APAN 5200 Assignment: Kaggle Project Report

Xinyi Qian (Uni: xq2190, Kaggle username: happy coder)

**Background:** For this Kaggle Project, a listing of over 25,000 Airbnb rentals in NYC is given. The goal is to predict the price for rentals using over 90 variables.

1. **Data analysis process**

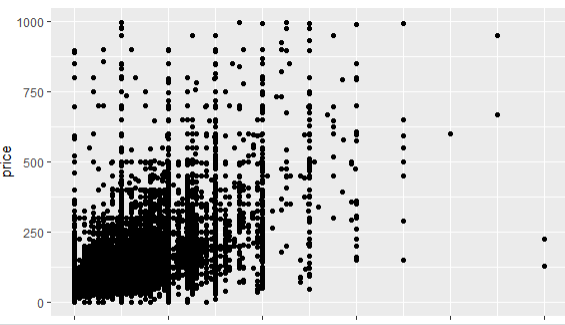
**Step 1: Data Exploration**

Since there are over 90 variables to choose from, the first step is to explore the dataset by using str() and summary() to look at the number of levels and data type of each variable in the dataset. I found that many variables contain a lot of levels, different data classes, many NA values, misspelled characters etc. I also used summary() for each individual variable to see how many NA’s it has. The first set of data that I decided to get rid of after looking at the structure is:

* **Unique** values: id, name, scrape\_id, etc.
* **URL** format data: host\_thumbnail\_url, host\_picture\_url, etc.
* Data that is **hard to lower the level** to be used: space, description etc.
* Data that **only has 1 level** thus useless in prediction: experiences\_offered etc.
* Data that is **very skewed** (only has 2 levels and most of the values belong to 1 level): jurisdiction\_names, country\_code, country, state, market, etc.
* Data that I think has **no correlation to the price**: host\_location etc.
* Data that is **redundant** with another variable: host\_listings\_count etc.
* Data that has **too many missing values** (% of NA >10% of total observations): square\_fee, weekly\_price, monthly\_price, etc.

**Step 2: Data cleaning**

The remaining data are what I thought might be useful at this stage but need further exploration. I grouped data based on different categories first. Then, I examined each category separately to narrow down my useful variables. I also used scatterpoint graph in *ggplor2 package* to visualize the relationship between each variable and price and got rid of the variable that has no correlation with price.

\*\*\* cleaning fee V.S. price

1. **Host** related:

I used substr() function in *stringr package* to extract the year from variable: **host\_since** to create a new variable that only has year because when the host started might affect the price. I also used as.numeric() function to change the data type for **host\_response\_rate** since the original data type has many factors and it is very likely that new level would occur in scoringdata set. Then I ran a linear model lm() to check the coefficiency and kept the data that has high coefficiency (p<0.05) such as **host\_since, host\_response\_rate, host\_total\_listings\_count.**

1. **Geographic** related:

I got rid of variables such as street, neighbourhood\_cleansed, city, smart\_location because there are lots of dirty values with misspelled and missing characters which makes it hard to clean for use. I also got rid of zipcode because it has too many levels and it is very likely that new level would appear in the scoringdata.

I kept **neighbourhood\_group\_cleansed** because the values are clean and the number of levels is reasonable, I also kept **latitude** and **longitude** because they are numeric and the coefficiency is high.

1. **Room type** **related**:

I noticed that there are NA values in variables such as **beds**. I used median value to replace the missing value by using *caret package* because I am not certain that missing = no bed in this case. I also used sapply and strsplit() function in *stringr package* to create a new variable that counts the number of **amenities** which I thought might be useful. Also, I used 0 to replace the NA values in **cleaning\_fee** and **security\_deposit** because I feel like this is more like a Yes/No question in the real life. I then piped my new variables to the dataset that I am using to build my model by using mutate function in the *caret package*.I then ran a linear model to check the coefficiency of all those room related variables and kept the variables that have high coefficiency: **amenities**, **room\_type, accommodates, bathrooms, bedroom, beds,** **guests\_included, extra\_people, minimum\_nights, instant\_bookable, is\_business\_travel\_ready, cancellation\_policy.**

1. **Availability** related:

I noticed that these variables are very similar in nature and therefore I run vif() in *car package* to check the multicollinearity and only kept **availability\_365** since it has the lowestmulticollinearity number.

