Predicting Airbnb Prices in New York

Vaishak Naik

November 30, 2021

Abstract

Airbnb is a platform that allows homeowners to list their houses online and charge guests to stay in them. Typically homeowners determine the price. Despite the fact that Airbnb and other sites offer some general advice, there are presently no free and accurate programs that assist hosts in pricing their properties using a variety of data factors. In this project, using machine learning techniques, we will try to predict and assist homeowners in determining the Daily rates of their property in New York. Price will be a dependant variable, taking into account numerous independent variables such as the neighborhood, number of bedrooms, security deposit, etc. Because we have a label and it is continuous, this is a supervised learning regression problem.

1 Abstract

Airbnb is a platform that allows homeowners to list their houses online and charge guests to stay in them. Typically homeowners determine the price. Despite the fact that Airbnb and other sites offer some general advice, there are presently no free and accurate programs that assist hosts in pricing their properties using a variety of data factors.

In this project, using machine learning techniques, we will try to predict and assist homeowners in determining the Daily rates of their property in New York. Price will be a dependant variable, taking into account numerous independent variables such as the neighborhood, number of bedrooms, security deposit, etc. Because we have a label and it is continuous, this is a supervised learning regression problem.

2 Initial exploration

The dataset used for this project is scraped Airbnb listings. The dataset consists of information on New York Airbnb. After initial data exploration, we can see that the analysis data consists of 91 variables with 41330 rows of data. The data set also consisted of 263294 empty values and 48562 NA values. There were different types of NA values, and these were converted into NA standard form.

A descriptive analysis was carried out to understand the relationship between a price and its 91 variables in the dataset. The variables can be categorized into four main groups: Property and Location Information, Host Information, All the Reviews, and Others.

Some of the Data Wrangling techniques used in this project are: 1) Using Regular Expression to Clean and Standardize Data 2) Checking and Converting to Standard NA Format. 3) Median Imputation 4) Data Imputation using quantiles 5) Date Manipulation, Using Only Month and Year. 6) Low-Frequency Data Grouping Using Forcats. 7) Data type Conversion.

```
## For Complete Data Wrangling Code Please refer datawranglingVariableSelection.Rmd File

## Imported Library
library(caret)
library(dplyr)

## Importing Data
data <- read.csv('analysisData.csv')
scoringData <- read.csv('scoringData.csv')

## Initial Data Exploration
dim(data)
dim(scoringData)
## Price Distribution
hist(data$price)

## Combining Analysis and Scoring Data for Cleaning the Data
combinedData <- bind_rows(data, scoringData)

### Checking and Converting to Standard NA Format
(sum(combinedData=="", na.rm=TRUE))
(sum(combinedData=="N/A", na.rm=TRUE))
(sum(combinedData=="NA", na.rm=TRUE))
```

3 Models and Feature Selection

Based on the domain exploration, I found 42 out of 91 variables relevant for this project. I used Shrinkage: Lasso Regression method as my feature selection technique. Since the shrinkage penalty of lasso forces some coefficient estimates to be exactly zero, it can be used as a feature selection method.

