Huizhe ZHU (hz2657)

Kaggle report

1. Introduction

This report summarizes the analysis process of Airbnb rentals' price prediction in NYC and the reflection after the project. Appendix contains the R code details of the model with lowest Root Mean Square Error (RMSE) in test data prediction. Some early stage coding is not included in the appendix, but the logic behind is stated clearly in the analysis process.

2. Data analysis process

1. Exploratory analysis

1) Use '*glimpse'* function from dpylr package to get a glimpse of the data, and diagnose which variables are more or less predictive. For example, host ID is generated randomly when hosts registered their account, so it is not predictive in price.
2) Use '*summary(is.na())*' and '*describe'* function to explore the variables with missing data. If a variable contains more than 1/3 missing values, it is risky to make predictions, thus, the variable will be considered to be dropped. For example, 'host_response_time'.
3) Test collinearity by '*VIF'* function, and use 10 as benchmark. In the dataset, the VIF between availability_60 and availability_90 is 22, thus the variable with less significance/importance level is dropped.

   Correlation plot is also used when comparing correlation between variables. However, since the number of variables is more than 40, it is difficult to read the graph. By framing the 1600 correlation data (40*40) into a data frame and using '*filter'* function to rank from the highest to lowest correlation, it is easier to find out the pairs of variables with highest correlation. However, due to the nature of housing variables, many variables are correlated, such as 'bedroom' and 'bathroom', 'bedroom' and 'guest included'.

   Note: 'cor' function can only be applied on numeric variables

Things to improve in this future:

1) To understand the overall distribution of variables and find the unusual outliers, it is efficient to use boxplot to view data visually. In this way, the price below 0 can be discovered and replaced.

2) For collinearity, it is difficult to determine the criteria to decide whether or not to keep the correlated variables with high importance. In other words, there is a tradeoff between importance level and correlation. I assume this requires more advanced knowledge to solve.

2. Variable selection

After exploration process and knowing more about the dataset and variables. Variables that are unable to contribute to prediction, such as host ID, are removed. Next step is to further select variables and clean data. Variable selection is conducted before cleaning data, because it is not necessary to clean the data that will not be used later.

Attempts: At the first attempt, common sense is used to select variables and it turns out to be wrong, because common sense creates bias. At the second attempt, feature selection is used to select variables. However, due to the collinearity issue in the variables, feature selection method could not work. Finally, I realized that because the dataset and relationship among all these variables are very complex, more flexible models such as random forest should be used, rather than linear regression model. And because the nature of linear regression and random forest are very different, variables selected by feature selection may not work for random forest. So, I decided to fit these variables into random forest model firstly, and then use the 'importance' function to select variables.

So far, the exploratory process gave me a basic idea about the variables I need to drop, clean, and further investigate: (1) Drop 31 variables (2) Clean 5 variables (3) Further explore 43 other variables.

Things to improve in the future:

Deciding to drop the 31 columns by assessing its NA values/ VIF /common sense still involves bias. For example, 'zipcode' is not included in selected variables, because I assumed neighborhood is already included to measure location. This is an invalid assumption.

Later I realized, because location is a critical indicator of pricing in real estate industry and 'zipcode' contains so much specific information, it is accurate to use zip code measure location.

Overall, in future, common sense should be used less, and more scientific methods need to be used to select variables.

3. Text analysis

Many valuable information is contained in text, such as description, access, and amenities. Extracting text data from this dataset (csv file) do not need advanced techniques such as NPL, instead, a simple function in R can solve this problem, the '*grepl'* function.

1) Amenities: extract the top 10 amenities, such as Airconditioning, Kitchen, and Elevator etc.
2) Neighborhood: extract neighborhoods that contains more than 1000 housings, such as Chelsea, Bushwick, and Harlem etc.
3) Description, Access, Transit, and Host_verification are also valuable information. However, due to the complexity, they are measured by wordcount. This is to assess how detail the information is provided by hosts, and it is initially assumed that more wordcount would lead to higher price.

3. Prepare the data

During the attempt to run random forest, many error messages appear. It's time to further prepare the data.

1. Missing values

The first reason fitting model does not work is due to the 'NA' values. If not consider the advanced machine learning methods, the NA can be replaced by mode or median, depending on how many values are missing and the type of variable. For example, 'cleaning fee' has high importance level in random forest model, but contains more than 1500 missing values. I replaced the missing values with 0, since most housings do not have cleaning fee. Other variables such as 'if host a super host' has one missing value, and the type of the variable is logical, then it is also appropriate to replace it by 'F', the mode.

