Using Decision Trees to Explore Airbnb Data in NYC

Mengyao Liu (ml7hc)

Nov. 14th, 2019

Keywords: business — internet — regression analysis — classification analysis — New York City — hotels and accommodations

1 Introduction

Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present more unique, personalized way of experiencing the world. In a big city like New York City (NYC), which attracts millions of tourists every year[1], there are over 10,000 hosts posting over 45,000 house listings on Airbnb in 2019 alone[2]. Suppose you have a spare room/house in NYC, you would like to take advantage of the prospering tourism in NYC and rent the room/house at a reasonable price, so that the spare room/house can be made use of most of the time to gain profit. However, making an appropriate price is no easy task. On the one hand, the host would prefer to set the price high to gain more profit. On the other hand, too high a price will decrease the willingness of the potential guest booking the house and thus result in fewer sales. Now suppose you are a tourist who is looking for accommodation on Airbnb. You have several options in your wishlist but you need more time for further comparison or deeper investigation. Compared with an imperfect choice, you are more worried about your favorite being taken by someone else before you make up your mind. Thus you may want to know the popularity of this listing to decide whether to place the order soon or you could wait until later. Fortunately, thanks to Internet, we can find public dataset from Airbnb and help make a wise decision by learning from the history.

We use the dataset describing the listing activity and metrics in NYC, NY for 2019 from Kaggle[2]. Essentially the data file is a `csv` table. This table includes all needed information to find out more about hosts, geographical availability, necessary metrics to make predictions and draw conclusions. This public dataset is part of Airbnb, and the original source can be found on this website.

The first question we want to answer is to predict the best price range of a room/house for a host on Airbnb using regression analysis to make sure the room/house will be rented more than two thirds of the time of a year. The response will be the price. The second question we want to answer is whether a room/house on Airbnb is popular or not. The response will be two classes of popularity based on the availability (see §2).

2 Data Processing & Cleaning

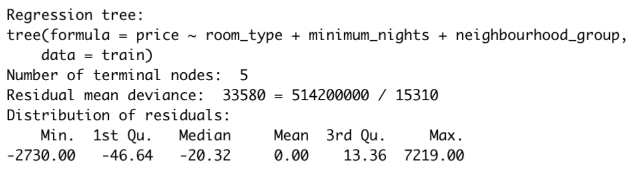
We will use the following variables in the data set: `neighborhood_group`, `room_type` and `minimum_nights`, `price`, `number_of_reviews`, `reviews_per_month`, and `availability_365`. `neighborhood_group` and `room_type` are categorical variables. `neighborhood_group` is a string of the neighborhood location name with five classes: Bronx, Brooklyn, Manhattan, Queens, and Staten Island. `room_type` is a string of space type with three classes: Entire home/apt, Private room, Shared room. The others are quantitative variables. `minimum_nights` is a number specifying the amount of nights minimum. `price` denotes the price per night in dollars. `availability_365` is the number of days when listing is available for booking. We define “Popular” as `availability_365` < 122. We define “Unpopular” as `availability_365` $\geqslant 122$.

We have also found that there are 11 observations that have `price = 0`. This is unusual and we do not aim to investigate the reasons here. Thus we remove those 11 observations. Additionally, we set `reviews_per_month` of those observations with missing `reviews_per_month` to zero since they have `number_of_reviews = 0`. In the sample of 48,884 observations, we have 30,632 observations in class “Popular” and 18,252 observations in class “Unpopular”.

3 Regression Tree

From the viewpoint of a host, the predictors we will use are `neighborhood_group`, `room_type` and `minimum_nights`. The response will be `price`. Since we expect the room/house to be rented most of the time, we would treat the listings that are rented most of the time in 2019 as successful examples to learn from. Thus we only use part of the observations which are in the ”Popular” class. There are 30,632 satisfactory observations.