I started off with linear regression, using the features selected from the Lasso Regression method. The initial submission on the public dashboard yielded an RMSE score of 78.16944, but the model was iteratively improved using polynomial function and variable interaction with the RMSE score of 71.88501. The second model used in the price prediction was Random Forest using Tuned Forest Ranger with Variables selected from Lasso Regression which yielded the RMSE score of 66.08891 on the public dashboard. The random forest model's time-consuming nature reduced its efficiency significantly. Lastly, I applied Boosting Using XGBoost with Variables selected from Lasso Regression. The use of XGBoost model greatly improved the accuracy of the prediction results and successfully lowered the model's RMSE to roughly 51 on test data. However, it eventually ran into the same problem of being unable to increase its performance and even experiencing some over-fitting issues. I eventually submitted this as my final RMSE score with 65.93 on the public dashboard and 61.99 on the private dashboard.

```
# Variables Shortlist Based on Domain Knowledge
combinedData = combinedData[, c('id','neighbourhood_cleansed', 'property_type',
'room_type', 'accommodates', 'bathrooms', 'bedrooms', 'price', 'cleaning_fee',
'availability_30', 'availability_60', 'number_of_reviews', 'review_scores_rating',
'reviews_per_month', 'zipcode', 'city',
'cancellation_policy', 'host_acceptance_rate', 'host_is_superhost', 'host_response_rate',
'host_response_time',
'neighbourhood_group_cleansed', 'host_listings_count', 'availability_90',
'extra_people', 'security_deposit', 'minimum_nights',
'maximum_nights', 'review_scores_value', 'review_scores_location',
'review_scores_communication', 'review_scores_checkin',
'review_scores_cleanliness', 'review_scores_accuracy', 'reviews_per_month',
'instant_bookable', 'host_listings_count', 'host_since',
'host_verifications', 'host_identity_verified', 'host_since_year',
'guests_included','require_guest_phone_verification',
'calculated_host_listings_count_private_rooms' )]


##Feature Selection: Using Lasso Regression
dataForVariableSelection <- combinedData[!is.na(combinedData$price),]
x = model.matrix(price~.-1, data = dataForVariableSelection)
y = dataForVariableSelection$price
cv.lasso = cv.glmnet(x,y,alpha=1)
coef(cv.lasso)

#Variables Shortlisted After Lasso Regression

variableNew <- c("id","price","host_listings_count" ,"accommodates" ,
"availability_90" ,"extra_people" ,"number_of_reviews" ,"bathrooms" ,"security_deposit"
,"room_type" ,"property_type" ,"host_acceptance_rate" ,"zipcode" ,"cleaning_fee"
,"bedrooms" ,"host_response_rate" ,"host_is_superhost"
,"availability_30" ,"minimum_nights" ,"maximum_nights"
,"cancellation_policy" ,"review_scores_value" ,"review_scores_location"
,"review_scores_communication" ,"review_scores_checkin"
,"review_scores_cleanliness" ,"review_scores_accuracy"
,"instant_bookable" ,"city" ,"host_since_year" ,"guests_included"
,"neighbourhood_cleansed" ,"require_guest_phone_verification")

combinedDataModified = combinedData[, variableNew]

##Modeling Technique

#Linear Regression Model with Varibles selected from Lasso Regression
linearModel = lm(price ~ host_listings_count + poly(accommodates, 2) + availability_90
+ extra_people + number_of_reviews + bathrooms + security_deposit
+ room_type + property_type + host_acceptance_rate + zipcode + cleaning_fee + bedrooms
+ host_response_rate + host_is_superhost + availability_30 \
+ minimum_nights + maximum_nights + cancellation_policy + review_scores_value
+ review_scores_location + review_scores_communication
+ review_scores_checkin + review_scores_cleanliness + review_scores_accuracy
+ reviews_per_month + instant_bookable + city + host_since_year
+ guests_included + neighbourhood_cleansed + require_guest_phone_verification
+ neighbourhood_cleansed:accommodates
+ neighbourhood_cleansed:cleaning_fee, data = airbnb)

coef_value = sort(coef(linearModel)

#Random Forest Using Tuned Forest Ranger with Variables selected from Lasso Regression
trControl=trainControl(method="cv",number=10)
mtryVaribale = sort(c(100,150,250,550,750,827))
tuneGrid = expand.grid(mtry=mtryVaribale,
                       splitrule="extratrees",
                       min.node.size = c(50,100,150))
```

```
set.seed(617)
cvModel = train(price ~ .,
                data=airbnb,
                method="ranger",
                num.trees=1500,
                trControl=trControl,
                tuneGrid=tuneGrid)

forest = ranger(price ~ .,
                data=airbnb,
                num.trees = 1500,
                mtry=cvModel$bestTune$mtry,
                min.node.size = cvModel$bestTune$min.node.size,
                splitrule = cvModel$bestTune$splitrule
                )

#Boosting Using XGBoost with Variables selected from Lasso Regression
xgboost = xgboost(data=as.matrix(trainMatrix),
                  label = train$price,
                  nrounds=5000,
                  verbose = 0,
                  subsample = 0.5,
                  eta = 0.1,
                  early_stopping_rounds = 150
                  )
xgboost$best_iteration
```

