Christina (Zhixiao) Zhang
December 3,2018
AA Frameworks and Methods
Professor François Scharffe

Kaggle Report

Introduction

The Kaggle competition on predicting Airbnb pricing provides us a great opportunity to work on a complete data project independently from beginning to the end. My project spans from data exploring, processing, modelling to evaluation, and each step are highly dependent on the performance of the other steps. The goal of my project is to build a model that predicts the pricing of Airbnb listing in New York. The key challenge for me during the process is data cleaning, model choice, and feature selection. In the following sections, I will provide more step-to-step details of my project.

Exploring the Data

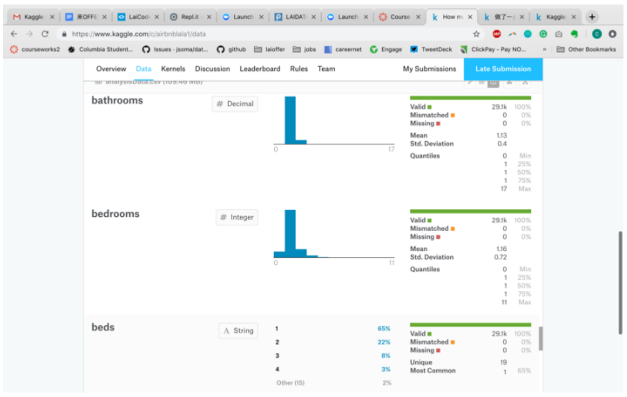
The first step of my project, which is the most important one to me, is to understand the data set. With such a huge and overwhelming dataset, it is important to know:

- What does each column represent? What are the different features?
- What is the data type of the columns?
- What is the data distribution like?
- How much of the data are valid (not n/a)?

To answer these questions, I first use summary(data) to get a general idea of the whole data, then I used table(data$featurename) to take a closer look at the data distribution of some columns that I am interested. For instance, by using summary(), I realized that there are many columns representing the geographical location of the listings. In order to know what's the difference of these geographical location columns, for instance, I used the following to see the data distribution of the three neighborhood features and then decide to use 'neighbourhood_group_cleansed' as my final neighborhood feature since the other two are too dirty (with too many variations and observations) to be used.

```r
summary(data)
table(data$neighbourhood)
table(data$neighbourhood_cleansed)
table(data$neighbourhood_group_cleansed)
```

In order to speed up my data exploration process, I used data section on Kaggle's competition page. As shown below, Kaggle data automatically showed users the column name, data type, and data distribution. This is very helpful as a reference for me in my exploration phase, and with the help of this function, I am able to quickly identify and remove useless columns (with most of the data shown as n/a).



https://www.kaggle.com/c/airbnblala1/data

bathrooms — # Decimal

```
Valid         29.1k  100%
Mismatched    0      0%
Missing       0      0%
Mean          1.13
Std. Deviation 0.4
Quantiles     0      Min
              1      25%
              1      50%
              1      75%
              17     Max
```

bedrooms — # Integer

```
Valid         29.1k  100%
Mismatched    0      0%
Missing       0      0%
Mean          1.16
Std. Deviation 0.72
Quantiles     0      Min
              1      25%
              1      50%
              1      75%
              11     Max
```

beds — A String

```
1           65%
2           22%
3           8%
4           3%
Other (15)  2%

Valid         29.1k  100%
Mismatched    0      0%
Missing       0      0%
Unique        19
Most Common   1      65%
```

Preparing the Data for Analysis:

Converting categorical variables to dummy variables

The preparation of data varies with the model that I chose. The very first model that I used is a linear regression model. Since linear regression model only takes numerical variables, I have to convert some categorical variables into something the linear model can digest. The method that I used is to turn categorical variables, such as 'neighborhood_group_cleansed' into dummy variables, by using the codes below:

```r
#Create dummy variables of neighbourhood_group_cleansed
---------------------------------------------------------------------------
  install.packages('dummies')
library(dummies)
data=cbind(data,dummy(data$neighbourhood_group_cleansed,sep = "_"))
str(data)
```

By doing so, I added some additional columns of neighborhood names indicating the neighborhood group (for instance neighbourhood_group_cleansed_Brooklyn with 0=No and 1=Yes). By including the dummy variable columns into my linear model, I got a lightly lower RMSE. However, I encountered an error when applying my model to the test data and I realized that I have to convert the 'neighbourhood_group_cleansed' columns in my test dataset to dummy variables as well.

```r
scoringData=cbind(scoringData,dummy(scoringData$neighbourhood_group_cleansed,
sep = "_"))
```

