APAN5200 Kaggle Project Report

Ferra Suryani

Insights from exploring the data

After understanding the objectives of the project, I gave a lot of thoughts to what makes Airbnb properties more or less desirable that could affect the price listings from the customers' perspectives. The major aspects could be the neighborhood, amenities, distance from subway stations or other major tourist spots, or how long the listings have been posted and booked in the past as the initial assumptions. These aspects helped a lot in considering which variables would affect the price most as I select the variables to work with.

Efforts to prepare the data

I encountered a few issues as I was exploring the dataset. I started the exploratory data analysis step by loading the listings and looking at the data types. I noticed how most of the values are 'messy', where most of the observations had inconsistencies in the formatting of values, missing entries, and multiple variables in one column to name a few. As a part of the data cleaning process, missing values were found for variables such as *cleaning_fee*, *security_deposit*, and *square_feet*. I also noticed that variables related to date (*host_since, first_review, last_review*) were not formatted consistently. While for *zipcode*, To avoid overlapping work as I perform data manipulation and data transformation, I combined both scoring and analysis datasets into one under *combinedData* alias.

----------Line 17: kaggle_sub11----------

```
## DATA PEPARATION
scoring_data <- read.csv('scoringData.csv', header = TRUE)
analysis_data <- read.csv('analysisData.csv', header = TRUE)
analysisData <- analysis_data %>% mutate(type = "analysis")
scoringData <- scoring_data %>% mutate(type = "scoring")
combinedData <- bind_rows(analysisData, scoringData)
```

----------Line 24: kaggle_sub11----------

I selected mostly variables related to neighborhood and important features of the Airbnb listings.

```
data <- combinedData %>% select(availability_365, bedrooms, bathrooms,
                                accommodates, number_of_reviews, cleaning_fee, security_deposit,
                                bed_type, host_has_profile_pic, neighbourhood_cleansed, square_feet,
                                host_since, zipcode, first_review, last_review, host_is_superhost,
                                property_type, amenities, room_type, neighbourhood_group_cleansed,
                                price, review_scores_rating)
```

As we jumped into the variable transformation step, I converted most of the variables I selected into factors to standardize the independent variables and for better fitting in the model.

----------Line 39: kaggle_sub11----------

```
# bed_type
data$bed_type <- as.factor(data$bed_type)

# host_has_profile_pic
data$host_has_profile_pic <- as.factor(data$host_has_profile_pic)

# neighbourhood_cleansed
data$neighbourhood_cleansed <- as.factor(data$neighbourhood_cleansed)

# host is superhost
data$host_is_superhost <- as.factor(data$host_is_superhost)

# property type
data$property_type <- as.factor(data$property_type)

# room_type
data$room_type <- as.factor(data$room_type)

# neighbourhood_group_cleansed
data$neighbourhood_group_cleansed <- as.factor(data$neighbourhood_group_cleansed)
```

----------Line 58: kaggle_sub11----------

As a part of data manipulation and transformation, I noticed how the date-related variables were not in the correct date format. To make the variables easier to measure, I transformed the *host_since* variable into *host_length* and subtracted it by the earliest date detected from the column. The same goes to *first_review* and *last_review* since these would allow me to measure the correlation with price.

----------Line 62: kaggle_sub11----------

```
#host_since_length
data$host_since = as.Date(data$host_since)
data$host_length = as.integer(as.Date("2018-11-29") - data$host_since)

#since_first_review
data$first_review = as.Date(data$first_review)
data$since_first_review = as.integer(as.Date("2018-11-29") - data$first_review)

#since_last_review
data$last_review = as.Date(data$last_review)
data$since_last_review = as.integer(as.Date("2018-11-29") - data$last_review)
```

----------Line 72: kaggle_sub11----------

As I was digging into the *zipcode* variable by extracting the unique values, I noticed how some of the values are inconsistent. To make it more aligned, this is how I approached it.

----------Line 74: kaggle_sub11----------

```
### transforming zipcode
table(data$zipcode)
unique(combinedData$zipcode)

data$zipcode[data$zipcode=="11103-3233"] = 11103
data$zipcode[data$zipcode=="11249\n11249"] = 11249
data$zipcode[data$zipcode=="10003-8623"] = 10003
data$zipcode[data$zipcode=="11413-3220"] = 11413
data$zipcode[data$zipcode=="10065"] = 10021
data$zipcode[data$zipcode==11249] = 11211
data$zipcode[data$zipcode=="11249"] = 11211
data$zipcode = as.factor(data$zipcode)
```

----------Line 85: kaggle_sub11----------

The next step would be to impute the missing values of a few variables I picked. For *host_is_superhost*, it was the best approach to convert N/A value to 'F' since there was no further information in regards to the superhost. For other variables, taking the average of all values in that specific column was the most appropriate way to impute the missing values since it incorporated all the values from its variable. For *square_feet* specifically, taking the middle value of the overall value was the best approach since it measures the size of the listings and would not make sense to take the average.