2. Transform format

Random forest model has restrictions on the type of predictors. In this case, 'character' variables are transformed to 'factors' by '*as.factor()*' function. The newly added dummy variables are transformed to numeric data type by '*as.numeric()*' function

3. Good to go: Able to run random forest model

After getting the model, '*importance()*' is used to further select variables. The variables with least importance level need to be dropped.

Note: (1) documentary is necessary for future review. (2) '*rm(list = ls())*' is necessary if need to start over the programming.

4. Prepare test data

Prepare test data to be ready in same order of preparing the train data, (1) create new variables from text data, (2) replace NA values, (3) transform data types to numeric or factor.

New levels in test data:

During the prediction process, the error message indicates new levels in test data that train data does not have. By '*glimpse'* function, the variable 'property type' is found to have 3 new levels. By replacing them with the similar and existed levels in train data, the problem is solved. For example, replace 'Castle' by 'Resort'.

4. Conclusion

After 31 submissions and attempts, my final score is 58.77. To further improve, it is suggested to

(1) Work on turning parameters, and try boosting model with cross validation

(2) Try more variable selection methods, such as Lasso

(3) Use less common sense, more specific data information, machine learning methods

(4) Use more advanced skills to balance the tradeoff between lower correlation between predictors and their importance level

Appendix: R code of the prediction with lowest rmse

```
# load the data

install.packages('dplyr')
library(dplyr)

# explotary data analysis
glimpse(analysisData)
str(analysisData)
sum(is.na(analysisData))
summary(is.na(analysisData))

# explore correlation
data_numeric = analysisData %>%
  select(price,
         beds,
         cleaning_fee,
         reviews_per_month,
         accommodates,
         bathrooms,
         bedrooms,
         guests_included,
         extra_people,
         minimum_nights,
         maximum_nights,
         availability_30,
         availability_365,
         number_of_reviews,
         number_of_reviews_ltm,
         review_scores_rating,
         review_scores_accuracy,
         review_scores_cleanliness,
         review_scores_checkin,
         review_scores_communication,
         review_scores_location,
         review_scores_value,
         calculated_host_listings_count,
         calculated_host_listings_count_entire_homes,
         calculated_host_listings_count_private_rooms,
         calculated_host_listings_count_shared_rooms)
describe(data_numeric)
# correlation plot
library(corrplot)
corrplot(cor(data_numeric[,-25]),method = 'square',type = 'lower',diag = F)

# calculate correlation
cor(data_numeric[,-25])
```

```
47.
48. # clean and select data in cleandata1
49. cleandata1 = analysisData %>%
50.   select(price,
51.         host_is_superhost,
52.         neighbourhood_group_cleansed,
53.         cleaning_fee,
54.         reviews_per_month,
55.         is_location_exact,
56.         room_type,
57.         accommodates,
58.         bathrooms,
59.         bedrooms,
60.         minimum_nights,
61.         maximum_nights,
62.         availability_365,
63.         availability_30,
64.         number_of_reviews,
65.         property_type,
66.         review_scores_communication,
67.         review_scores_value,
68.         review_scores_rating,
69.         instant_bookable,
70.         cancellation_policy,
71.         calculated_host_listings_count,
72.         host_listings_count,
73.         host_has_profile_pic,
74.         host_identity_verified,
75.         guests_included,
76.         extra_people)
77.
78. # find the median of missing values
79. median(analysisData$host_listings_count,na.rm = T) #  median
80. summary(analysisData$host_has_profile_pic,na.rm = T)
