Stella Huang
STATS 202A
14 December 2020

# Analyzing Airbnb Rental Prices in Los Angeles

## Introduction:

Airbnb has transformed the traditional travel and lodging experience; instead of traditional hotel rooms, users can filter on a map to find lodging options suitable to match their ideal location and needs. Travelers can save by renting private rooms in apartments or experience living in beach-side vacation homes, thus having more flexibility over their finances. However, each Airbnb listing page contains an abundance of information- reviews, host information, amenities, neighborhood- that make assessing the listing's value for the price difficult. This is crucial for budget travelers since lodging can be pricey.

This report analyzes the factors influencing Airbnb rental prices in the greater Los Angeles region. The report will explain the dataset, variable selection methods, regression results, findings, and future steps.

## About the Dataset:

The dataset was obtained from the third-party website Inside Airbnb[1]. The website owners collected data on rental listings in various cities by web-scraping the Airbnb website and update it monthly.

This dataset contains 30,500 Airbnb listings in the Los Angeles region from 2011 to October 2020. For each listing, there is information about the rental price, neighborhood, bathroom/bedroom counts, property type, geographic coordinates, number of reviews, availability, and maximum/minimum nights to stay, etc. We also have information on the listings' hosts, such as their verification status, Airbnb user age, number of total listings, and response/acceptance rate, etc.

With a mixture of quantitative and qualitative variables, the raw dataset was reduced and reshaped for performing regression. First, features containing website links and neighborhood overviews were excluded from the analysis. Qualitative variables, such as a comprehensive list of amenities and the date since the late review, were converted to quantitative variables to reflect the number of amenities and years since the last review. Instead of using neighborhoods, I classified them by their respective regions using a list compiled by the L.A. Times[2]. Features with a correlation above 0.75 were removed from the dataset too. It is imperative to mention that features like the listing description and property type (90 different types found) were excluded from the analysis because they could not be quantified or further summarized. Lastly, observations with missing values were removed.

After removing and feature engineering variables, the final dataset had 33 features.

## Exploratory Analysis:

This analysis aimed to find variables that strongly influence the listing prices from five tourist-popular regions: Santa Monica Mountains, Westside, Central L.A., South L.A., and

[1] Cox, Murray. Inside Airbnb. Updated 09 October, 2020. http://insideairbnb.com/about.html
[2] Los Angeles Times. Neighborhoods and Regions. http://maps.latimes.com/neighborhoods/neighborhood/list/

Eastside. I hypothesize that the longitude impacts the price. Specifically, I predict higher rental prices in Malibu, lower prices in Hollywood/DTLA, and the lowest prices in Eastern L.A, all based on the common property types and activities/sites found there. This analysis intends to help budget travelers identify potential aspects to cut costs down on. Should the model achieve a high prediction accuracy, it can be used to assess the quality of a listing; if the listed price is below the estimated price, the listing may be a good deal.

The preliminary analysis consisted of variable selection and individual examinations of the features against the response variable. Kernel regression allows us to examine the non-linear relationship between two continuous variables, thus I will explore the relationship between price and longitude. In the kernel density estimate in Figure 1, we examine that prices on the Westside are relatively higher and are estimated by a wider confidence interval, implying that prices greatly vary. To assess the kernel density estimate, I created a map of listings color-coded by their rental prices shown in Figure 2. While the KDE oversimplifies the price estimates, its 95% confidence interval captures the variation in prices on the coastal side, while rental prices in Central/Eastside L.A. are more homogenous.

Another variable possibly relevant to the price is the room type. There are four room types present in the dataset- shared rooms, private rooms, hotel rooms, and entire homes/apartments. The boxplot in Figure 3 displays the rental price per room type and shows that shared rooms are significantly cheaper in all regions. Another interesting observation is that listings in the Santa Monica Mountains only offer private rooms and entire houses (with no outliers too), while Westside and Central L.A. offer all types. Furthermore, most outliers for all room types are highly-priced rather than underpriced. This suggests that good bargains would only be reasonably priced, but never significantly below market value, and thus fuels the need to find which features make a worthy deal.

**Methods:**

The dataset contained 12,300 samples and was partitioned using a 70-30 train-test split. In the analysis, I trained a step-wise linear regression model and identified significant predictors by interpreting their p-values and coefficients. To verify this approach, I implemented recursive feature elimination on a linear model and random forest and compared the top variables selected with ones found in step-wise regression.

The table below shows the models' performance on the training and testing datasets. Given the similar performances on both datasets, the models most likely did not overfit, thus confirming that the following analysis on variable coefficients and importance has some degree of accuracy and reproducibility.

| Method | Train RMSE | Train Adjusted $R^2$ | Test RMSE | Test Adjusted $R^2$ |
|---|---|---|---|---|
| Linear Regression with all features | 0.4160 | 0.6502 | 0.4233 | 0.641 |
| Step-wise Regression | 0.4184 | 0.6449 | 0.4181 | 0.657 |
| Random Forest | 0.3498 | 0.7554 | 0.3423 | 0.775 |

**Linear Regression with All Predictors:**

The initial model predicts price by building a multilinear model using all 32 predictors. A logarithmic transformation was applied to the response to meet model assumptions, which also improved the Adjusted $R^2$. The linear model summary shows that 27 predictors are significant at the 95% confidence level, with the longitude, number of bedrooms, availability in the next 60 days, room type, number of amenities, and have private bathrooms (binary variable) as the most significant ones. By inspecting the p-values, one can notice that the most significant variables describe the listings themselves rather than the host.

The residual plots in Figure 6 show that variance is relatively constant across the residuals. While there are three outliers, it is not too concerning since the training dataset is sufficiently large to offset this occurrence. The residuals mostly follow a normal distribution, as indicated by the nearly straight line in the normal QQ plot. Moreover, this is a significant improvement over predicting the original variable because the previous model failed to meet model assumptions.

**Stepwise Regression:**

While most predictors were significant in the linear model, I employed forward-backward step-wise regression to reduce predictors and simplify the model. The Adjusted $R^2$, Cp, and BIC plots (Figure 7) indicate that the model performs best using 26 predictors, however, I opted for a subset of 23 predictors since the improvement in those metrics was minimal for larger subsets.