We randomly split the data into a training set and a test set of equal sizes. Then we fit a regression tree on the training set using recursive binary splitting. The results are shown in Figure 1 and Fig 2. All the three predictors are used. The tree has 5 terminal nodes. It tells us that if you have a “Private room” or “Shared room”, if `minimum_nights` < 94.5, `price` should be set around 83.32 to reach an average level; if `minimum_nights` > 94.5 and the room/house is located in Brooklyn, the average `price` of popular listings is around 83.17; if `minimum_nights` > 94.5 and the room/house is not located n Brooklyn, you can even set `price` at 2780. Otherwise, if you have a “Entire home/apt”, and it is not located in Manhattan, `price` should be set around 161.60; if the house is located in Manhattan, `price` should be set around 216.90.



```
Regression tree:
tree(formula = price ~ room_type + minimum_nights + neighbourhood_group,
    data = train)
Number of terminal nodes:  5
Residual mean deviance:  33580 = 514200000 / 15310
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2730.00   -46.64   -20.32     0.00    13.36  7219.00
```

Figure 1

We then try to prune the tree based on cost complexity pruning. We perform 10-fold cross-validation to determine the size of the tree that leads to the smallest deviance. The best number of terminal nodes is 5 and the pruned tree is exactly the same as the previous tree as shown in Figure 2. The test MSE is 45952.85.

Then we seek to improve the performance of the tree by using bagging and random forests. With bagging, we can see from Figure 3 that `room_type` and `minimum_nights` seem to be the most important predictors because one of them results in the highest increase in MSE if excluded and the other results in the highest increase in node impurity if excluded. The test MSE from bagging is 46332.31.

With random forest, we set $m = 1$ and the importance result is essentially the same as shown in Figure 4 that `room_type` and `minimum_nights` are more important than `neighborhood_group`. The test MSE from random forest is 43505.16.

We have built tree models to predict `price` given `neighborhood_group`, `room_type` and `minimum_nights` using recursive binary splitting, pruning, bagging and random forests, where the tree derived from random forests gives the lowest test MSE. Since we only have three predictors in total, it is not

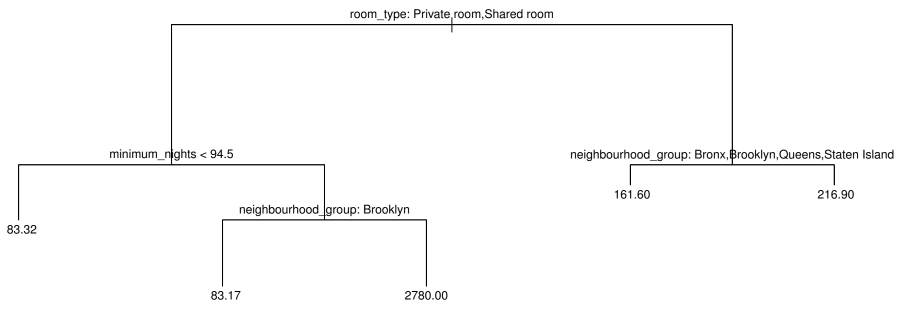

room_type: Private room,Shared room
minimum_nights < 94.5
neighbourhood_group: Bronx,Brooklyn,Queens,Staten Island
neighbourhood_group: Brooklyn
83.32
83.17
2780.00
161.60
216.90