81. summary(analysisData$host_identity_verified,na.rm = T)
82.
83. # replace missing values by median
84. cleandata1[is.na(cleandata1$host_listings_count), ]$host_listings_count <- 1 #  median
85. cleandata1[is.na(cleandata1$host_has_profile_pic), ]$host_has_profile_pic <- TRUE
86. cleandata1[is.na(cleandata1$host_identity_verified), ]$host_identity_verified <- TRUE
87. cleandata1$host_has_profile_pic <- as.factor(cleandata1$host_has_profile_pic)
88. cleandata1$host_identity_verified <- as.factor(cleandata1$host_identity_verified)
89.
90.
91. # add new variable (1): wordcount
92. install.packages('ngram')
93. library(ngram)
94. cleandata1$wordcount_description<- sapply(strsplit(analysisData$description, " "), length)
```

```
cleandata1$wordcount_transit<- sapply(strsplit(analysisData$transit, " "), length)
cleandata1$wordcount_interaction<- sapply(strsplit(analysisData$interaction, " "), length)
cleandata1$wordcount_access<- sapply(strsplit(analysisData$access, " "), length)
cleandata1$wordcount_host_about<- sapply(strsplit(analysisData$host_about, " "), length)
cleandata1$wordcount_host_verifications<- sapply(strsplit(analysisData$host_verifications, ","), length)

    # add new variable (2): neighborhood
    cleandata1$Bedford_Stuyvesant <- ifelse(grepl('Bedford-Stuyvesant', analysisData$neighbourhood_cleansed), "1","0")
    cleandata1$Bushwick <- ifelse(grepl('Bushwick', analysisData$neighbourhood_cleansed), "1","0")
    cleandata1$Chelsea <- ifelse(grepl('Chelsea', analysisData$neighbourhood_cleansed), "1","0")
    cleandata1$Crown_Heights <- ifelse(grepl('Crown Heights', analysisData$neighbourhood_cleansed), "1","0")
    cleandata1$East_Village <- ifelse(grepl('East Village', analysisData$neighbourhood_cleansed), "1","0")
    cleandata1$Harlem <- ifelse(grepl('Harlem', analysisData$neighbourhood_cleansed), "1","0")
    cleandata1$West_Village <- ifelse(grepl('West Village', analysisData$neighbourhood_cleansed), "1","0")

    # add new variable (3): amenities
    cleandata1$Airconditioning <- ifelse(grepl('Air conditioning', analysisData$amenities), "1","0")
    cleandata1$Kitchen <- ifelse(grepl('Kitchen', analysisData$amenities), "1","0")
    cleandata1$Essentials <- ifelse(grepl('Essentials', analysisData$amenities), "1","0")
    cleandata1$Elevator <- ifelse(grepl('Elevator', analysisData$amenities), "1","0")
    cleandata1$Doorman <- ifelse(grepl('Doorman', analysisData$amenities), "1","0")
    cleandata1$Gym <- ifelse(grepl('Gym', analysisData$amenities), "1","0")
    cleandata1$Internet <- ifelse(grepl('Internet', analysisData$amenities), "1","0")
    cleandata1$Smoking <- ifelse(grepl('Smoking allowed', analysisData$amenities), "1","0")
    cleandata1$parking <- ifelse(grepl('Free street parking', analysisData$amenities), "1","0")
    cleandata1$Heating <- ifelse(grepl('Heating', analysisData$amenities), "1","0")

    sum(is.na(cleandata1))


    # continue to replace NA by median in cleandata1
    cleandata1[is.na(cleandata1$cleaning_fee), ]$cleaning_fee <- 0 # 20
    cleandata1[is.na(cleandata1$host_is_superhost), ]$host_is_superhost <- FALSE # after help
    cleandata1[is.na(cleandata1$reviews_per_month),]$reviews_per_month <- 0.96
    median(analysisData$reviews_per_month, na.rm = T) #0.96

    # convert data to factors (in order to run random forest)
    cleandata1$neighbourhood_group_cleansed <- as.factor(cleandata1$neighbourhood_group_cleansed)
```

```
cleandata1$room_type <- as.factor(cleandata1$room_type)
cleandata1$cancellation_policy <- as.factor(cleandata1$cancellation_policy)
cleandata1$property_type <- as.factor(cleandata1$property_type)
cleandata1$host_is_superhost <- as.factor(cleandata1$host_is_superhost)
cleandata1$is_location_exact <- as.factor(cleandata1$is_location_exact)
cleandata1$instant_bookable <- as.factor(cleandata1$instant_bookable)
cleandata1$Airconditioning <- as.numeric(cleandata1$Airconditioning)
cleandata1$Kitchen <- as.numeric(cleandata1$Kitchen)
cleandata1$Essentials <- as.numeric(cleandata1$Essentials)
