Rental Price Prediction of Airbnb

Mengdie(Celia) Ji

Preparations

1. Data source

https://www.kaggle.com/c/pricelala2. The dataset contains information on 90 variables related to the property, host, and reviews for over 35,000 Airbnb rentals in New York.

2. Read data and translate all the missing values to 'NA'.

I read the analysis data and scoring data into R session. In the meantime, the 'NA', 'N/A', ' ', are transformed as missing values to be prepared for the further cleaning.

3. Build a RMSE function

I build a RMSE to be reused multiple times later.

Exploring data

1. Cleaning the pricing value

The goal is to predict price as accurate as possible, so that the missing values in 'price' would be harmful for the prediction. I delete the rows where price is equal to 0, the total size of which is 15.

2. Examine variables

Since not all the variables are predictive, I pick out all the variables that seems would affect pricing in the common sense of business world, and examine them about variable type, the number of levels, its correlation with price, its correlation with each other and its value distribution. I use function to calculate the result or draw a boxplot to have a visual judgement.

3. Pick variables manually

I delete the variables which have too many levels (100+), have only one level, have high correlation with each other, or the value are highly skewed, and leave the variables which are not deleted for further data cleaning. The deleted variables are:
'space','notes','interaction','host_name','host_about','host_response_time','host_response_rate', 'host_acceptance_rate','host_verifications','square_feet','weekly_price','monthly_price','license', 'jurisdiction_names','id','name','security_deposit','country_code','country','minimum_minimum_nights','minimum_maximum_nights','maximum_minimum_nights','maximum_maximum_nights','minimum_nights_avg_ntm',' maximum_nights_avg_ntm','has_availability','requires_license','license','is_business_travel_ready','require_guest_phone_verification','require_guest_profile_picture','host_location','host_neighbourhood','street','neighbourhood','neighbourhood_cleansed','city','state','smart_location','calendar_updated','house_rules' ,'calculated_host_listings_count_shared_rooms','availability_30','availability_90'.

Clean data

1. Fill in the missing values

1) For Boolean type

Filled in with the largest portion of the value, ‘t’ or ‘f’. Variables: host_is_superhost, host_has_profile_pic, host_identity_verified.

2) For some numerical type

Filled in with mean values. Variables: host_listings_count (for the outliers).

3) For other numerical type

Filled in with 0, which have more sense than mean value. Variables: cleaning_fee, host_total_listings_count, reviews_per_month, host_listings_count.

4) For beds

Filled in with the bedrooms value in the same row, because the two values have high correlation (but no enough high to be deleted).

5) For the categorical variables which contains many levels but still worth a try

I replace the levels of factors from each other dataset to maintain that analysis dataset levels are exactly the same as the scoring dataset levels. Variables: property_type, cancellation_policy, market, zipcode.

2. Deal with variables containing long text

In the variables which contain the long text and contain more than 10000 levels, I count the number of characters and take the number as the transformed new variable to reflect how long is the text. The longer text is, I assume the more probability the price is high. Variables: summary, decription, neighborhood_overview, transit, access.

3. Deal with variables containing dates

In the variables which contain dates, I count the number of duration (days) from the given dates to now. I take the number of days as the transformed new variable values. Variables: host_since, first_review, last_review.

4. Deal with keywords in amenities

After looking at multiple lines in amenities, there are many keywords which could have high possibility to affect the pricing, like Wifi or TV. I use string detect function to examine whether the specific keywords occur in the text. I add a newly self-defined variable to reflect its existence or not. The keywords are: Wifi, TV, Air conditioning, Elevator, Heating, Dryer, Private, Extra.

Split data

Analysis data is split. 80% is in train and 20% is in test set.

Build model

1. The goal.

I build model with less variables and less trees with only roughly cleaned data. Without consideration of categorical variables or the meaning of text variables.

2. Build model

I intuitionally choose two variables and build a linear model. The RMSE is 82.98611. Then I add gradually all the previously selected variables to the linear model. The RMSE is 69.95377.

3. Feature selection

Best subset selection runs out of time so it is aborted.

Forward selection provides some selected variables. With these variables, the linear model constructs a RMSE of 70.55142, random forest model constructs a RMSE of 60.39637, boosting model constructs a RMSE of 80.31188.

I try Lasso to select features. With the selected features, the linear model RMSE is 70.93437, random forest model RMSE is 61.89076.

I try to use tree to select features. With the select features by tree (default cp value), the linear model RMSE is 72.79401. With the cp = 0.001, the linear model RMSE is 71.68822. Then I use cross validation to find out that the best tune of the tree is cp = 0.001.

Then I apply the selected features by tree (cp = 0.001) to random forest model, the RMSE is 62.52535. When I try to cross validation the random forest model, it runs out of time.

I apply the same features used in random forest model to build a boosting model, but the RMSE turns out high.

Tune model

1. The goal.

I go back to the data exploration and deeply clean the data and do some parameter tuning to find out the best RMSE.

2. Find best model

After the data is deeply cleaned, I build the models again to find the best model.

With all the variables, linear model RMSE is 69.3815, tree model RMSE is 73.65942, random forest model RMSE is 58.96559, boosting model RMSE is 73.78254.

3. Find best parameter

I find random forest model is lowest in RMSE, so I tune the parameters inside. The best model so far is with ntree = 800, cp = 0.01. (RMSE = 58.96559)

4. Try a faster model

I tune the boosting model since it runs more quickly than random forest model. The best performance model so far is with ntree = 1000, interaction.depth = 40,shrinkage = 0.01. The RMSE is 57.83948, which is the lowest so far.

5. Feature selection.

I use Lasso and forward selection. The features selected by each method are applied with models above. But none of them perform better than the same model with all variables.

Optimize model

1. The goal.

Based upon tuning result, I decide to use boosting model with ntree = 1000, interaction.depth = 40,shrinkage = 0.01. To optimize the result, I go back to explore the data again thoroughly and add categorical variables like ‘zipcode’ and keywords in ‘amenities’ into consideration, because the location and facilities would affect how much people would pay for a house.

2. RMSE result.

The result RMSE is 57.02161 in test set.

3. Tuning.

After I use Lasso to select features, and more keywords in the ‘amenities’, and multiple try to tune the parameters in the model, the RMSE doesn’t get better.

4. Conclusion.

As a result, the model constructing RMSE of 57.02161 in test set is my final submit. And the result ranks 40/464 in public leaderboard and 20/464 in private leaderboard.

Insights

Extreme large model and feature selection method to expect to find out all the best features will hang due to the laptop computational ability. Small model at the start would be better and build it strong step by step.
What I do right is to continuously try different models and tune the parameters. It’s kind of lucky for me to find out a good parameter combination without too much tries. Also, I plan ahead of the outline of plan of every step. This makes the process organized and stay on the right direction.