Executive Summary: Airbnb Prices in New York City

Data Science II (STAT 301-2)

Noah Holubow

15 March 2021

Contents

Overview

The goal of this project is to predict the expected nightly price of an Airbnb rental in New York City. Airbnb is a vacation rental marketplace based in San Francisco, California. I am primarily investigating the relationship between nightly price and a host of other factors, enumerated below. The dataset is confined to New York City and its five boroughs: Manhattan, Queens, Staten Island, Brooklyn, and The Bronx in the year 2019.

There are 48,895 observations. The dataset provides a good mix of categorical and numerical data. Columns of interest include NYC borough, room type, minimum number of nights, number of reviews, reviews per month, host listings, yearly availability. These are the regressors. The outcome variable is then, of course, the nightly rate.

Splitting & feature engineering

The data was first log-transformed to eliminate skewness and then split with a proportion of 0.75 and stratified on price. Repeated v-fold cross-validation was used in order to appropriately train the models on the training set.

Recipe

I created two recipes: one for the linear model and one for the machine learning models. The only difference is really in the creation of the dummy variables. In my recipe, I linearly imputed missing values, created dummy variables, and normalized the data.

Models

Four separate models were defined: linear, random forest, boosted trees, and k-nearest neighbors.

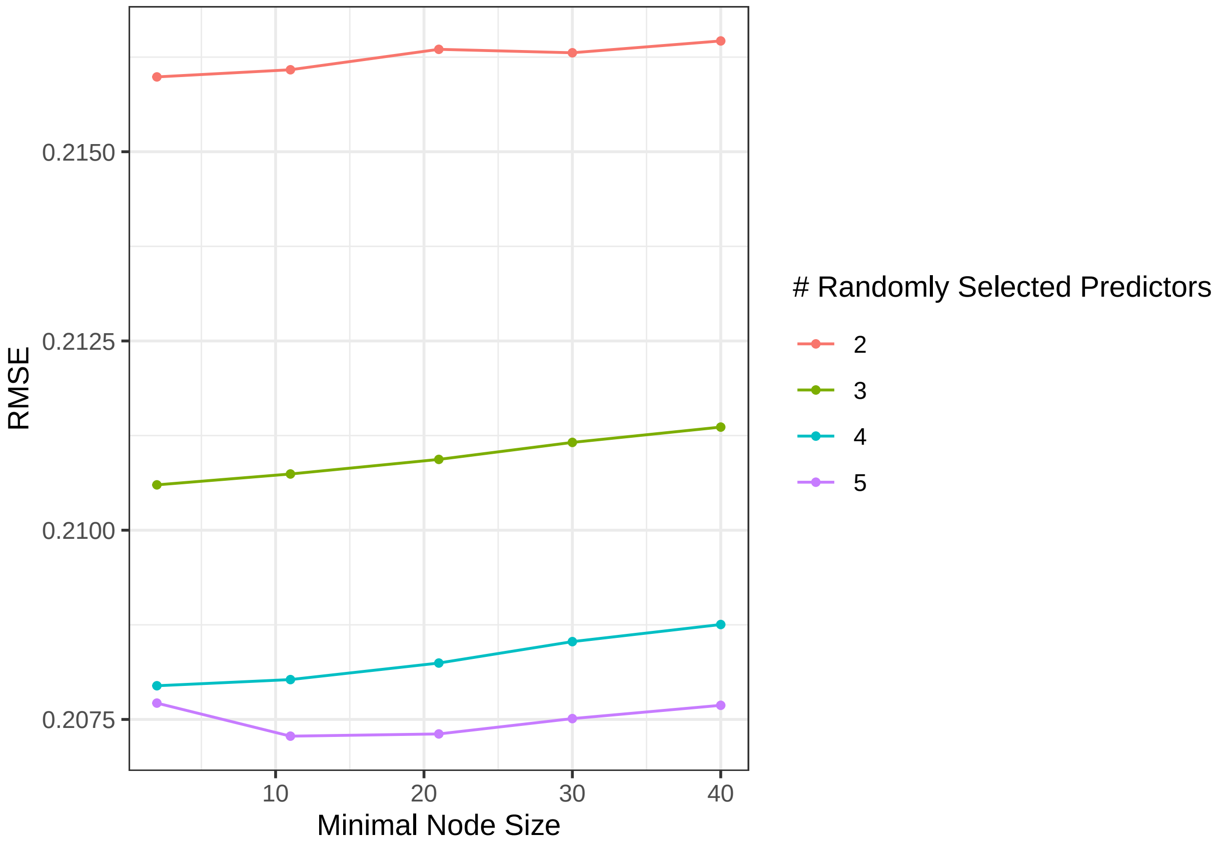
Inspecting tuned info

Once the models were complete, the winning model was selected:

model_type	mtry	min_n	.estimator	mean	std_err	neighbors	learn_rate
rf	5	11	standard	0.2072788	0.0010548	NA	NA
rf	5	21	standard	0.2073080	0.0010415	NA	NA
rf	5	30	standard	0.2075107	0.0010352	NA	NA
rf	5	40	standard	0.2076862	0.0010312	NA	NA
rf	5	2	standard	0.2077156	0.0010377	NA	NA
rf	4	2	standard	0.2079448	0.0010413	NA	NA

As we can see, the random forest (`rf`) model with an `mtry` of 5 and a `min_n` of 11 performs the best, with a slightly smaller RMSE than the next random forest model with a `min_n` of 21. The `neighbors` and `learn_rate` columns have a value of `NA` because these values are only applicable to the KNN models. The lower the value of RMSE, the better.

This is confirmed by the random forest model's tuning plot:



Fitting training & testing sets

Now that we have chosen our winning model (random forest with `mtry` of 5 `min_n` of 11), we fit the data on the entire training set. The RMSE is 0.2066715 and the r-squared is 0.538. This is fantastic because the entire training set has a lower RMSE and higher r-squared than any of the cross-validated v-folds.

Fitting on testing set

When we fit our testing set on the winning model, get the following results: an RMSE of 0.2043358 and an r-squared of 0.538. Again, this is really great. The RMSE is significantly lower than that of the entire training set, and the r-squared is still just as good as before. This means that the model, while it may still have overfit in some ways, did not do a terrible job of running itself on new data. In fact, it appears to have done better.

Conclusion & next steps

The random forest model with `mtry` of 5 and `min_n` of 11 appears to be our winning model. This is great because it means the tuning parameters were well-chosen by the machine. It is important to realize that this model is by no means close to perfect. While the log-transformation done initially was certainly helpful, it did not take away all the skewness or outliers, so this certainly contributes to no model being perfect. Overall, I am pleased with the outcome of this project. The results are somewhat puzzling, especially in the tuning plots for the random forest and boosted trees models, where the RMSE curve slopes up as the minimal node size increases. That being said, the random forest did perform the best and had a decent RMSE and r-squared, so this was a good sign.