Since the response variable is log-transformed, we can interpret the transformed coefficient $\left(e^{\widehat{\beta}_i} - 1\right) * 100$ as the percentage change resulting from a unit increase of predictor $X_i$.[3] Such coefficients can be telling of the variables' influence on price. The model summary in Figure 8 shows that latitude and longitude are associated with the greatest change in the rental price. The latitude in the dataset spans (33.91284, 34.1676) and a unit increase (i.e. moving north) decreases the price by 70.15%, while the longitude runs between (-118.9342, -118.1281) and yields a 66.21% decrease per degree (i.e. moving east). The coefficients of the region variable also reflect this; listings in South L.A. yield a 20.78% decrease, listings in East L.A. yield a 9.4% decrease, while listing in Central L.A. and the Santa Monica Mountains (Northeast coastal area) yield a 14.86% and 17.62% increase. This confirms the simple pattern captured by the kernel regression between longitude and price. It also confirmed my initial hypothesis that listings closer to the coast are priced higher.

We can further inspect variables related to in-unit facilities and the guest experience. Privacy plays an immensely crucial role; an upgrade from shared rooms to private rooms, hotel rooms, or entire apartments increases the price by 39.84% each. Upgrading from a private room to an entire apartment increases the price by nearly 80%! Having a private bathroom and having an additional bed both separately raise the cost by 36.6% too. The number of amenities increases the price by 0.62% per item but surely can add up; for instance, the median found is 27 items, which totals to a 17% increase in price. Surprisingly, to obtain the "superhost" status, Airbnb hosts must maintain a 90% response rate, 1% cancellation rate, 4.8 overall ratings, and have leased for over 100 nights. While superhost-owned listings are found to be 6.6% more expensive, the host and living experience are certainly credible.

[3] https://data.library.virginia.edu/interpreting-log-transformations-in-a-linear-model/

**Linear Model with Recursive Feature Elimination (RFE):**

To validate my approach of inspecting variable p-values and coefficients, I applied recursive feature elimination on a linear model to extract the most important variables. The results are shown in Figure 9. Interestingly, the availability in the next 60 days (measured by days) and the host's number of shared rooms listed are selected. The step-wise linear regression model shows that a unit increase changes the price by 0.35% and -1.73%, respectively. This implies that hosts with more shared rooms listed will offer slightly cheaper rental prices. While this appears to be a lot, most hosts in our dataset have very few listed, hence perhaps is not the most useful information. The only caveat is that RFE only accepts quantitative variables, thus the listings' regions were excluded from the procedure. Overall, the top variables selected by RFE were significant in stepwise regression too.

**Random Forest:**

A random forest model was trained on 500 trees and 11 predictors (conventionally, 1/3 of predictors are recommended) per split. This model outperformed the linear regression model on the test dataset with a 75.5% Adjusted $R^2$. With improved and consistent performance, this model can also help validate and verify findings from the step-wise regression model.

As shown in Figure 10, This model ranked the minimum nights for reservation, longitude, latitude, room type, number of listings within 3 miles, and bedrooms as the most important variables. In the step-wise regression model, the minimum nights required is only associated with a -0.11% change in price. Most listings required 1-30 days, which at most decreases the rental price by 3.3%; a possible explanation is that this variable's interaction or association with other variables makes it significant. The number of listings within 3 miles was even excluded from the step-wise model; this is perhaps because each listing, on average, has 1200 listings nearby, thereby making the coefficient very small hence insignificant. Besides that, the other selected variables were also significant predictors in the stepwise regression model. Results from random forest and recursive feature elimination add confidence to results from step-wise regression- the variables associated with great percentage changes in price are indeed significant.

**Future Steps:**

The original dataset is extremely informative. For each listing, it recorded the image URL, listing description, neighborhood description, and specific amenities offered, etc, which all could be relevant to the rental price. One great area for improvement is to extract information (Natural Language Processing/Image Analysis) from these fields and transform them into variables- essentially, to quantify these measures of quality. As the analysis shows that location greatly influences the price, there is potential to further classify regions through geospatial analysis, instead of using the designated regions or neighborhoods. Another area to explore is outlier detection; simply using basic statistics may cause type 1 and type 2 errors, thus it is crucial to formulate some criteria for detecting anomalies based on multiple variables to render more accurate regression results.

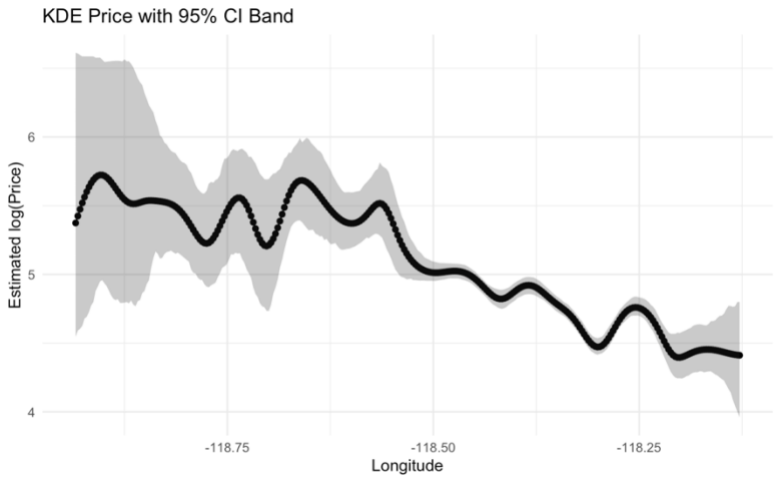
**Conclusion:**

This report has analyzed features strongly associated with Airbnb rental prices in Los Angeles through interpreting p-values and coefficients from a stepwise linear regression model. This approach was validated by comparing results from employing recursive feature elimination and random forest, which showed to be relatively consistent. In summary, the location (coordinates and region) and privacy (property type, private bathrooms, bedrooms) are the two biggest factors. Features related to both aspects can raised the price by over 30%, which travelers should consider when reserving accommodations. Furthermore, the analysis has identified inconspicuous and hidden factors like the minimum nights required to stay and the number of amenities, where their impact on rental prices show with small/large quantities.

