cleaning_fee + manhattan + room_type_entire
                         + bathrooms + bedrooms + availability_30 +  neighb_tag + longitude + from_last_review
                         + review_scores_location + weekly_price + room_type_private + review_scores_rating
                         + review_scores_value + reviews_per_month + minimum_nights + amenities_cnt + parking
                         + availability_365 + beds + host_is_superhost + pop + review_scores_checkin
                         + review_scores_cleanliness + is_business_travel_ready + security_deposit + latitude
                         + review_scores_communication + host_identity_verified + extra_people + availability_90
                         + review_scores_accuracy + guests_included + calculated_host_listings_count
                         + require_guest_phone_verification + availability_60,
                  train, ntree = 400, mtry = 6)
```

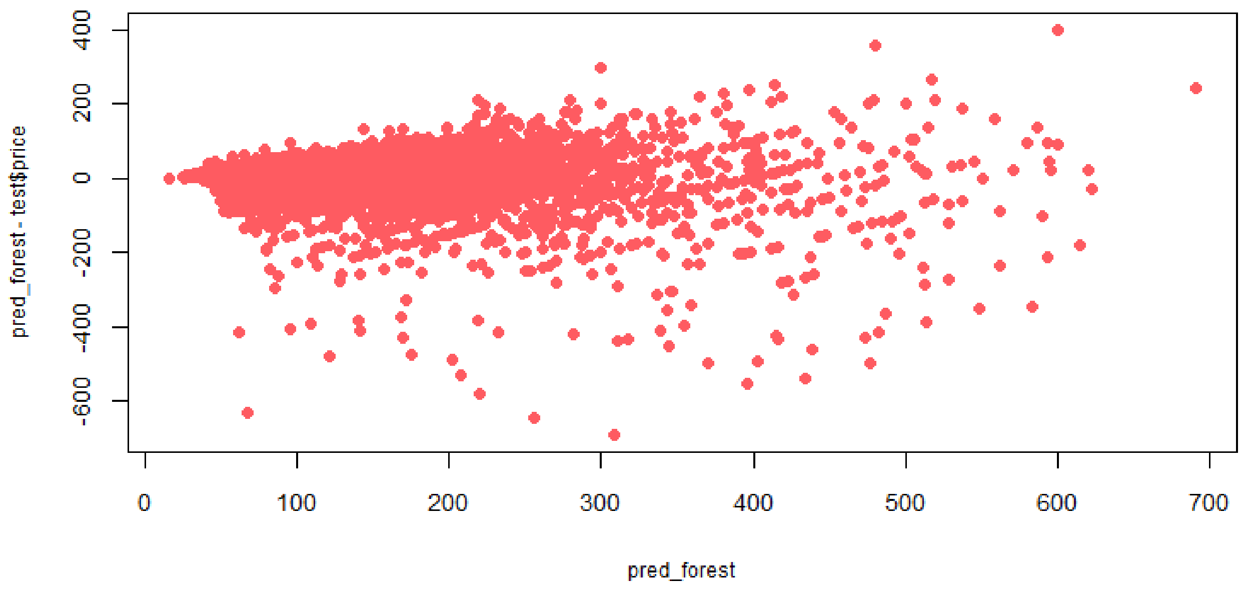
In the randomForest package, the number of trees in the random forest and the number of variables to be examined at a given split are the two user-definable parameters. After several manual iterations of testing, the combination of ntree = 400 and mtry = 6 provided the lowest RMSE on the test dataset. See below for the different combinations of these parameters and the resulting impact on the test RMSE:



The decision to build a random forest model with the selected features was made in comparison to a linear regression model, linear support vector machine model, and other tree models. These models were not as accurate using the same features as the random forest (ex: for comparison, a linear regression using the same features generates an RMSE of 64.94332).

