https://github.com/pilipentseva/predictions

Goal of the project: Predict prices for small and medium apartments in San Diego

Data set: for my project I used a data set from Airbnb Inside. Initial dataset has 9673 (almost 10 K) observations and is collected for the data 25th of September 2021.

Data cleaning and transformations: Firstly, I got rid of the columns which didn't have any meaning for my analysis. Secondly, I deleted the columns which repeated information listed in another column or stating the obvious information. For example it was the case for the neighborhood columns. I could see that the neighborhood column has a lot of missing data (almost 1/3 of the observations). Looking more carefully into the data I could see that neighborhood information is much better depicted in neighborhood cleansed (and it has no missing values). Thus, I dropped the variable neighbourhood. Host neighborhood sometimes also had missing values and looked very similar to neighbourhood cleansed, but with more missing values for reasons I couldn't identify. I decided to only keep the neighborhood cleansed. The column with the number of bathrooms as a number variable was missing, but I extracted information about the number of bathrooms from the column which had it in text format. Prices and percentage columns contained symbols which I didn't need for my analysis - I got rid of them. I encoded variables containing true and false (t, f) to dummy variables 1, 0.

After this I looked at data types and transformed those who belonged to a mistaken data type. I also evaluated the missing values. Depending on the logic and structure of the data, I took decisions on how to replace missing values. For some observations I replace them with 0, for others with mean or mode. I checked duplicate rows and found none. I created several important variables. Firstly, I calculated the distance from the city center to the location of the airbnb. I could do it using coordinates of the city center of San Diego and given coordinates of airbnb locations. Amenities and host verifications had many variables encoded in one cell. For each amenity and host verification method I created a dummy variable. I took a decision not to group amenities in groups since I believed that I didn't have a field expertise for this and didn't know which ones are more or less important. I also was afraid to make a mistake to combine them in the wrong groups based on the wrong understanding of the field. It turned out to be the decision with drawbacks since later with a higher number of variables it took me much more time to estimate the models. Combining amenities in groups and dropping less frequent ones would have simplified this process. There are also amenities with strange symbols in the beginning. Even after cleaning amenities I left them as separate ones since I consider these symbols to be specifications.

I worked with available time variables and created a variable for host which referred to the duration a person has been a host calculated on the 25th of September 2021 date. Moreover, I created a variable determining the overall number of amenities for each of the accommodations, since it also may be an important variable.

Last but not least, I encoded the categorical variables. Many machine learning libraries, including the ones I use, can't work with categorical variables. One way to solve this issue is to implement the encoder which will code categorical variables as numerical ones, which will allow the libraries to perform this analysis. I proceeded by using one of the available encoders.

Subsampling: Since I am only interested in small and medium apartments, I only considered places that could accommodate less than 8 people. I also subsampled it to the type of room: Entire home/ apartment, which should reflect my targeted types of apartment.

Model building: I built 3 models. For all the models I split the data into X and y datasets. y contains the target variable - the price of apartment. X contains all other variables which I want to use as predictors. I split the both data sets into train and test with the test being 20%.

The first one is implementing the Random Forest method. For a random forest I suggest several parameters and ask algorithms to choose the best ones. Then I estimate the model with these best parameters. The second model is LASSO. For LASSO I suggest several tuning parameters and ask the algorithm to choose the best one, then I estimate the model with the best tuning (in this case it is alpha = 0.01). The last model is the simple Linear Regression model.

Models evaluation: To evaluate the models, I compare RMSE which I got on test data.

Training Random Forest - 70.625
Training RMSE LASSO - 25.205
Training RMSE Linear Regression - 0.0003

Test Random Forest - 85.054
Test RMSE LASSO - 81.095
Test RMSE Linear Regression - 86.1316

In terms of computational time, performing RF and LASSO took more time than performing Linear Regression. For the test data set the lowest RMSE belongs to LASSO. The second lowest is Random Forest and then goes Linear Regression. However, I can see that differences in RMSE between training and test data sets are very high for LASSO and Linear Regression. It suggests that in this case I overfit the data quite significantly. Also, although the difference in test RMSE between RF and LASSO exists, it is not that big. Thus, I choose Random Forest to be the best model in this case.

T.