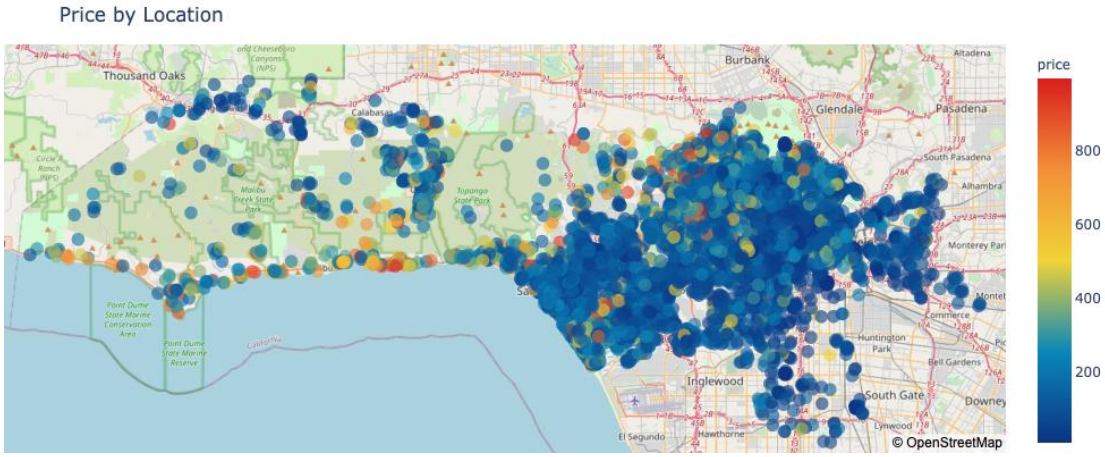
## Appendix

Graphics

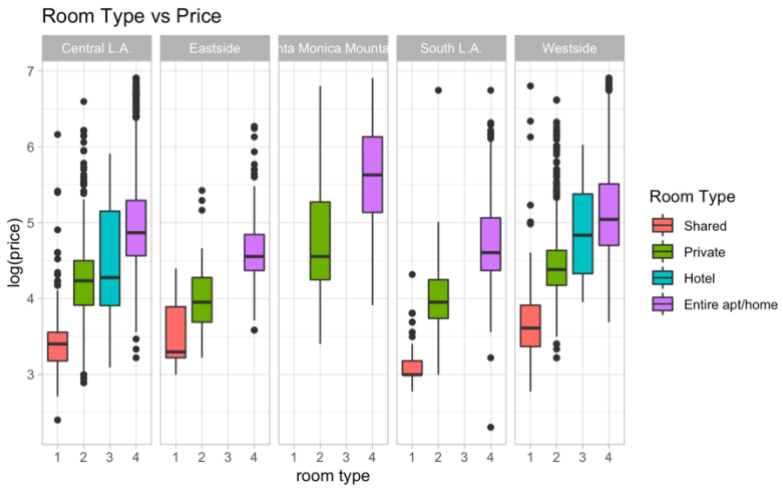
1. Kernel Regression




2. Map of Prices by Location




3. Prices for Different Room Types in Regions

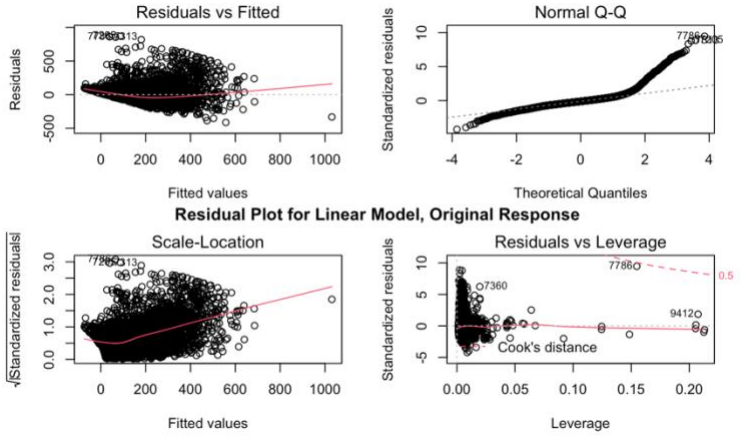


4. Model Summary: Linear model with all 32 predictors

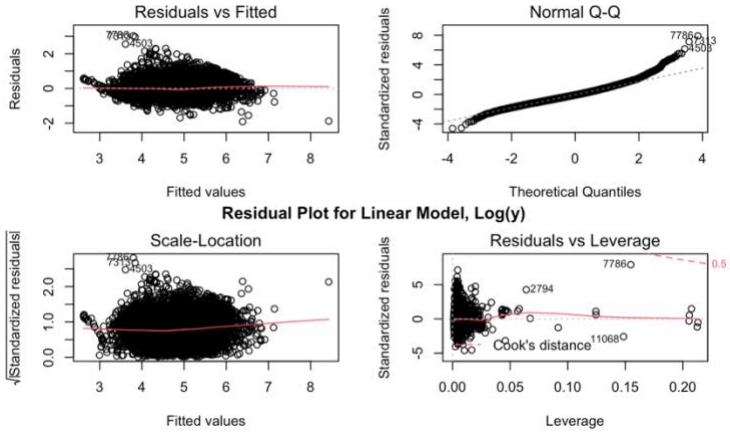
```
Call:
lm.default(formula = log(train_y) ~ ., data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1.91059 -0.26924 -0.03175  0.23160  3.02739

Coefficients:
                                              Estimate         Std. Error t value             Pr(>|t|)
(Intercept)                                -74.01377487227204  14.29203275125994  -5.179  0.00000022855477571 ***
host_response_rate                          -0.00063170111146   0.00015420738936  -4.096  0.00004234209113215 ***
host_acceptance_rate                         0.00049566784292   0.00018230124716   2.719             0.006562 **
host_is_superhost                            0.06042298675214   0.01116983029326   5.409  0.00000006491375933 ***
host_listings_count                          0.00003279691296   0.00003531128663   0.929             0.353022
host_identity_verified                       0.06021577116704   0.01469251060656   4.098  0.00004198583680786 ***
latitude                                    -1.18117143271148   0.16503950310651  -7.157  0.00000000000089329 ***
longitude                                   -0.98663197188358   0.11298348564177  -8.733 < 0.0000000000000002 ***
bathrooms                                    0.05430070014509   0.00650818540067   8.343 < 0.0000000000000002 ***
bedrooms                                     0.30310688705382   0.00683361463837  44.355 < 0.0000000000000002 ***
beds                                         0.00962887101723   0.00414305985789   2.324             0.020144 *
minimum_nights                              -0.00104588638530   0.00014935316952  -7.003  0.00000000000269894 ***
maximum_nights                               0.00000319977389   0.00000918843012   0.348             0.727669
maximum_nights_avg_ntm                       0.00000000031335   0.00000000008793   3.563             0.000368 ***
availability_60                              0.00364090042507   0.00023565881869  15.450 < 0.0000000000000002 ***
availability_365                            -0.00007220270633   0.00004120907774  -1.752             0.079791 .
number_of_reviews                           -0.00036637275184   0.00009442483618  -3.880             0.000105 ***
number_of_reviews_l30d                       0.01313279616065   0.00395342733334   3.322             0.000898 ***
review_scores_rating                         0.00559396545570   0.00071774335440   7.794  0.00000000000000726 ***
review_scores_checkin                       -0.05994798476960   0.00797925295210  -7.513  0.00000000000006361 ***
review_scores_location                       0.03330521619018   0.00810476798919   4.109  0.00004004987715167 ***
instant_bookable                            -0.02185543829627   0.01010466138776  -2.163             0.030576 *
calculated_host_listings_count_private_rooms -0.01619268966273  0.00216194607392  -7.490  0.00000000000007580 ***
calculated_host_listings_count_shared_rooms  -0.01768525123586  0.00135636826981 -13.039 < 0.0000000000000002 ***
reviews_per_month                            0.00003663686403   0.00518824903105   0.007             0.994366
num_host_verif                              -0.00484731641408   0.00287018846841  -1.689             0.091285 .
num_amenities                                0.00612911379003   0.00056224080347  10.901 < 0.0000000000000002 ***
is_private_bath                              0.31249053054409   0.01934293364336  16.155 < 0.0000000000000002 ***
room_type2                                   0.33308227090394   0.00690043035614  48.270 < 0.0000000000000002 ***
host_years                                   0.00225659542996   0.00219484861190   1.028             0.303917
last_review_date                             0.02793614425736   0.00559655738594   4.992  0.00000061041223775 ***
region1                                      0.11389292893581   0.01820993766876   6.254  0.00000000041806864 ***
region2                                     -0.10145349837865   0.03432742244314  -2.955             0.003130 **
region3                                      0.19660708038254   0.03987758718088   4.930  0.00000083661393058 ***
region4                                     -0.23332899486336   0.02081208129300 -11.211 < 0.0000000000000002 ***
num_proximity                                0.00002420208398   0.00001037770458   2.332             0.019717 *
local_host                                  -0.02128347879765   0.01013263622465  -2.100             0.035715 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4162 on 8560 degrees of freedom
Multiple R-squared:  0.6473,    Adjusted R-squared:  0.6458
F-statistic: 436.3 on 36 and 8560 DF,  p-value: < 0.00000000000000022
```

- Significant variables (smallest p-value) highlighted

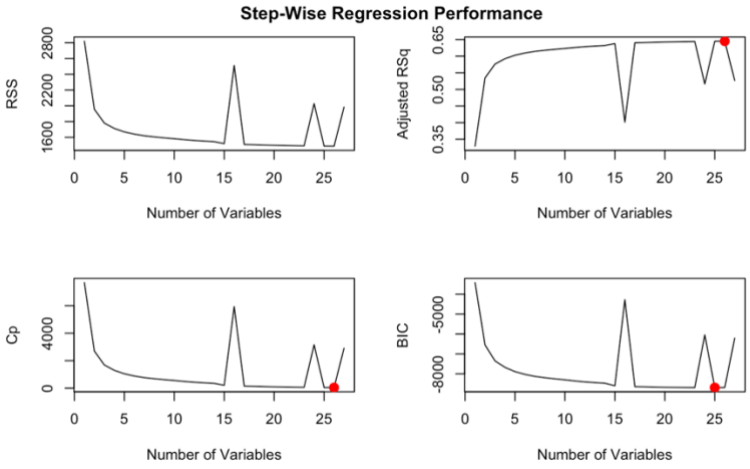
5. Linear Model Residuals with Original Response