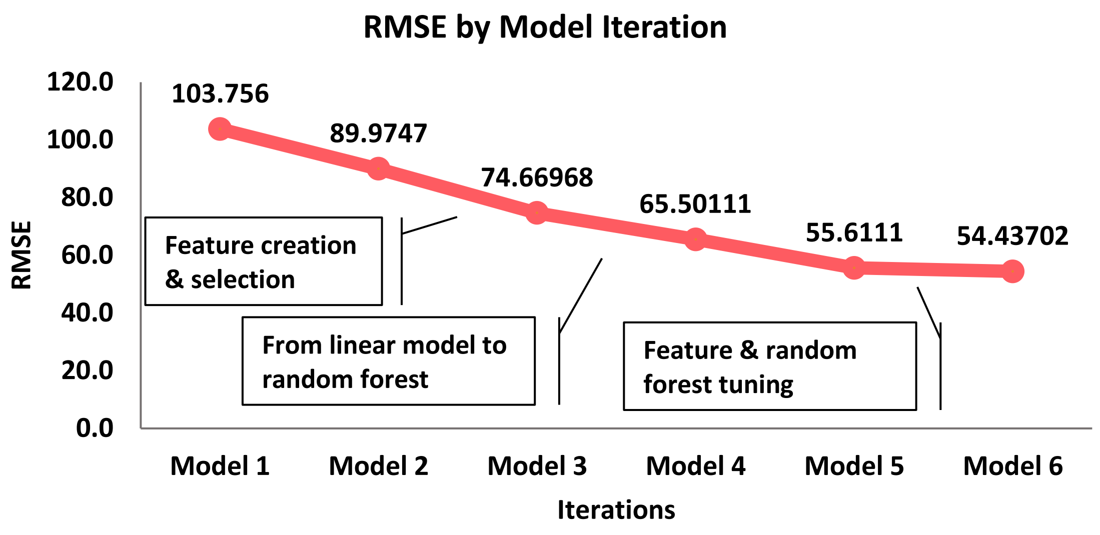
V. Results

The final model generated an RMSE of 54.43702 and the following residuals:



Analysis of the residual plot indicates that the model's most inaccurate predictions were more commonly less than the actual price value as opposed to greater than the actual price value. The majority of the residuals just slightly overestimate the actual price of the listing, and the outliers tend to underestimate the actual value.

The model's performance improved drastically over time as various levers were pulled. Below illustrates the various iterations and relevant points along the development of the final model to arrive at the RMSE below 55:



VI. Discussion

I was satisfied with the results of the feature selection process and random forest model implementation against the Airbnb data set. There were three areas that I believe could have been valuable for increasing the model's predictive power: (1) scraping and predicting prices over an entire year as opposed to just three days in March, (2) further analyzing the text attributes in the dataset, (3) leveraging historical prices of the same listing, and (4) leveraging unsupervised statistical methods.

1. <u>Elements of Seasonality:</u> Each listing scraped in both the base and scoring dataset was posted on Airbnb between 03/04/2018 and 03/06/2018. While this was adequate for this exercise because all postings were taken during the same time period, I feel that an influential variable in this analysis could be the date that listing was posted. For example, an Airbnb owner in Midtown is likely charging a different price for an apartment near Times Square on New Year's Eve compared to a Tuesday night in April.
2. <u>Further text analysis:</u> Some of the most valuable attributes in the random forest model were very simple aggregations of text fields in dataset (as seen above, and another example below):

```
base$parking <- grepl('Free street parking|Free parking on premises', base$amenities)
```

   Here, a variable called 'parking' is created and leverages a simple function that assigns a 1 or a 0 to a listing if "Free street parking" or "Free parking on premises" are found in the amenities text field. More technical sentiment analysis on these fields would likely uncover additional patterns about these listings, but was out of scope for this model.
3. <u>Historical prices of the same listing ID:</u> If this price prediction model were ever to move to production at Airbnb, it could leverage the historical prices of the same listing to predict future prices of that listing. For example, if the listing has been rented 100 times in the past three years, the model can consume the average value of the listing over time, account for potential inflation, and leverage that history to predict what the current and future value of that listing should be.
4. <u>Unsupervised methods:</u> While not covered in APAN Frameworks I, I believe that unsupervised statistical methods such as K-Means clustering could be valuable for analyzing and predicting these Airbnb listing prices. Because of the geographical components of the housing market, as well as the ability to group similar listings by common features makes unsupervised methods a logical next step to take this model.

VII. Citations

James et. al (2017). An Introduction to Statistical Learning. Springer Texts in Statistics. Springer. Retrieved on September 15, 2018.

U.S. Census (2018). American FactFinder. The U.S. Census. Retrieved on November 1st, 2018, from https://factfinder.census.gov/faces/nav/jsf/pages/index.xhtml

The following R packages were utilized to develop this model:
dplyr, ggplot2, tidyr, sqldf, caTools, caret, car, glmnet, rpart, rpart.plot, randomForest, corrplot, stringr, e1071.