Kaggle Report

——How much for your Airbnb

Xinying Xu
xx2314

Introduction

This Kaggle competition is to predict Aribnb price per night based on 95 variables and over 25,000 observations. We need to generate a prediction for each id in scoringData.csv. And submissions will be evaluated based on RMSE (root-mean-square error). We need to lower RMSE as much as possible.

Exploring the data

First, we need to explore the data and do a lot of data cleaning. By taking a quick overview of the dataset, I deleted some useless dirty variables such as 'listing_url' and left the dataset with 63 variables. Then, I realized that there are some duplicated levels in the variable such as 'smart_location' and below is a portion of this variable:

	Var1	Freq
1	Brooklyn, NY	2
2	New York , NY	1
3	New York, NY	1
4	8425 Elmhurst avenue , NY	1
5	Arverne , NY	2
6	Arverne, NY	5
7	Astoria - New York, NY	1
8	Astoria , NY	28
9	Astoria New York, NY	2
10	Astoria, New York , NY	1

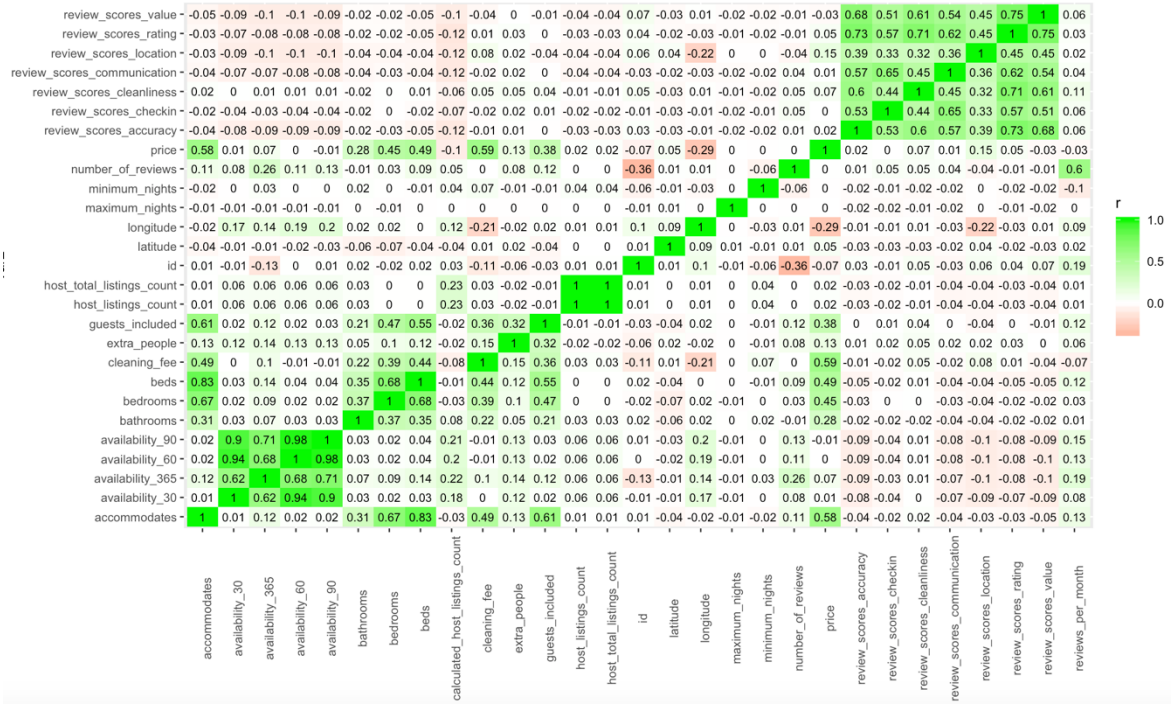
We can see that the second and third row represent the same location. So I deleted such variables. Besides, the variables 'has_availability' and 'requires_license' have only one level of 't' or 'f', so they have no influence on the prediction process. Now there are 57 variables in the dataset.