Figure 2

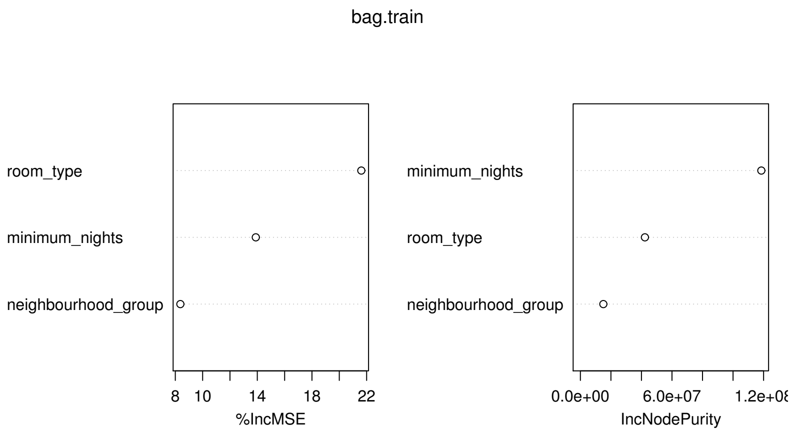


Figure 3

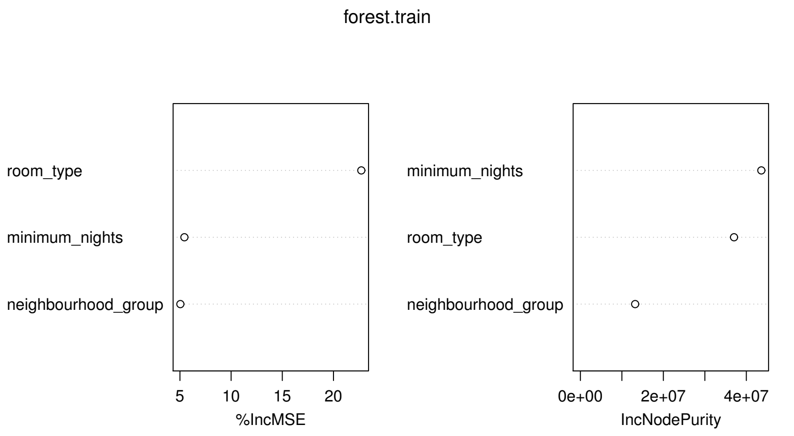


Figure 4

surprising that the pruned tree is exactly the same as the tree before pruning. The test MSE of bagging is worse than that of recursive binary splitting. This indicates the test MSE is primarily driven by bias rather than variance. It seems `neighborhood_group` is not as important as `room_type` and `minimum_nights`. And by including only one predictor in each split with the random forests, the test MSE gets improved.

The challenge would be to understand what it means for the accuracy of tree models when we have only a small number of predictors and how to interpret the high test MSE.

4 Classification Tree

From the viewpoint of a guest, the predictors we will use are `neighborhood_group`, `room_type`, `price`, `minimum_nights`, `number_of_reviews`, and `reviews_per_month`. The response will be two classes of popularity. Note that in the models "Unpopular" is the positive class while "Popular" is the negative class.

We randomly split the data into a training set (60% of the observations) and a test set (40% of the observations). Then we fit a classification tree on the training set using recursive binary splitting. The results are shown in Figure 5 and Fig 6. The tree has 3 terminal nodes. Surprisingly, the tree only uses two predictors `minimum_nights` and `reviews_per_month` but essentially only `minimum_nights`. The tree predicts that a house/room is likely to be popular if `minimum_nights` < 27.5 and unpopular otherwise.

```
Classification tree:
tree(formula = popularity ~ price + minimum_nights + number_of_reviews +
    reviews_per_month + neighbourhood_group + room_type, data = train)
Variables actually used in tree construction:
[1] "minimum_nights"    "reviews_per_month"
Number of terminal nodes:  3
Residual mean deviance:  1.179 = 34580 / 29330
Misclassification error rate: 0.3239 = 9499 / 29330
```

Figure 5

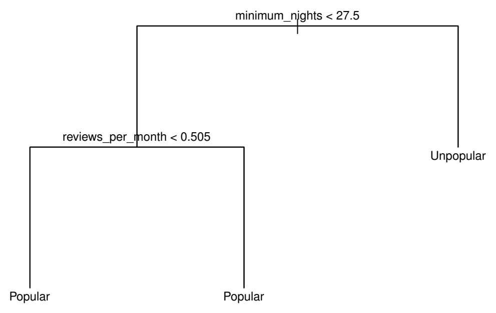


Figure 6

We then try to prune the tree based on cost complexity pruning. We perform 10-fold cross-validation to determine the size of the tree that leads to the smallest deviance. The best number of terminal nodes is 3 and the pruned tree is exactly the same as the previous tree as shown in Figure 6, which is somewhat expected since the tree before pruning is already very small. The test error rate is 0.323.