6. Linear Model Residuals with Logarithmically Transformed Response




7. Step-wise Linear Regression Model Accuracy

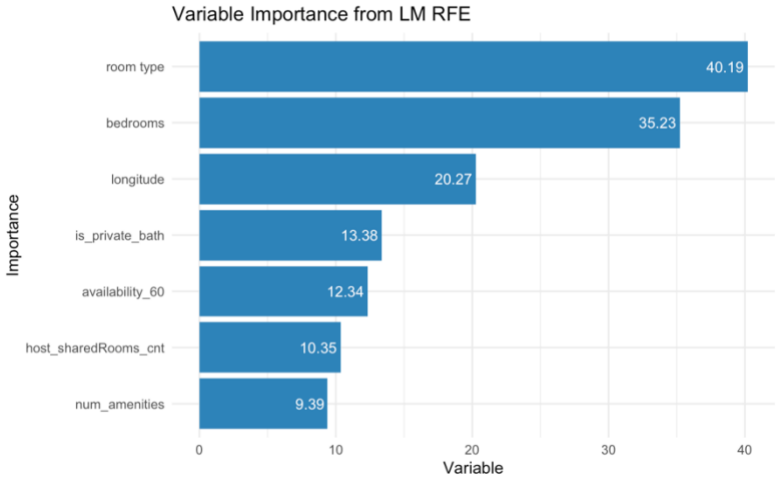


8. Model Summary: Step-wise Linear Regression Model

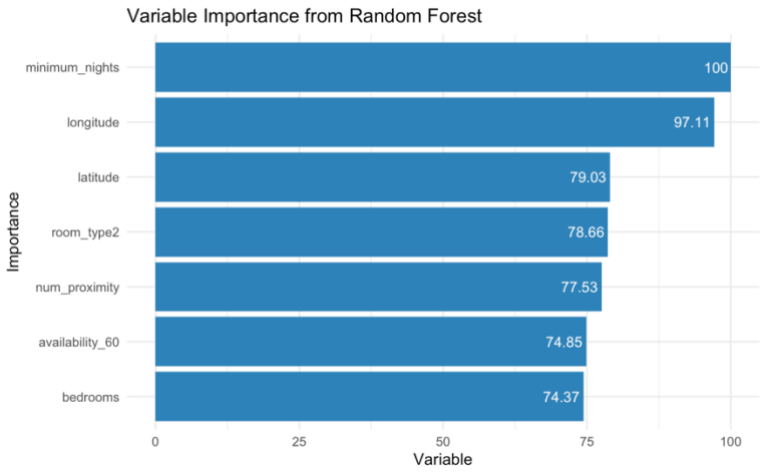
```
Coefficients:
                                          Estimate        Std. Error t value             Pr(>|t|)
(Intercept)                        -100.00000000000000 13.81459778293127  -6.133 0.00000000090187500 ***
host_response_rate                   -0.04969353857099  0.00013579994704  -3.660             0.000254 ***
host_is_superhost                     6.60997515570272  0.01082592336312   5.912 0.00000000350119148 ***
host_identity_verified                4.85697188571277  0.01212466832698   3.912 0.00009238814082114 ***
latitude                            -70.15169113661980  0.16413549223005  -7.366 0.00000000000019195 ***
longitude                           -66.21664211859526  0.10789492991143 -10.058 < 0.0000000000000002 ***
bathrooms                             5.83244113730834  0.00640917712007   8.845 < 0.0000000000000002 ***
bedrooms                             36.51842313218545  0.00583161068826  53.380 < 0.0000000000000002 ***
minimum_nights                       -0.10499045927576  0.00014897444237  -7.051 0.00000000000191069 ***
maximum_nights_avg_ntm                0.00000003141472  0.00000000008788   3.575             0.000353 ***
availability_60                       0.34517925790221  0.00020468880427  16.835 < 0.0000000000000002 ***
number_of_reviews                    -0.03425070847245  0.00006538251836  -5.239 0.00000016492381679 ***
number_of_reviews_l30d                1.25750063798649  0.00313442224491   3.987 0.00006750074083394 ***
review_scores_rating                  0.54569965614151  0.00071759914077   7.584 0.00000000000003705 ***
review_scores_checkin                -5.86634659927308  0.00796287482814  -7.592 0.00000000000003479 ***
review_scores_location                3.55482151049973  0.00809145452770   4.317 0.00001599289030539 ***
calculated_host_listings_count_private_rooms -1.67413006123794  0.00214962378173  -7.854 0.00000000000000452 ***
calculated_host_listings_count_shared_rooms -1.72937743072727  0.00135189376521 -12.904 < 0.0000000000000002 ***
num_amenities                         0.62104239019265  0.00055609563310  11.133 < 0.0000000000000002 ***
is_private_bath                      36.58813503956302  0.01933385092554  16.127 < 0.0000000000000002 ***
room_type2                           39.83518577835014  0.00687210031532  48.791 < 0.0000000000000002 ***
last_review_date                      2.46939179404280  0.00501815068117   4.861 0.00000118783314590 ***
region1                              14.86111433553838  0.01430175857580   9.688 < 0.0000000000000002 ***
region2                              -9.37722561072015  0.03427307195361  -2.873             0.004077 **
region3                              17.62142978594272  0.03684552923301   4.405 0.00001071158434873 ***
region4                             -20.77996444016348  0.02077294144607 -11.214 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4168 on 8571 degrees of freedom
Multiple R-squared:  0.6459,    Adjusted R-squared:  0.6449
F-statistic: 625.4 on 25 and 8571 DF,  p-value: < 0.00000000000000022
```

- Model coefficients have been scaled to reflect the percentage change in the response (rental price) per unit increase
- Significant variables mentioned are highlighted

9. Top Variables Found via Recursive Feature Elimination



10. Top Variables Found via Random Forest



# 202A Final Project Code

Stella Huang

12/14/2020

## Load Libraries

```
library(tidyverse)
library(dplyr)
library(lubridate)
library(bigmemory)
library(biganalytics)
library(bigtabulate)
library(caret)
library(corrplot)
library(anytime)
library(klaR)
library(devtools)
library(xgboost)
library(gbm)
library(leaps)
```

## 1. Data Cleaning

```
# load data
data <- read.csv("full_listings.csv") # there are 30533 listings


#########################
### data cleaning ###
#########################