----------Line 91: kaggle_sub11----------

```
### impute missing values on host_is_superhost
data$host_is_superhost[is.na(data$host_is_superhost)] <- "f"

### impute missing values on cleaning_fee with mean value
data$cleaning_fee[is.na(data$cleaning_fee)] <- mean(data$cleaning_fee, na.rm=TRUE)

### impute NA value in square feet with median value
data$square_feet[is.na(data$square_feet)] <- median(data$square_feet, na.rm=TRUE)

### impute missing values on zipcode
data$zipcode <- as.character(data$zipcode)
data$zipcode[is.na(data$zipcode)]<- "0"
data$zipcode <- as.factor(data$zipcode)

### impute more missing values
data$host_length[is.na(data$host_length)] <- mean(data$host_length, na.rm=TRUE)
data$since_first_review[is.na(data$since_first_review)] <- mean(data$since_first_review,
na.rm=TRUE)
data$since_last_review[is.na(data$since_last_review)] <- mean(data$since_last_review,
na.rm=TRUE)
data$security_deposit[is.na(data$security_deposit)] <- mean(data$security_deposit, na.rm=TRUE)
```

----------Line 112: kaggle_sub11----------

Another important approach that I took into consideration was to break down the *amenities* variable and create dummy variables based on levels that would possibly add costs to the overall price listings.

----------Line 114: kaggle_sub11----------

```
### creating dummy variables for amenities
data$Pool <- as.integer(str_detect(data$amenities, "Pool"))
data$Beachfront <- as.integer(str_detect(data$amenities, "Beachfront"))
data$Indoorfireplace <- as.integer(str_detect(data$amenities, "Indoorfireplace"))
data$Gym <- as.integer(str_detect(data$amenities, "Gym"))
data$Elevator <- as.integer(str_detect(data$amenities, "Elevator"))
data$Smartlock <- as.integer(str_detect(data$amenities, "Smartlock"))
data$Breakfast <- as.integer(str_detect(data$amenities, "Breakfast"))
data$Washer_and_Dryer <- as.integer(str_detect(data$amenities, "Washer_and_Dryer"))
```

----------Line 112: kaggle_sub11----------

I did a total of 8 submissions in the end and I constantly added new variables in every submission. After going through all the variables listed and doing more data cleaning, this would be the final list of variables mainly used to predict the price listings as compared to the first submission.

----------Line 127: kaggle_sub11----------

```
used_vars <- c("Washer_and_Dryer", "Breakfast", "Indoorfireplace", "Gym", "Elevator", "Smartlock",
"Pool", "Beachfront",
           "availability_365", "bedrooms", "bathrooms", "host_length", "since_first_review",
           "since_last_review", "accommodates", "number_of_reviews", "cleaning_fee",
           "security_deposit","bed_type", "host_has_profile_pic", "neighbourhood_cleansed",
           "square_feet","zipcode", "host_is_superhost","property_type", "room_type",
           "neighbourhood_group_cleansed","price", "review_scores_rating")
data_vars <- subset(data, select = used_vars)
```

----------Line 133: kaggle_sub11----------

Analysis techniques explored

The way I approached the process was by adding more variables after each submission. I mostly hand-picked variables to start with but the lasso regression I ran after preparing all the data helped in determining which ones to include for the actual modeling phase. After ensuring all the data under each variable are in the same formatting and no more missing values, I performed lasso regression to select the best variables to use in the model building step as measured by its coefficient values.

For the third submission, I used Random Forest technique with 1000 trees and only included 4 variables. It didn't result in low RMSE because it didn't involve any boosting and tuning. This was when I started considering using cvBoosting for my next submission since boosting algorithms helps in improving the accuracy of decision trees. I still did not get the result I wanted so I considered

using GBM Boosting technique for my next submission since I understood the algorithm consider all errors occurred in the previous trees and reduced the likelihood of bias. My final submission involved GBM Boosting and managed to give me the lowest RMSE value.

Best Analysis technique

Performance was mainly measured by the RMSE value and GBM technique seems to work best out of the other techniques I did. I also noticed that the greater the number of trees and the lower the shrinkage value is, the better the accuracy of the model becomes. I initially tried running the model with just 1000 trees and shrinkage of 0.01 and the RMSE did not return low. I continuously made adjustments to both parameters and the best combination I found was 10,000 trees and shrinkage of 0.005.

Results

After a few tries with different modeling techniques, the Gradient Boosting Machines gave me the best results in terms of its RMSE value, This resulted in being ranked 205th on the leaderboard rank. To improve the accuracy of the model, I would have attempted to tune the model by using XGBoost. Another thing I would have done differently would be to perform a feature selection method to screen the best predictors and to ensure I select the variables that positively affect the price listings.

Conclusion

Through this project specifically, I got hands-on experience working with real datasets and mostly learned about the most appropriate approach of using machine learning techniques to solve real problems in terms of the steps to follow prior model building. I also learned throughout the project that the most important step of doing predictive modeling is in the data preparation phase instead of running the model itself.