Then we seek to improve the performance of the tree by using bagging and random forests. With bagging, we can see from Figure 7 that `minimum_nights` and `reviews_per_month` result in the highest mean decrease in accuracy if excluded. `reviews_per_month` and `price` result in the highest mean decrease in Gini index followed by `number_of_reviews` and `minimum_nights`. It seems `reviews_per_month` is the most important predictor and `minimum_nights` and `price` are somewhat important. The test error rate from bagging is 0.295.

With random forest, we set $m = 3$. From Figure 8, we can see the most important predictors are still the same as that in Figure 7 with `reviews_per_month` being the most important predictor followed by `minimum_nights` and `price`. The test error rate from random forest is 0.284.

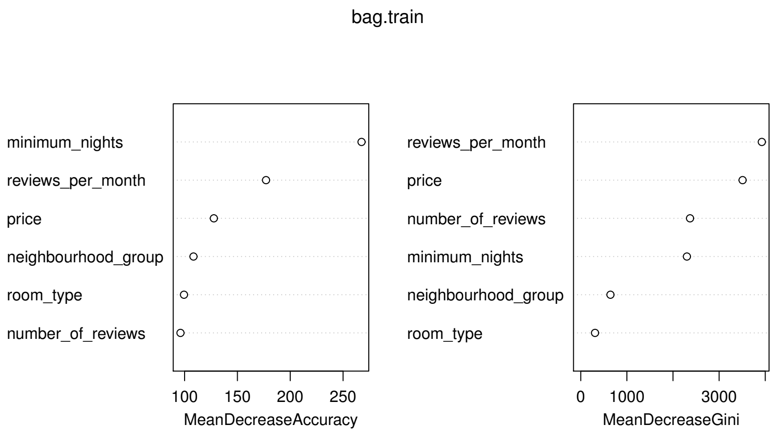


Figure 7

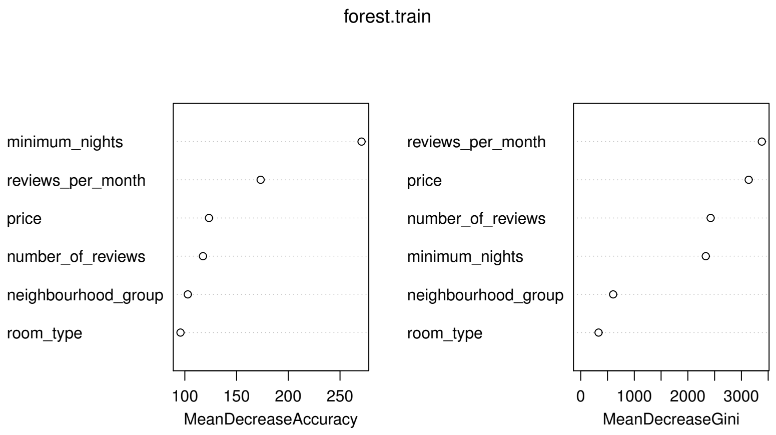


Figure 8

We have built tree models to predict the popularity of a listing given `neighborhood_group`, `room_type`, `price`, `minimum_nights`, `number_of_reviews`, and `reviews_per_month` using recursive binary splitting, pruning, bagging and random forests, where the tree derived from random forests gives the lowest test error rate. It is a little surprising that the tree generated with recursive binary splitting is quite small and only makes use of 2 predictors with only 1 predictor essentially effective. The test error rate is ∼32%, which is not great. This indicates most variables are not strong predictors of popularity. It is not very intuitive that `reviews_per_month` turns out to be the most important predictor among all the 6 variables. The test error rate gets improved with bagging, which is designed to reduce the variance. The test error rate gets even improved with random forests, which only includes 3 predictors at each split. But overall the best test error rate of ∼28% is still not great. This indicates the tree method may not be the best way to classify popularity.

Reference

[1] https://nycfuture.org/research/destination-new-york
[2] https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data