# remove unnecessary variables
colnames(data)
col_drops <- c("last_scraped", "name", "description","listing_url", "scrape_id", "neighborhood_overview
               "host_name", "host_url", "picture_url", "host_about", "host_thumbnail_url", "host_pictur
               "host_location","host_neighbourhood", "host_has_profile_pic", "neighbourhood_group_clean
               "calendar_last_scraped", "license", "has_availability", "host_response_time", "neighbour

data <- data[ , !(names(data) %in% col_drops)]


# remove listings wit no rating
data <- data[!is.na(data$review_scores_rating),]
```

```r
data


##############################################
### text conversion / feature engineering ###
##############################################


### PRICE PROCESSING
# change price to numeric
data$price <- as.numeric(substr(data$price,2,nchar(data$price)-3))


### HOST INFO
# change percentages to numeric
data$host_response_rate[data$host_response_rate=="N/A"] <- "0%"
data$host_acceptance_rate[data$host_acceptance_rate=="N/A"] <- "0%"
data$host_response_rate <- as.numeric(substr(data$host_response_rate,1,nchar(data$host_response_rate)-1)
data$host_acceptance_rate <- as.numeric(substr(data$host_acceptance_rate,1,nchar(data$host_acceptance_ra
# count number of host verification documents and amenities
data <- data %>% mutate(num_host_verif = str_count(host_verifications, "\'")/2, num_amenities=str_count


### BATHROOM COUNT CONVERSION
# add bathroom num
# get first element, which is the number of bathrooms
data$bathrooms <- unlist(lapply(str_split(data$bathrooms_text, " "), `[[`, 1))
# check which entries are half baths, and replace with 0
data[data$bathrooms_text %in% c("Half-bath", "Private half-bath", "Shared half-bath"),]$bathrooms <- 0.5
# create variable to indicate if private or not (0:no, 1:yes)
data$is_private_bath <- ifelse(str_detect(data$bathrooms_text, "private"), 1, 0)
# convert to numeric
data$bathrooms <- as.numeric(data$bathrooms)


### ROOM TYPE CONVERSION
# check room type
table(data$room_type)
# assign number encodings to room type
room_type2 <- data$room_type
room_type2[room_type2 == "Shared room"] <- 1
room_type2[room_type2 == "Private room"] <- 2
room_type2[room_type2 == "Hotel room"] <- 3
room_type2[room_type2 == "Entire home/apt"] <- 4
data$room_type2 <- as.numeric(room_type2)


### MISSING VALUES
data$bedrooms[is.na(data$bedrooms)] <- 0
data$beds[is.na(data$beds)] <- 0
data$reviews_per_month[is.na(data$reviews_per_month)] <- 0
```

```
### TRUE/FALSE CONVERSION
# find such col
binary_feat <- c("host_is_superhost", "host_identity_verified", "instant_bookable")
# replace with 0/1 and convert to numeric
binary_feat_data <- data[names(data) %in% binary_feat] %>% mutate_all(funs(str_replace(., "t", "1"))
                                                        ) %>% mutate_all(funs(str_replace(., "f", "0"))) %>% m
data[names(data) %in% binary_feat] <- binary_feat_data


### DATE CONVERSION
data %>% select_if(is.character)
date_feat <- c("host_since", "first_review", "last_review")
# convert to date format
for(i in date_feat){
  data[,i] <- as.Date.character(mdy(data[,i]))
}


# calculate durations
data$host_years <- interval(data$host_since,lubridate::ymd("2020-11-23")) / years(1)
data$last_review_date <- interval(data$last_review,lubridate::ymd("2020-11-23")) / years(1)


### MAP NEIGHBORHOOD TO REGION
regions <- read.csv("la_regions.csv")
table(regions$NAME)
# find unmapped neighborhoods and replace names
data[!data$neighbourhood_cleansed %in% regions$NAME,]
data$neighbourhood_cleansed <- str_replace(data$neighbourhood_cleansed, "La Canada Flintridge", "La Cañ
data$neighbourhood_cleansed <- str_replace(data$neighbourhood_cleansed, "East Whittier", "Whittier")
sum(!data$neighbourhood_cleansed %in% regions$NAME)==0
# add regions
data$region <- regions[match(data$neighbourhood_cleansed, regions$NAME),]$REGION
# combine angeles forest w/ antelope valley b/c there's only 9 obs in angeles forest
data$region <- str_replace(data$region,  "Angeles Forest", "Antelope Valley")
table(data$region)
```