1. **Review** related:

I kept most of review\_scores related data since I think that reviews are useful in real life. I also noticed that seasonality might affect price. So, I extracted the year from **last\_review** and created a new variable called **last\_review\_year** to use in my model.

In my R-Markdown file, I have also attached some additional data cleaning procedures that I thought might be useful for reference, though I did not end up using them in my model because they did not help improve the performance score when I tested my model performance.

**Step 3: Feature selection:**

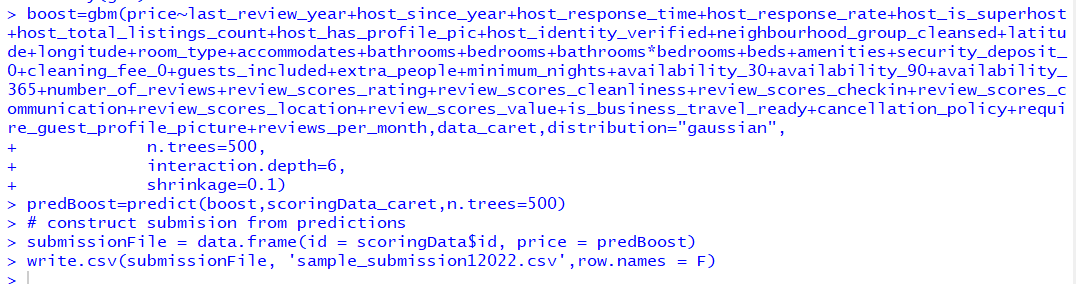
1. After data cleaning procedure above, I still have about 20-30 variables. I used **lasso** model from *glmnet package* and the variables got shrinked were: host\_response\_rate, host\_total\_listings\_count, beds, is\_business\_travel\_ready and cancellation\_policy. However, the higher model performance score belongs to the model that contains more variables/before using lasso.
2. I have also tried tree model from *rpart package* but only a few number of variables were selected and therefore I did not think tree was helpful.
3. I have also used hybrid stepwise regression model and the only variable it suggested to remove was: is\_business\_travel\_ready.

I was using summary() and vipImp() in *caret package* the whole time to check the coefficiency/importance of each variable in my model whenever I update my model. I used the feature selection result as a reference and kept changing my variables to assess my model performance.

**Step 4: building predictive models and submitting to Kaggle to check the performance:**

1. I firstly used **linear regression model,** but it did not generate high model performance score. My RMSE was never below 60.
2. **Linear regression with interaction** improved my R^2 a lot but my model performance was still around 60.
3. Then I used **Boosting with cross validation** which took real long time but it gave me a higher model performance: RMSE = 54.65070
4. I have also tried **random forest** with cross validation which took the longest time and it gave me similar model performance: RMSE = 54.60954
5. Finally, I added some more variables and changed my tuning parameters in **boosting model**, I achieved my highest model performance: RMSE = 53.32276
6. **What I have learned from this experience**
7. **The ingredients of the final analysis:**