4 Model Comparison

The final submitted model was Boosting Using XGBoost with Variables selected from Lasso Regression with 61.98775 on Private dashboard, but best performing model on the private dashboard was Random Forest using Tuned Forest Ranger with Variables selected from Lasso Regression with the RMSE of 60.58088.

Model from previous section	RMSE on test data	RMSE on Public Dashboard	RMSE on Private Dashbard	Other notes
Model 1: Linear Regression	64.73594	71.62483	65.74975	
Model 2: CV-tuned Random forest	54.41611	66.08891	60.58088	
Model 3: XGBoost	51.14096	65.93043	61.98775	Final Kaggle RMSE: 61.98775

5 Discussion

The final model is based on the XGBoost Algorithm and contains 28 variables before dummy encoding. To examine the influence of each predictor in a more specific and accurate way, xgb.importance() method was employed to generate the following bar chart.

Some of the most important features are:

1) How many people the property accommodates.
2) The security deposit.
3) How many days are available to book out of the next 30 days.
4) Cleaning Fee.
5) Review Scores.
6) If the property is in Soho.
7) Host Since.
8) Zip code.

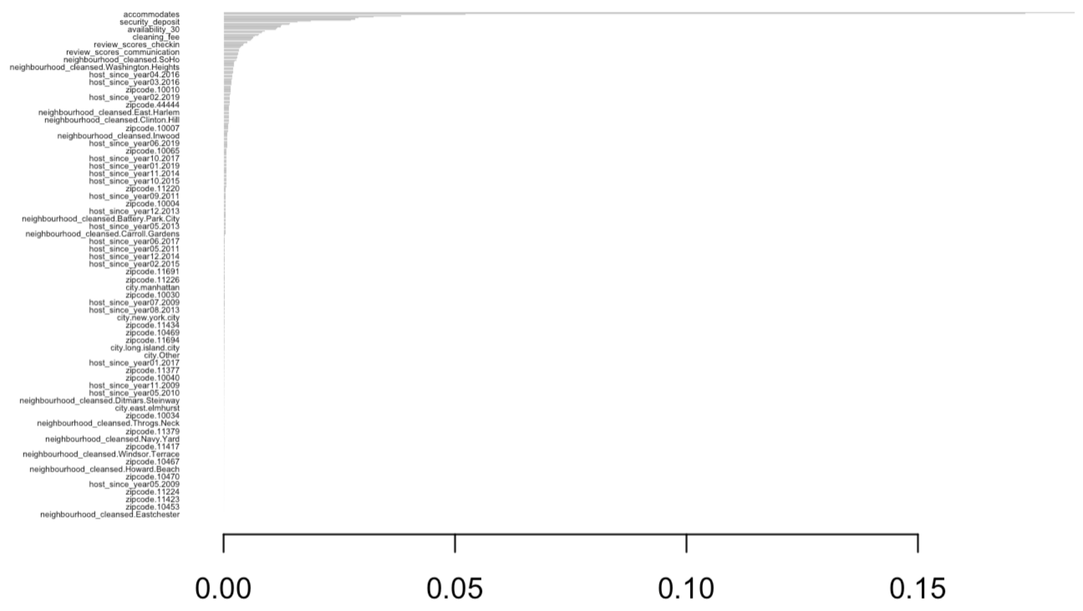

Figure 1: Feature Importance

6 Future directions

If I had more time to improve this model, I would have incorporated data extraction from the descriptive variables. And I would have tried to gather more domain knowledge which would have improved some of the decisions regarding data imputation.