Furthermore, by using summary() function, I found out the variables 'square_feet', 'weekly_price' and 'monthly_price' have over 10,000 NAs, so it would be imprecise to use them to predict price and I deleted them too.

Now I have 52 variables to explore except 'id' and 'price'.

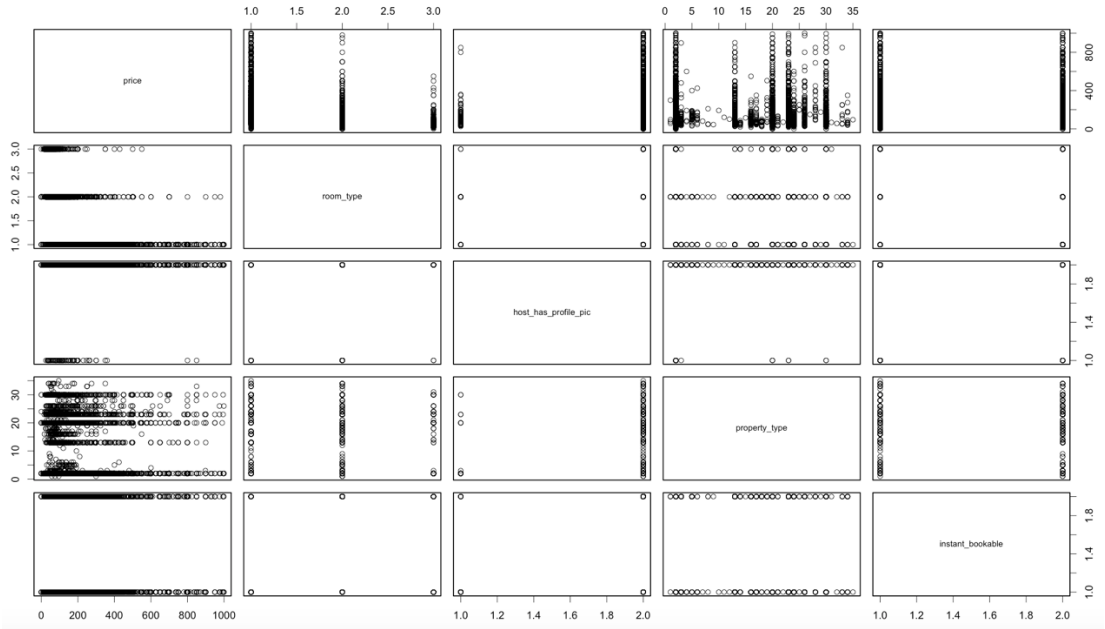
Preparing the data for analysis

To prepare the data for analysis, I need to select some useful variables that have great relationships with 'price'. For numerical data, I built a correlation matrix to visualize the correlation between numerical variables and price.



From the matrix, we can conclude that 'review_scores_location', 'review_scores_cleanliness', 'review_scores_rating', 'review_scores_value', 'longitude', 'latitude', 'guests_included', 'extra_people', 'id', 'cleaning_fee', 'beds', 'bedrooms', 'bathrooms', 'availability_365' and 'accommodates' have strong correlations with price. So I put them in the model.

As for non-numeric variables, I used scatterplot matrix to see the correlation between price and other categorical variables. And we can select suitable categorical variables according to the matrix below. (This matrix includes only a part of categorical variables)



After drawing some scatterplot matrices, I selected these variables: 'host_is_superhost', 'neighbourhood', 'require_guest_phone_verification', property_type', 'room_type', 'cancellation_policy', 'is_business_travel_ready', 'instant_bookable', 'require_guest_profile_picture', 'is_location_exact', and 'host_identity_verified'.

Since we first need to create a model then apply the model to scoringData to make predictions, I combined analysisData and scoringData by rows using rbind() function and named it 'combinedData'. This function requires the column of the two datasets be the same, so I add a column of 'price' in the scoringData and make the value be NA.