## 2. Remove Correlated Features

```
################################
### remove unneeded features ###
################################
var_to_remove <- c("host_verifications", "amenities", "first_review", "last_review", "host_since", "bath
data <- data[!names(data) %in% var_to_remove]


############################
### feature correlation ###
############################
feat_leaveout <- c("id", "host_id", "latitude", "longitude")
# find correlated feats
cor_matrix <- cor((data[!names(data) %in% feat_leaveout] %>% select_if(is.numeric)), use="complete.obs"
```

```r
cor_matrix[upper.tri(cor_matrix)] <- 0 #no symmetry => redundant pairs
diag(cor_matrix) <- 0
# create dataframe of cor var
cor_df <- as.data.frame(as.table(cor_matrix))
# filter with 0.75+ correlation
correlated_var <- cor_df %>% filter(Freq >= 0.75) %>% filter(Var1!=Var2)%>% mutate_if(is.factor, as.cha
# feat_to_keep <- cor_df %>% filter(Freq >= 0.75) %>% filter(Var1!=Var2) %>% select(Var2) %>% pull()

# selected some feat to keep
correlated_var
feat_to_keep <- c("review_scores_rating", "host_listings_count", "bedrooms", "beds",
                  "minimum_nights", "maximum_nights_avg_ntm", "availability_60",
                  "review_scores_rating", "reviews_per_month")
corfeat_to_remove <- unique(c(correlated_var$Var1, correlated_var$Var2)[!c(correlated_var$Var1, correla
# remove correlated var
corfeat_to_remove
data <- data[!names(data) %in% corfeat_to_remove]
```

### 2.5 Count listings nearby

```r
# use complete data only
data <- subset(data, select = -c(neighbourhood_cleansed, property_type, room_type)) # remove more var
data <- data[complete.cases(data),]


## DISTANCE BASED METRIC
# wanna find the number of listings within 2 mile radius
library(geosphere)
dt.haversine <- function(lat_from, lon_from, lat_to, lon_to, r = 6378137){
    radians <- pi/180
    lat_to <- lat_to * radians
    lat_from <- lat_from * radians
    lon_to <- lon_to * radians
    lon_from <- lon_from * radians
    dLat <- (lat_to - lat_from)
    dLon <- (lon_to - lon_from)
    a <- (sin(dLat/2)^2) + (cos(lat_from) * cos(lat_to)) * (sin(dLon/2)^2)
    return(2 * atan2(sqrt(a), sqrt(1 - a)) * r)
}

num_proximity <- c()

for(i in c(1:nrow(data))){
# for(i in c(1:10)){
  samp1 <- as.matrix(data[i,c("longitude", "latitude")])
  other_samp <- as.matrix(data[-i,c("longitude", "latitude")])
  pair_distances <- distHaversine(samp1, other_samp)/1609
  # keep samples within 3 miles
  close_listings <- sum(pair_distances<=3)
  num_proximity[i] <- close_listings

}
```

```r
summary(num_proximity)
hist(num_proximity, breaks=100)
data$num_proximity <- num_proximity

# ## 2D KERNEL REGRESSION
library(splancs)
library(maps)
latlon_bw <- sqrt(bw.nrd0(data$longitude)^2+bw.nrd0(data$latitude)^2)
latlon <- as.points(data$longitude, data$latitude)


boundary <- matrix(c(range(data$longitude)[1]-0.1,range(data$latitude)[1]-0.1,
                     range(data$longitude)[2]+0.1,range(data$latitude)[1]-0.1,
                     range(data$longitude)[2]+0.1,range(data$latitude)[2]+0.1,
                     range(data$longitude)[1]-0.1,range(data$latitude)[2]+0.1,
                     range(data$longitude)[1]-0.1,range(data$latitude)[1]-0.1),ncol=2,byrow=T)

latlon_dens <- kernel2d(latlon, boundary, latlon_bw)
par(mfrow=c(1,2))
image(latlon_dens, col=gray((64:20)/64),xlab="Latitude",ylab="Longitude", main="2D Kernel Smoothing of
points(latlon, col='red')
map('county', 'California', add=T)
```