Normalizing numeric data

Another data processing that I tried is to normalize numeric columns which has different scales. For instance, review_scores_rating is on a scale from 0 to 10, and number_of_reviews has a greater dispersion (with minimum value of 1 and maximum value of 521). By using scale()function, I normalized the following numeric columns and added to my linear regression mode:

```r
#Normalize Data
---------------------------------------------------------------------------
data = mutate(data, normal_review_scores = scale(data$review_scores_rating))
data = cbind(data, normal_accommodates = scale(data$accommodates))
data = cbind(data, normal_bedrooms = scale(data$bedrooms))
data = cbind(data, normal_bathrooms = scale(data$bathrooms))
data = cbind(data, normal_number_of_reviews = scale(data$number_of_reviews))
summary(data$number_of_reviews)
```

Modeling Techniques

Feature Selection: Manual Selection with forward stepwise tuning

A thorough understanding of the original dataset in the data exploring phase is crucial for me to select many of the features based on gut feeling. Based on my user experience of Airbnb and my understanding of the dataset, I included the following categories of data in my dataset: rating, accomodates, location, availability, room and facility, and host. I then further tuned my model using subset feature selection techniques. For instance, I used forward stepwise technique to successfully remove the availability_90 columns in my second linear model:

```r
#forward stepwise
start_mod = lm(price~1,data=data)
empty_mod = lm(price~1,data=data)
full_mod =
lm(price~accommodates+neighbourhood_group_cleansed+number_of_reviews+bathroom
s+bedrooms+availability_90+availability_365+availability_30+number_of_reviews
+guests_included+cancellation_policy+room_type,data=data)
forwardStepwise =
step(start_mod,scope=list(upper=full_mod,lower=empty_mod),direction='forward'
)
summary(forwardStepwise)
```

Linear Regression

I tried linear regression first by including only numeric columns and some additional dummy variable columns. However, the model performance didn't improve a lot after I've reached an RMSE around 75.

Trees and Random Forest

After encountering the limitation of linear regression, I tried tree models. To my surprise, with the same features include, the tree model improved my RMSE to 70. Then, I include more categorical features and used a random forest model with ntree = 100. One thing I didn't do very well is to tune the model in a systematic way. I tuned the model and changed different parameters by trial-and-error. For instance, I first run a random forest with ntree=100, and then with ntree=1000. The model only improved slightly.

Boosting

After trying random forest model, I decided to try some advanced tree models such as 10-fold cross validation and boosting. For some reason, I was not able to run the cross validation on my computer but was able to run the gbm model on my features. However, the result was not so satisfying. I got an RMSE that is around 104, which is much higher than my previous random forest model. I haven't figured why so far, but I think it might be an issue of my choice of boosting model and parameter setting.

Results:

Commit	RMSE	Model
1st	104.439	Liner
2nd	79.7599	Liner
3rd	74.3585	Liner
4th	72.2148	Liner
5th	68.3485	Tree
6th	62.349	Random Forest(ntree = 100)
7th	56.5868	Random Forest(ntree = 1000)
8th	55.4266	Random Forest(ntree = 1000)

Below is my final model with random forest:

```
forest2 =
randomForest(price~accommodates+review_scores_rating+latitude+longitude+neigh
bourhood_group_cleansed+availability_365+availability_30+guests_included+numb
er_of_reviews+room_type+host_is_superhost+host_response_time+extra_people+min
imum_nights+bedrooms+bathrooms+review_scores_cleanliness+review_scores_checki
n+review_scores_communication+review_scores_value+review_scores_location+is_b
usiness_travel_ready+cancellation_policy,data,ntree = 1000)
predForest2 = predict(forest2)
rmse_f2 = sqrt(mean((predForest2-data$price)^2))
rmse_f2
```

Discussion and Reflection

I really enjoyed my first complete and independent R project on predicting Airbnb listing prices. During the process, I struggled through many steps, tried different models and techniques we learnt in class, failed a lot of times, and got a somewhat satisfying prediction model in the end. Below are some very valuable takeaways throughout the project:

- The most valuable thing I've done for this project is to spend a whole afternoon going over each column of the original dataset with the help of R and Kaggle Data section. This laid a solid foundation for my data processing and modelling phase later and also improved my manual feature selection.
- I didn't fully utilize all the feature selection methods we've learnt in class, and if I have more time, I would've tried more methods such as hybrid subset selection and shrinkage methods such as Lasso.
- I failed my boosting model with a rocket high RMSE. I should've tried some different boosting models in the future if I have more time. I also want to learn more about parameter tuning.