cleandata1$Elevator <- as.numeric(cleandata1$Elevator)
cleandata1$Doorman <- as.numeric(cleandata1$Doorman)
cleandata1$Gym <- as.numeric(cleandata1$Gym)
cleandata1$Internet <- as.numeric(cleandata1$Internet)
cleandata1$Smoking <- as.numeric(cleandata1$Smoking)
cleandata1$parking <- as.numeric(cleandata1$parking)
cleandata1$Heating <- as.numeric(cleandata1$Heating)

# explore clean dataset
str(cleandata1)
sum(is.na(cleandata1))
glimpse(cleandata1)

# run random forest model
install.packages('randomForest')
library(randomForest)
set.seed(66)
forestmodel = randomForest(price~., data = cleandata1,ntree = 300)
plot(forest)
varImplot(forest)
importantlist3 = importance(forestmodel)
varImpPlot(forest); importance(forest)  ## see variable importance



# clean data in test (scroing data)
cleandata1_test = scoringData %>%
  select(host_is_superhost,
         neighbourhood_group_cleansed,
         cleaning_fee,
         reviews_per_month,
         is_location_exact,
         room_type,
         accommodates,
         bathrooms,
         bedrooms,
         minimum_nights,
         maximum_nights,
         availability_365,
```

```
            availability_30,
            number_of_reviews,
            property_type,
            review_scores_communication,
            review_scores_value,
            review_scores_rating,
            instant_bookable,
            cancellation_policy,
            calculated_host_listings_count,
            host_listings_count,
            host_has_profile_pic,
            host_identity_verified,
            guests_included,
            extra_people)

    # replace missing values by median
    cleandata1_test[is.na(cleandata1_test$host_listings_count), ]$host_listings_count <- 1 # median
    cleandata1_test[is.na(cleandata1_test$host_has_profile_pic), ]$host_has_profile_pic <- TRUE
    cleandata1_test[is.na(cleandata1_test$host_identity_verified), ]$host_identity_verified <- TRUE
    cleandata1_test$host_has_profile_pic <- as.factor(cleandata1_test$host_has_profile_pic)
    cleandata1_test$host_identity_verified <- as.factor(cleandata1_test$host_identity_verified)

    median(cleandata1_test$host_listings_count,na.rm = T) #  median
    summary(scoringData$host_has_profile_pic,na.rm = T)
    summary(scoringData$host_identity_verified,na.rm = T)

    # (1) add new variable: wordcount
    library(ngram)
    cleandata1_test$wordcount_description<- sapply(strsplit(scoringData$description, " "), length)
    cleandata1_test$wordcount_transit<- sapply(strsplit(scoringData$transit, " "), length)
    cleandata1_test$wordcount_interaction<- sapply(strsplit(scoringData$interaction, " "), length)
    cleandata1_test$wordcount_access<- sapply(strsplit(scoringData$access, " "), length)
    cleandata1_test$wordcount_host_about<- sapply(strsplit(scoringData$host_about, " "), length)
    cleandata1_test$wordcount_host_verifications<- sapply(strsplit(scoringData$host_verifications, ","), length)

    # (2) add new variable: neighborhood
    cleandata1_test$Bedford_Stuyvesant <- ifelse(grepl('Bedford-Stuyvesant', scoringData$neighbourhood_cleansed), "1","0")
    cleandata1_test$Bushwick <- ifelse(grepl('Bushwick', scoringData$neighbourhood_cleansed), "1","0")
```

```
219.    cleandata1_test$Chelsea <- ifelse(grepl('Chelsea', scoringData$neighbourhood_cleansed),
    "1","0")
220.    cleandata1_test$Crown_Heights <- ifelse(grepl('Crown Heights',
    scoringData$neighbourhood_cleansed), "1","0")
221.    cleandata1_test$East_Village <- ifelse(grepl('East Village',
    scoringData$neighbourhood_cleansed), "1","0")
222.    cleandata1_test$Harlem <- ifelse(grepl('Harlem', scoringData$neighbourhood_cleansed),
    "1","0")
223.    cleandata1_test$West_Village <- ifelse(grepl('West Village',
    scoringData$neighbourhood_cleansed), "1","0")
224.
225.    # (3) add new variable: amenities
226.    cleandata1_test$Airconditioning <- ifelse(grepl('Air conditioning', scoringData$amenities),
    "1","0")