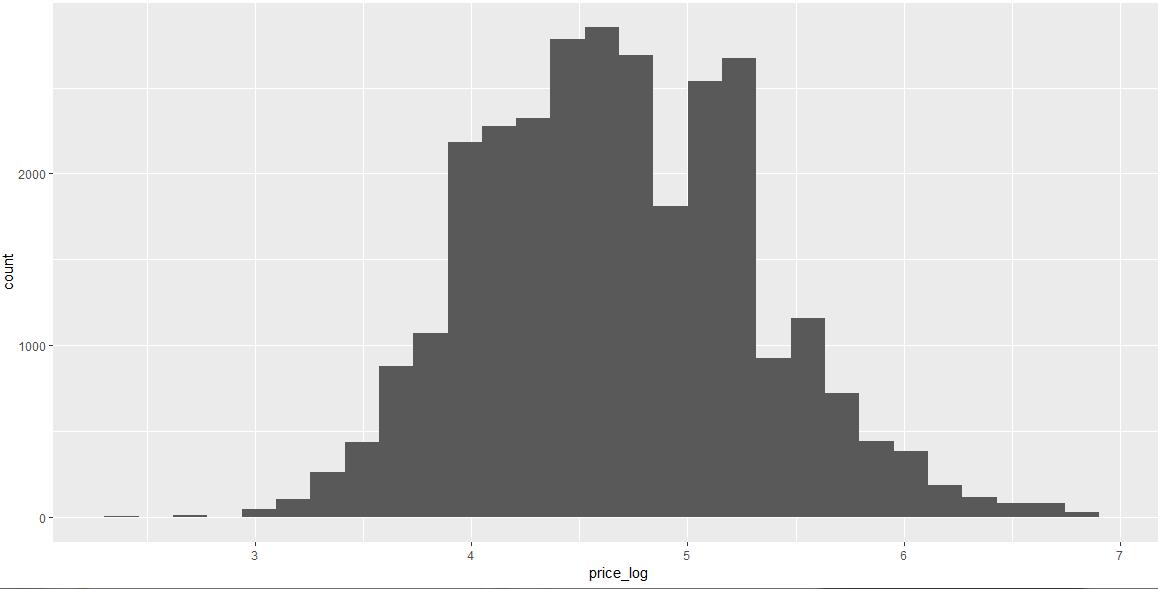
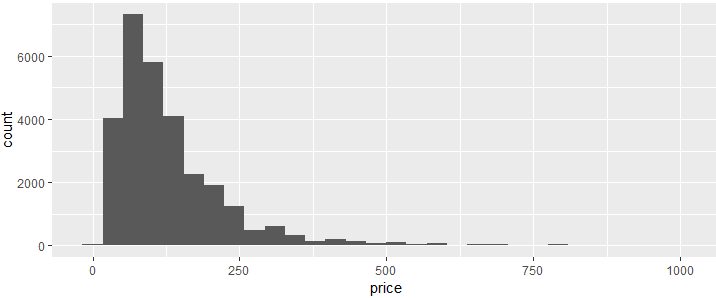
The model that gave me the highest performance is boosting with below variables and tuning parameters: n.trees = 500, interaction.depth=6, shrinkage =0.1.



1. **Failed steps/missteps along the way + lesson learned**
   1. Jumped into conclusion too fast: I thought variables such as latitude and longitude were useless in the beginning because the numbers are very similar. But they were actually great indicators since they are numeric and location is key for pricing in the real life. So I learned that I should not remove data too fast based on my initial judgement. Also, some variables such as summary, amenities, host\_since etc. did not seem useful to me in the beginning, but they became useful after I extract the year or count the number of them.
   2. Significance of variables keep changing: Variables could become less/more statistically significant when they were with different other variables.
   3. Failure to convert the data back to factor type after data cleaning will cause error when building the model.
   4. Categorical variables, especially with many levels, were slower to be processed in R compared to numeric ones, therefore we would want fewer levels if possible.
   5. I had error running the prediction function because I did not prepare the scoring data the same way as I prepared analysis data.
   6. Not all newly created variables were helpful for my model, the best combination came from constantly testing.
   7. Bigger number of trees does not necessarily lead to a better model. I tried n.tree = 10,000 / 1,000 / 500. Surprisingly, the lower n.trees gave me better performance.
   8. I was little surprised that the variables selected from lasso did not generate better performance, neither did tree plot. Therefore, I spent most of time being frustrated to keep changing and trying the combination of my variables to use. However, after my last Kaggle submission which generated my best score, I just realized that I might have made a mistake that I used lasso and stepwise after I already manually made judgement about variable selection. I should have prepared my data to use in lasso and stepwise as my first variable selection step. This is one of the biggest lessons that I learned from this project because in the real life, efficiency is just as important as the accuracy of the model. The preparation step is very important to generate a better outcome later.
2. **Remaining questions to keep in mind during future study:**

Overall, this project was a great experience for me to apply both inside and outside class techniques to handle dataset in the real life. However, during my process, I still have some unsolved questions that I need to keep in mind during my future study:

1. I still used a lot of personal judgement when selecting variables. How can I make sure my selection is not biased?
2. The tuning for my best model was different from the bestTune parameters of the cross validation. Why is that?
3. How does each tuning parameter affect my outcome? (for instance, higher number of n.tree does not necessarily lead to higher performance, but how low/high could it go?)
4. The value of price is not normally distributed, how did that affect my prediction?



1. Since the final performance is judged by the other half of the data (private data), I am also very curious about how can we ensure the constant high performacne of our model and why would there be a (big) differnce of the RMSE score between public and private data?