- Patience is important. Running a random forest tree model and boosting models took more than 4 hours and at first, I had to force quit a lot of time since I thought there was something wrong.

Appendix: R Script

Kaggle Project

```r
setwd('/Users/christina/Documents/Applied_Analytics/Frameworks/all')

# ensure analysisData.csv and scoringData.csv are in your working directory
install.packages('tidyverse')
library(dplyr)
library(tidyverse)


#data exploring
------------------------------------------------------------------------------
------
  data = read.csv('analysisData.csv')
summary(data)
table(data$neighbourhood)
table(data$neighbourhood_cleansed)
table(data$neighbourhood_group_cleansed)
summary(data$bedrooms)
summary(data$bathrooms)
summary(data$neighbourhood_cleansed)
summary(data$neighbourhood_group_cleansed)
summary(data$bed_type)
summary(data$accommodates)
table(data$bed_type)
table(data$accommodates)

#Remove columns that are not useful by going over the data table under Kaggle
 data section
data <- select(data, -listing_url,-scrape_id,-last_scraped,-name,-summary,-sp
ace,-description,-experiences_offered,-neighborhood_overview,-notes,-transit,
-access,-interaction,-house_rules,-thumbnail_url,-medium_url,-xl_picture_url,
-host_id,-host_url,-host_name,-host_since,-host_about,-jurisdiction_names,-pi
cture_url,-host_acceptance_rate,-host_thumbnail_url,-host_picture_url,-amenit
ies,-first_review,-last_review,-requires_license,-license,-country,-country_c
ode,-has_availability,-calendar_last_scraped)

#Linear Regression
------------------------------------------------------------------------------
------
  #Trial model
  model = lm(price~minimum_nights+review_scores_rating,data)

# read in scoring data and apply model to generate predictions
scoringData = read.csv('scoringData.csv')
pred = predict(model,newdata=scoringData)
```

```r
# construct submision from predictions
submissionFile = data.frame(id = scoringData$id, price = pred)
write.csv(submissionFile, 'sample_submission.csv',row.names = F)

#linear regression (mannual feature selection)
model1 = lm(price~accommodates+review_scores_rating, data)
summary(model1)
pred = predict(model1)

rmse1 = sqrt(mean((pred-data$price)^2))
rmse1

#Adding mroe variables
model2 = lm(price~accommodates+review_scores_rating+bathrooms+bedrooms, data)
summary(model2)
pred = predict(model2)

rmse2 = sqrt(mean((pred-data$price)^2))
rmse2

model3 = lm(price~accommodates+review_scores_rating, data)
summary(model3)
pred = predict(model3)

rmse3 = sqrt(mean((pred-data$price)^2))
rmse3

#Create dummy variables of neighbourhood_group_cleansed
-----------------------------------------------------------------------------
------
  install.packages('dummies')
library(dummies)
data=cbind(data,dummy(data$neighbourhood_group_cleansed,sep = "_"))
str(data)
data$Staten = data$'data_Staten Island'
data$'data_Staten Island'

#Adding dummy columns to the model
model5 = lm(price~normal_accommodates+normal_review_scores+normal_bedrooms+no
rmal_bathrooms+data_Bronx+data_Brooklyn+data_Manhattan+data_Queens+normal_num
ber_of_reviews,data)
summary(model5)
pred = predict(model5)
rmse5 = sqrt(mean((pred-data$price)^2))
rmse5

#add dummy variables to test data
scoringData = read.csv('scoringData.csv')
scoringData=cbind(scoringData,dummy(scoringData$neighbourhood_group_cleansed,
sep = "_"))

str(scoringData)

library(data.table)
setnames(scoringData, old=c("scoringData_Bronx","scoringData_Brooklyn","scori
ngData_Manhattan","scoringData_Queens"), new=c("data_Bronx", "data_Brooklyn",
"data_Manhattan","data_Queens"))

str(scoringData)
pred = predict(model4,newdata=scoringData)
submissionFile = data.frame(id = scoringData$id, price = pred)
write.csv(submissionFile, 'submission4.csv',row.names = F)

#Normalize Data
-----------------------------------------------------------------------------
------
data = mutate(data, normal_review_scores = scale(data$review_scores_rating))
data = cbind(data, normal_accommodates = scale(data$accommodates))
data = cbind(data, normal_bedrooms = scale(data$bedrooms))
data = cbind(data, normal_bathrooms = scale(data$bathrooms))
data = cbind(data, normal_number_of_reviews = scale(data$number_of_reviews))