227.    cleandata1_test$Kitchen <- ifelse(grepl('Kitchen', scoringData$amenities), "1","0")
228.    cleandata1_test$Essentials <- ifelse(grepl('Essentials', scoringData$amenities), "1","0")
229.    cleandata1_test$Elevator <- ifelse(grepl('Elevator', scoringData$amenities), "1","0")
230.    cleandata1_test$Doorman <- ifelse(grepl('Doorman', scoringData$amenities), "1","0")
231.    cleandata1_test$Gym <- ifelse(grepl('Gym', scoringData$amenities), "1","0")
232.    cleandata1_test$Internet <- ifelse(grepl('Internet', scoringData$amenities), "1","0")
233.    cleandata1_test$Smoking <- ifelse(grepl('Smoking allowed', scoringData$amenities), "1","0")
234.    cleandata1_test$parking <- ifelse(grepl('Free street parking', scoringData$amenities),
    "1","0")
235.    cleandata1_test$Heating <- ifelse(grepl('Heating', scoringData$amenities), "1","0")
236.
237.    cleandata1_test$Airconditioning <- as.numeric(cleandata1_test$Airconditioning)
238.    cleandata1_test$Kitchen <- as.numeric(cleandata1_test$Kitchen)
239.    cleandata1_test$Essentials <- as.numeric(cleandata1_test$Essentials)
240.    cleandata1_test$Elevator <- as.numeric(cleandata1_test$Elevator)
241.    cleandata1_test$Doorman <- as.numeric(cleandata1_test$Doorman)
242.    cleandata1_test$Gym <- as.numeric(cleandata1_test$Gym)
243.    cleandata1_test$Internet <- as.numeric(cleandata1_test$Internet)
244.    cleandata1_test$Smoking <- as.numeric(cleandata1_test$Smoking)
245.    cleandata1_test$parking <- as.numeric(cleandata1_test$parking)
246.    cleandata1_test$Heating <- as.numeric(cleandata1_test$Heating)
247.    sum(is.na(cleandata1_test))
248.
249.    # (4) replace NA value
250.    cleandata1_test[is.na(cleandata1_test$cleaning_fee), ]$cleaning_fee <- 0 # after help
251.    cleandata1_test[is.na(cleandata1_test$host_is_superhost), ]$host_is_superhost <- FALSE #
    after help
252.    cleandata1_test[is.na(cleandata1_test$reviews_per_month),]$reviews_per_month <- 0.95
253.    median(cleandata1_test$reviews_per_month, na.rm = T) #0.95
254.
255.    # (5) convert to factors
256.    cleandata1_test$neighbourhood_group_cleansed <-
    as.factor(cleandata1_test$neighbourhood_group_cleansed)
257.    cleandata1_test$room_type <- as.factor(cleandata1_test$room_type)
```

```
cleandata1_test$cancellation_policy <- as.factor(cleandata1_test$cancellation_policy)
cleandata1_test$bed_type <- as.factor(cleandata1_test$bed_type)
cleandata1_test$property_type <- as.factor(cleandata1_test$property_type)

# (6) Deal with new levels in Property type
# replace with levels that are similar
cleandata1_test[cleandata1_test$property_type == 'Castle',]$property_type <- 'Resort'
cleandata1_test[cleandata1_test$property_type == 'Farm stay',]$property_type <- 'House'
cleandata1_test[cleandata1_test$property_type == 'Casa particular (Cuba)',]$property_type
<- 'House'
dada = cleandata1_test[cleandata1_test$property_type == 'Casa particular (Cuba)',]
levels(cleandata1$property_type)
levels(cleandata1_test$property_type)
str(cleandata1)
str(cleandata1_test)
?droplevels.factor
library(gdata)
drop.levels(cleandata1_test$property_type)

x <- x[x!="Casa particular (Cuba)"]
df <- data.frame(a=x,b=x,c=x)
df <- drop.levels(df)
factor
cleandata1_test$property_type <- droplevels(cleandata1_test$property_type )
cleandata3_test = select(cleandata1_test , -c('minutes','train','subway'))


# convert test data to types of train data
cleandatatestaaaa = cleandata1_test
cleandatatestaaaa <- rbind(cleandata1[1, -1], cleandatatestaaaa)
cleandatatestaaaa <- cleandatatestaaaa[-1,]
str(cleandata3_test)
str(cleandata1)
sum(is.na(cleandata1_test))

# predict testing data
install.packages('caret')
install.packages('ggplot2')
library(randomForest)
library(caret)

library(ggplot2)
library(caret)
predforest2 = predict(forestmodel,newdata=cleandatatestaaaa)
predforest2
# score = 58.77108

# write file in csv
```

```
submission14 = data.frame(id=scoringData$id, price = predforest2)
write.csv(submission14,'14submission.csv',row.names = F)
```