## 3. Exploratory Analysis

```r
data <- read.csv("cleaned_data.csv")
full_data <- read.csv("full_listings.csv", stringsAsFactors = FALSE)
full_data <- full_data[match(data$id,full_data$id),]
data$local_host <- ifelse(str_detect(full_data$host_location, "Los Angeles"), 1, 0)

# convert char to factor
data[sapply(data, is.character)] <- lapply(data[sapply(data, is.character)], as.factor)
data <- model.frame(price~., data=data, drop.unused.levels = TRUE)
# filter region
data <- data %>% filter(region %in% c("Santa Monica Mountains", "Westside", "Central L.A.", "South L.A.

data$num_amenities

ggplot(aes(x=as.factor(data$room_type2), y=log(price), group=as.factor(data$room_type2)), data=data) +
  geom_boxplot(aes(fill = as.factor(data$room_type2))) + facet_grid(.~region) + theme_light() +
  ggtitle("Room Type vs Price") + xlab("room type") +
  scale_fill_discrete(name = "Room Type", labels = c("Shared","Private","Hotel", "Entire apt/home"))

ggplot(aes(x=price), data=data[data$price<600,]) + geom_histogram(bins=100) + facet_grid(.~region) + th

histogram(log(data$price))
histogram(data$price[data$price<700])
```

### 3.5. Kernel Regression

## C Code

```c
# include <R.h>
# include <Rmath.h>
void kernel_reg(int *n, double *x, double *y, double *b, int* m,
    double *g2, double *res2)
{
    int i,j;
    double c;
    double est_sum_y;
    double est_sum;


    // loop through each grid index
    for(i=0; i<*m; i++){
        est_sum = 0.0;
        est_sum_y = 0.0;

        // calculate density; loop through each data index
        for(j=0; j<*n; j++){
            c = dnorm(x[j]-g2[i], 0, *b, 0)/ *n; //compute kernel at x on grid
            est_sum += c; // denominator
            est_sum_y += y[j] * c; //multiply by y and add; numerator
        }

        res2[i] = est_sum_y / est_sum;
    }
}
```

```r
# let's examine kernel regression density of price vs longitude
system("R CMD SHLIB kernel.c")
dyn.load("kernel.so")

# declare var
x <- data$longitude
y <- log(data$price)
xgrid <- c(seq(from = range(x)[1], to = range(x)[2], length.out=300))
bw <- bw.nrd(x)


# load function
gaussian_kernel <- function(x, y, bw, xgrid){
  n <- length(x)
  m <- length(xgrid)
  a=.C("kernel_reg", as.integer(n), as.double(x), as.double(y),
       as.double(bw), as.integer(m), as.double(xgrid), y=double(m))
  a$y
}

kernel_est <- gaussian_kernel(x,y,bw,xgrid)
plot(kernel_est~xgrid, type='l')
kernel_est
```

```r
# resampling to get estimate
est_matrix <- c()
for(i in c(1:200)){
  new_samples <- sample_n(data, 2000, replace=TRUE)
  new_y <- as.vector(log(new_samples$price))
  new_x <- as.vector(new_samples$longitude)

  new_kernel_est <- gaussian_kernel(new_x, new_y, bw, xgrid)
  est_matrix <- cbind(est_matrix, new_kernel_est)

}

# extract 5th largest and 195th largest: order each row -> get 5th & 195th col of matrix
dim(est_matrix)
ci95 <- t(apply(est_matrix, 1, sort))[, c(10,195)]
colnames(ci95) <-c("lower", "upper")

kernelest_df <- as.data.frame(cbind(xgrid, kernel_est, ci95))

# plot KDE with 95% CI
ggplot(aes(x=xgrid, y=kernel_est), data=kernelest_df) + geom_point() +
  geom_line() + geom_ribbon(data=kernelest_df,aes(ymin=lower,ymax=upper),alpha=0.3) +
  xlab("Longitude") + ylab("Estimated log(Price)") + ggtitle("KDE Price with 95% CI Band") + theme_mini
```

## 4. Train test split

```r
# write.csv(data,"cleaned_data.csv")
data <- read.csv("cleaned_data.csv")
full_data <- read.csv("full_listings.csv", stringsAsFactors = FALSE)
full_data <- full_data[match(data$id,full_data$id),]
data$local_host <- ifelse(str_detect(full_data$host_location, "Los Angeles"), 1, 0)

# convert char to factor
data[sapply(data, is.character)] <- lapply(data[sapply(data, is.character)], as.factor)
data <- model.frame(price~., data=data, drop.unused.levels = TRUE)
# filter region
data <- data %>% filter(region %in% c("Santa Monica Mountains", "Westside", "Central L.A.", "South L.A.

# train test split
train_index <- createDataPartition(data$price, p = .7, list = FALSE, times = 1)
train_data <- data[train_index,]
train_y <- train_data$price
train_data <- subset(train_data, select = -c(price, id, host_id))

test_data <- data[-train_index,]
test_y <- test_data$price
test_data <- subset(test_data, select = -c(price, id, host_id))
```

### Function to calculated adjusted r^2

```
adj_r2 <- function(r2, n, p){
  adj = 1-((1-r2)*(n-1)/(n-p-1))
  return (adj)
}
```

## 5. Linear Model

```
################################
### BUILDING LINEAR MODEL ######
################################
# drop unused levels in train data
train_data <- model.frame(train_price~., data=train_data, drop.unused.levels = TRUE)
linearmodel <- lm(train_y~., data=train_data) # original var
linearmodel <- lm(log(train_y)~., data=train_data) # transformed var
summary(linearmodel)
par(mfrow=c(2,2))
plot(linearmodel)
mtext(text=expression(bold("Residual Plot for Linear Model, Log(y)")), side = 3, line = -14, outer = TRU

# bc <- boxcox(linearmodel)
# lam1 <- bc$x[which.max(bc$y)]
# linearmodel <- lm(((train_price^lam1) - 1) / lam1~., data=train_data)

lm_predictions <- predict(linearmodel, test_data)
postResample(pred = lm_predictions, obs = log(test_y))


# USE CV
fit.control <- trainControl(method = "repeatedcv", number = 5, repeats = 3)
set.seed(123)
# train linear model with CV
lm_caret <- train(log(train_y)~.,train_data, method = "lm", trControl = fit.control)
summary(lm_caret)
lm_predictions <- predict(lm_caret, test_data)
# calculate prediction accuracy
postResample(pred = lm_predictions, obs = log(test_y))
adj_r2(postResample(pred = lm_predictions, obs = log(test_y))[2], nrow(test_data),36)
```