summary(data$number_of_reviews)

#feature selection
-----------------------------------------------------------------------------
------
  #testing all possible subsets - failed
  install.packages('leaps')
library(leaps)
subsets = regsubsets(price~accommodates+neighbourhood_group_cleansed+number_o
f_reviews+bathrooms+bedrooms+availability_90+availability_365+availability_30
+number_of_reviews+guests_included+cancellation_policy+room_type,data=data,nv
max=5,really.big = T)
summary(subsets)
names(summary(subsets))
subsets_measures = data.frame(model=1:length(summary(subsets)$cp),cp=summary
(subsets)$cp,bic=summary(subsets)$bic, adjr2=summary(subsets)$adjr2)
subsets_measures

#forward stepwise
start_mod = lm(price~1,data=data)
empty_mod = lm(price~1,data=data)
full_mod = lm(price~accommodates+neighbourhood_group_cleansed+number_of_revie
ws+bathrooms+bedrooms+availability_90+availability_365+availability_30+number
_of_reviews+guests_included+cancellation_policy+room_type,data=data)
forwardStepwise = step(start_mod,scope=list(upper=full_mod,lower=empty_mod),d
irection='forward')
summary(forwardStepwise)
```

```
#decision trees and random forest
------------------------------------------------------------------------------
------
  library(rpart); library(rpart.plot)
library(ROCR)

tree1 = rpart(price~accommodates+review_scores_rating+latitude+longitude+neig
hbourhood_group_cleansed+availability_365+availability_30+guests_included+num
ber_of_reviews+room_type,data)
pred = predict(tree1)
rmse4 = sqrt(mean((pred-data$price)^2))
rmse4

install.packages('randomForest')
library(randomForest)
library(caret)
trControl=trainControl(method="cv",number=10)
tuneGrid = expand.grid(mtry=1:5)
cvForest = train(price~accommodates+review_scores_rating+latitude+longitude+n
eighbourhood_group_cleansed+availability_365+availability_30+guests_included+
number_of_reviews+room_type,data=data,method="rf",ntree=1000,trControl=trCont
rol,tuneGrid=tuneGrid )
cvForest

#add more feature (host_is_superhost+bedrooms+bathrooms+review_scores_cleanli
ness+review_scores_checkin+review_scores_communication+review_scores_value+is
_business_travel_ready)
forest = randomForest(price~accommodates+review_scores_rating+latitude+longit
ude+neighbourhood_group_cleansed+availability_365+availability_30+guests_incl
uded+number_of_reviews+room_type+host_is_superhost+bedrooms+bathrooms+review_
scores_cleanliness+review_scores_checkin+review_scores_communication+review_s
cores_value+is_business_travel_ready,data,ntree = 100)
predForest = predict(forest)
rmse4 = sqrt(mean((predForest-data$price)^2))
rmse4

pred = predict(forest,newdata=scoringData)
submissionFile = data.frame(id = scoringData$id, price = pred)
write.csv(submissionFile, 'submission_8.csv',row.names = F)

#try to use ntree=1000, add more features
forest2 = randomForest(price~accommodates+review_scores_rating+latitude+longi
tude+neighbourhood_group_cleansed+availability_365+availability_30+guests_inc
luded+number_of_reviews+room_type+host_is_superhost+host_response_time+extra_
people+minimum_nights+bedrooms+bathrooms+review_scores_cleanliness+review_sco
res_checkin+review_scores_communication+review_scores_value+review_scores_loc
ation+is_business_travel_ready+cancellation_policy,data,ntree = 1000)
predForest2 = predict(forest2)
rmse_f2 = sqrt(mean((predForest2-data$price)^2))
rmse_f2
```

```r
pred = predict(forest2,newdata=scoringData)
submissionFile = data.frame(id = scoringData$id, price = pred)
write.csv(submissionFile, 'submission_9.csv',row.names = F)

#try to use boosting (with differnt parameters)
library(gbm)
boost = gbm(price~accommodates+review_scores_rating+latitude+longitude+neighb
ourhood_group_cleansed+availability_365+availability_30+guests_included+numbe
r_of_reviews+room_type+bedrooms+bathrooms,data,distribution="gaussian",n.tree
s = 500,interaction.depth = 5,shrinkage = 0.04)
predBoost = predict(boost,n.trees = 500,newdata=scoringData,type='response')
rmse4 = sqrt(mean((predBoost-data$price)^2))
rmse4

boost = gbm(price~.,data,distribution="gaussian",n.trees = 100,interaction.de
pth = 3,shrinkage = 0.001)
predBoost = predict(boost,n.trees = 100,newdata=scoringData,type='response')
rmse4 = sqrt(mean((predBoost-data$price)^2))
rmse4
```