With data visualization, I managed to select useful variables. Since some variables have NAs, they cannot be used to predict price. So I used the following code to fill missing values with medians. The results were stored in the data.frame 'combinedDataClean'.

```
# Fill in missing values (NAs) ------------------------------------------
combinedDataClean <- predict(preProcess(combinedDataSelected, method = 'medianImpute'),
                             newdata = combinedDataSelected)
```

Then I converted factor levels to numbers such as 0 and 1, because XGBoost manages only numeric vectors. Although 'neighbourhood' and 'property_type' have relatively strong correlations with price, they have too many levels and are not suitable for XGBoost. So they were excluded. Next, I split 'combinedDataClean' into 'analysis' and 'scoring' data, with 'analysis' representing analysisData and 'scoring' representing scoringData.

Modeling techniques and Results

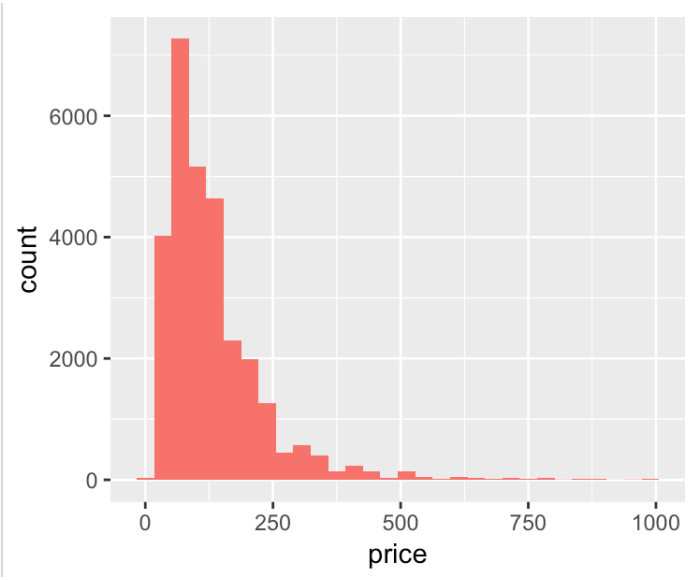
After exploring the data and preparing the data for analysis, I came to the modeling part. First, I used linear regression model to predict Airbnb price per night. I actually made some progress compared to the original codes in the website. But I ran into some error when applying the model to the test data:

```
Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xlevels) :
  factor cancellation_policy has new levels super_strict_30, super_strict_60
```

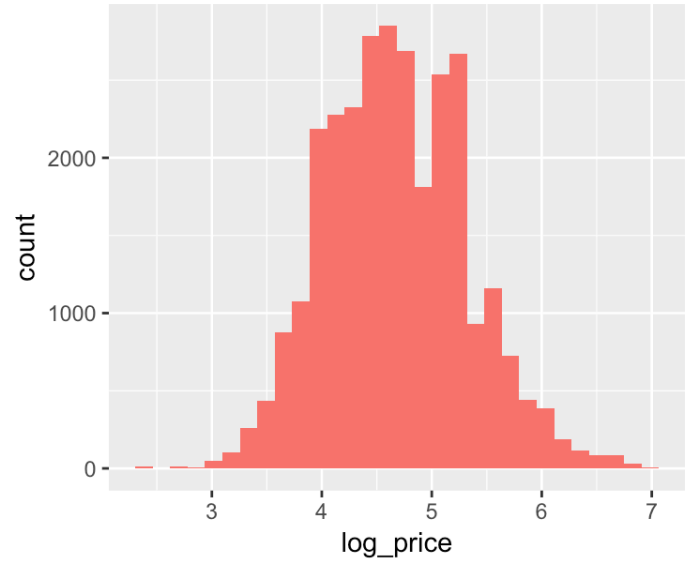
The error told us that there were some new levels existing in the scoringData, and I came up with a solution to replace the new levels with the existed levels in the train dataset. And the problem was solved.