## 6. Step-wise regression

```
# step wise selection with leaps library
sw_linearmod <- regsubsets(log(train_y)~., data=train_data, nvmax = 27, method = "seqrep")
reg_summary <- summary(sw_linearmod)
par(mfrow = c(2,2))
plot(reg_summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
plot(reg_summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")

# We will now plot a red dot to indicate the model with the largest adjusted R^2 statistic.
```

```r
# The which.max() function can be used to identify the location of the maximum point of a vector
adj_r2_max = which.max(reg_summary$adjr2) # 11

# The points() command works like the plot() command, except that it puts points
# on a plot that has already been created instead of creating a new plot
points(adj_r2_max, reg_summary$adjr2[adj_r2_max], col ="red", cex = 2, pch = 20)

# We'll do the same for C_p and BIC, this time looking for the models with the SMALLEST statistic
plot(reg_summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
cp_min = which.min(reg_summary$cp) # 10
points(cp_min, reg_summary$cp[cp_min], col = "red", cex = 2, pch = 20)

plot(reg_summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
bic_min = which.min(reg_summary$bic) # 6
points(bic_min, reg_summary$bic[bic_min], col = "red", cex = 2, pch = 20)
mtext(text=expression(bold("Step-Wise Regression Performance")), side = 3, line = -1.5, outer = TRUE)

# choose best number of var to include
# find such coefficients
sw_bestvar <- names(coef(sw_linearmod, 23))


fit.control <- trainControl(method = "repeatedcv", number = 5, repeats = 3)

set.seed(123)
# fit model with best subset selected
lm_caret <- train(log(train_y)~.,train_data[,names(train_data) %in% c(sw_bestvar, "region")],
                  method = "lm", trControl = fit.control)
lm_summary <- as.data.frame((exp(coef(summary(lm_caret)))-1)*100)
options(scipen=999)
lm_summary[order(lm_summary$Estimate),]

# calculate accuracy
lm_predictions <- predict(lm_caret, test_data)
postResample(pred = lm_predictions, obs = log(test_y))
adj_r2(postResample(pred = lm_predictions, obs = log(test_y))[2], nrow(test_data),36)
```

## 7. Recursive Feature Elimination

```r
################################
# ### RFE FOR LINEAR MODEL #######
# ################################
ctrl <- rfeControl(functions = lmFuncs,
                   method = "repeatedcv",
                   repeats = 5,
                   number = 3,
                   verbose = FALSE)


lmProfile <- rfe(x=train_data%>%select_if(is.numeric), y=log(train_y), sizes = c(1:20), rfeControl = ctr

# rank of variables
```

```r
lmProfile$variables
best_rfe_lm <- lmProfile$variables %>% filter(Variables==32)

# sapply(best_rfe_lm, mean)

best_rfe_lm_var <- best_rfe_lm %>% filter(var %in% c("bedrooms", "room_type2", "longitude", "is_private_

best_rfe_lm_var$var <- str_replace(best_rfe_lm_var$var,
                                   "calculated_host_listings_count_shared_rooms","host_sharedRo
                                   oms_cnt")
best_rfe_lm_var$var <- str_replace(best_rfe_lm_var$var,"room_type2","room type")
ggplot(aes(x=reorder(var, Overall), y=Overall), data=best_rfe_lm_var) + geom_bar(stat="identity", fill="
+ coord_flip() + theme_minimal() + geom_text(aes(label=round(Overall,2)), hjust=1.1, size=3.5, col="whi
+ ggtitle("Variable Importance from LM RFE") + xlab("Importance") + ylab("Variable")
```

## 8. Random Forest

```r
#################################
### RANDOM FOREST REGRESSION ###
#################################
library(randomForest)
library(e1071)

mtry <- round(ncol(train_data)/3)
tunegrid <- expand.grid(.mtry=mtry)
train.control <- trainControl(method = "repeatedcv", number = 5, repeats=3)

# Train the model
rf_model <- train(log(train_y)~., data=train_data,
                  method = "rf", tuneGrid=tunegrid,
                  trControl = train.control, importance=T)

# RESULTS
print(rf_model)
#################################
# 8597 samples
#   32 predictor
#
# No pre-processing
# Resampling: Cross-Validated (5 fold, repeated 3 times)
# Summary of sample sizes: 6877, 6877, 6877, 6878, 6879, 6878, ...
# Resampling results
#
#   RMSE       Rsquared   RMSE SD      Rsquared SD
#   0.3498099  0.7528557  0.007383131  0.01263049
#
# Tuning parameter 'mtry' was held constant at a value of 11
#################################


rf_model$finalModel #output below
#################################
```

```r
# Call:
#  randomForest(x = x, y = y, mtry = param$mtry, importance = ..1)
#                Type of random forest: regression
#                      Number of trees: 500
# No. of variables tried at each split: 11
#
#           Mean of squared residuals: 0.1196127
#                     % Var explained: 75.54
#################################

# make prediction
rf_predictions <- predict(rf_model, test_data, type="raw")
postResample(pred = rf_predictions, obs = log(test_y))
adj_r2(postResample(pred = rf_predictions, obs = log(test_y))[2], nrow(train_data), 32)


# VARIABLE IMPORTANCE
write.csv(varImp(rf_model, scale = TRUE)$importance, "price_rfImp.csv")
rf_imp <- read.csv("price_rfImp.csv")

# plot variable importance
ggplot(aes(x=Overall), data=varImp(rf_model, scale = TRUE)) + geom_bar(stat="identity", fill="steelblue
  coord_flip() + theme_minimal() + geom_text(aes(label=round(Importance,2)), hjust=1.1, size=3.5, col="
  ggtitle("Variable Importance from Random Forest") + xlab("Importance") + ylab("Variable")

ggplot(aes(x=reorder(X, Overall), y=Overall), data=(rf_imp %>% arrange(desc(Overall)))[1:7,]) +
  geom_bar(stat="identity", fill="steelblue") + coord_flip() + theme_minimal() +
  geom_text(aes(label=round(Overall,2)), hjust=1.1, size=3.5, col="white") + ggtitle("Variable Importan
  xlab("Importance") + ylab("Variable")
```