As the competition progressed, I did more research and found out that using log(price) will be more efficient in the linear regression model. Logarithmically transforming variables in a regression model is a very common way to handle situations where a non-linear relationship exists between the independent and dependent variables. Using the logarithm of one or more variables instead of the un-logged form makes the effective relationship non-linear, while still preserving the linear model.

Logarithmic transformation is also a convenient means of transforming a highly skewed variable into one that is more approximately normal. For instance, if we plot the histogram of price, we see a significant right skew in this data:



If we plot the histogram of the logarithm of price, however, we see a distribution that looks much more like a normal distribution:



I used log(price) to reduce the RMSE for the model a lot.

As the course went on, we learned advanced trees such as random forest and boosting. So I used gbm() function to make predictions, and set parameters such as number of trees to be 100,000 and shrinkage to be 0.001. But the result I got was not so satisfying and I still need time to explore this algorithm.

Then I used random forest to predict price per night of Aribnb and reduced RMSE further. But there are some limitations when it comes to random forest. For example,

random forest can be time-consuming and it can not handle categorical variables with more than 53 categories. So I had to delete some useful variables such as 'neighbourhood', 'calendar_updated' and 'zipcode'.

```
Error in randomForest.default(m, y, ...) :
  Can not handle categorical predictors with more than 53 categories.
```

But in general, I got the lower RMSE using random forest than linear model and boosting.

Finally, I used XGBoost to get my best result. XGBoost is an optimized gradient boosting library designed to be highly efficient and portable. Although the learning process was not easy, it eventually paid off. It took me much less time to run the model than random forest, which enabled me to tune parameters in a quick way. Also, as discussed above, logarithmic transformation of price can deal with skewness of response variables. So I created a new vector, 'LogPrice' to be included in the XGBoost model. But such an efficient model can have its shortcomings. For example, XGBoost manages only numeric vectors. So I had to convert those categorical variables to numeric variables before running the model.

Conclusion

Below is the result of my submissions of RMSE sorted in increasing order (33 submissions):

	RMSE
sample_submission.csv	52.40795
sample_submission.csv	52.43394
sample_submission.csv	52.56867
sample_submission.csv	53.47827
sample_submission.csv	53.65771
sample_submission.csv	53.66130
sample_submission.csv	53.68110
sample_submission.csv	53.75483
sample_submission.csv	56.41717
sample_submission.csv	56.60258
sample_submission.csv	56.75259
sample_submission.csv	56.88747
sample_submission.csv	57.01161
sample_submission.csv	57.01161
sample_submission.csv	57.41305
sample_submission.csv	59.46114
sample_submission.csv	60.10719
sample_submission.csv	60.37497
sample_submission.csv	63.11787
sample_submission.csv	63.91719
sample_submission.csv	64.05329

sample_submission.csv	64.41454
sample_submission.csv	64.54415
sample_submission.csv	65.87818
sample_submission.csv	66.59628
sample_submission.csv	84.06251
sample_submission.csv	84.09725
sample_submission.csv	84.09725
sample_submission.csv	91.08256
sample_submission.csv	104.21001
sample_submission.csv	104.43897
sample_submission.csv	104.83673

And the table below is the best result I got from XGBoost, random forest, GBM and linear model respectively.

	RMSE
XGBoost	52.40795
Random Forest	53.65771
GBM	57.01161
Linear model	60.10719

In conclusion, I first explored the dataset and deleted some useless variables. Then I used plots to visualize data and selected variables that have strong relationships with price. Next, for the selected data, I did some data cleaning, for example, I dealt with NAs in some variables. Finally, I used different models and tuned parameters to get the lowest RMSE.

This Kaggle competition actually gives me some valuable insights. Since I am a new coder with no former experience regarding programming, I learned how to analyze data using technical skills. And I learned that I need to be patient with data cleaning and not rush to the modeling part. This report can also give some advice to Aribnb about the factors giving rise to higher price.