Assignment 2: Final Report

Timur Naushirvanov

10/24/2021

Introduction

In this report, I am presenting the results of my analysis of a market of small and mid-size appartments, which can accomodate 2-6 people, in Paris. The **dependent variable** is a natural logarithm of appartments' prices. After doing some feature engineering, I am building random forest models and comparing the best one with other types of algorithms, including LASSO, CART and GBM. The results show that the hand-tuned random forest with all relevant predictors is the second best models in terms of RMSE minimisation (only GBM showed slightly better results).[1]

Sample Preparation

Initial dataset contained 50133 observations and 74 different variables. According to the task, I need to focus only on small and mid-sized appartements which can accomodate 2-6 people. In case of Paris, there were not very clear descriptions of different appartments types. Therefore, I focused on a room type category "Entire home/apt", and then deleted from a property types categories such as rooms, cottages, houses, villas, islands, boats, caves, etc. After saving only the appartments with less than 4 bedrooms and which accommodate 2-6 people, I have got 28878 observations.

Label and Feature Engineering

I have chosen to focus on natural logarythm of prices due to a very strong skewedness of the original price variable, while the distribution of a ln price was quite normal (even though it is not so important for random forest, other algorithms can be susceptible to this difference).

In terms of relevant predictors for analysis, I focused on the following variables: n_accommodates, f_beds, n_bathrooms, b_superhost, n_minimum_nights, n_availability_60, f_neighbourhood_clean, n_review_scores_rating, flag_review_rating, n_review_scores_location, flag_review_location, n_reviews_per_month, flag_reviews_per_month, n_days_since_first, flag_days_since_first, n_days_since_last, flag_days_since_last, b_wifi, b_tv, b_coffee, b_netflix, b_breakfast, b_freepark, b_cleanprod, b_bedlinens, b_workspace.

I decided to transform n_beds variable into factor because of a huge disproportionality between different numbers (less or equal to 1 bed, 2 beds, 3 beds, more or equal to 4 beds). I also decided to choose only 11 dummies from the amenities variable (which originally contained more than 1200 unique categories), because they were quite widespread in the data and can affect prices. I created additionally two variables: the number of days since the first and the last review. Moreover, I also had to ocasionally transform the variables' types and delete unnecessary signs (like $).

While dealing with missings, I imputed them with the median if their number was very small (n_bathrooms and n_beds), and also imputed them with the median if their number was relatively huge (several thousands) but also flagged these variables (n_reviews_per_month, n_days_since_first, n_days_since_last, n_review_scores_rating, n_review_scores_location). One predictor I had to delete, because more than 50% of observations were missing (n_host_acceptance_rate). I also created several interactions of variables and squares for the LASSO algorithm.

[1] Additional Technical Report and code can be accessed through this link.

Model Building

I specified 4 models (the last one is for LASSO only):

- Model 1: lnprice ~ n_accommodates + f_beds + n_bathrooms + b_superhost + n_minimum_nights + n_availability_60 + f_neighbourhood_clean − only *basic variables*
- Model 2: lnprice ~ *basic variables* + n_review_scores_rating + flag_review_rating + n_review_scores_location + flag_review_location + n_reviews_per_month + flag_reviews_per_month + n_days_since_first + flag_days_since_first + n_days_since_last + flag_days_since_last − *basic variables* + largely *imputed variables*
- Model 3: lnprice ~ *basic variables* + *imputed variables* + b_wifi + b_tv + b_coffee + b_netflix + b_breakfast + b_freepark + b_cleanprod + b_bedlinens + b_workspace − *basic variables* + *imputed variables* + *selected amenities*
- Model 4 (for LASSO): lnprice ~ *basic variables* + *imputed variables* + *selected amenities* + *squares* + *interactions*

70% of the dataset were randomly selected as a train data, 30% - as a holdout data. For building random trees, I also allowed the algorithm to choose an appropriate tuning automatically (Model 3 auto). It looks like model 3 has the best performance (the lowest RMSE), and hand-tuned model 3 works even better than autotuned. Therefore, I will focus further on the hand-tuned model 3.

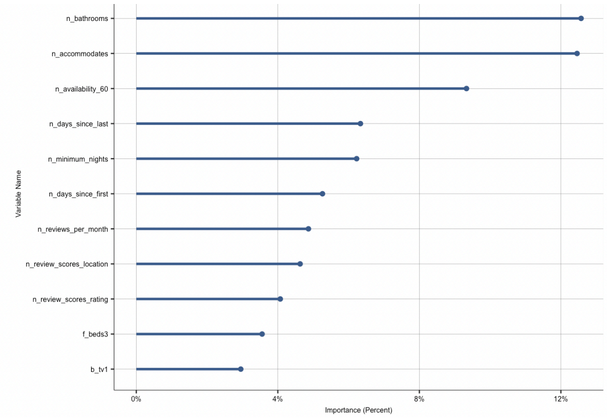
Model	Mean RMSE
Model 1	0.4106
Model 2	0.4018
Model 3	0.3979
Model 3 auto	0.4019

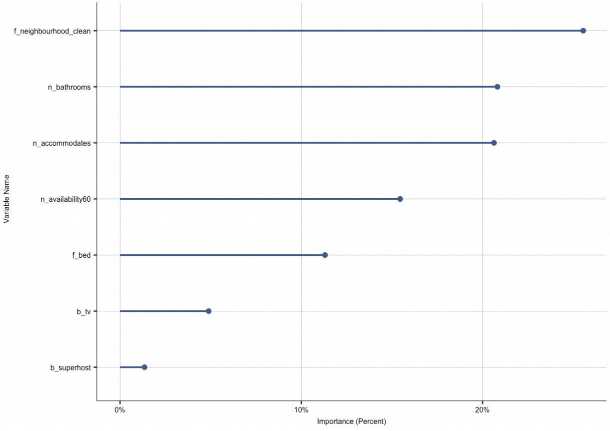
Model Performance and Diagnostics

The following table helps to see that the lowest RMSE is associated with 5 observations in the terminal nodes and 8 variables to consider at each split. However, it also should be noted that the differences in RMSE are very small - 0.0014 log price (1/280 of the mean RMSE values). It means that the random forest tends to be robust.

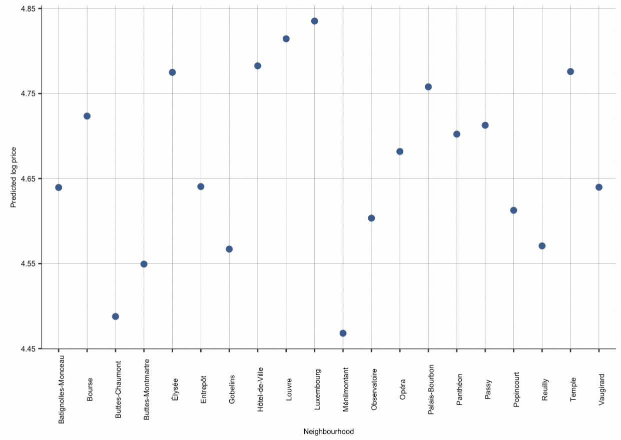
nodes	8	10	12
5	0.3979	0.3986	0.3993
10	0.3981	0.3985	0.3989
15	0.3985	0.3987	0.3988

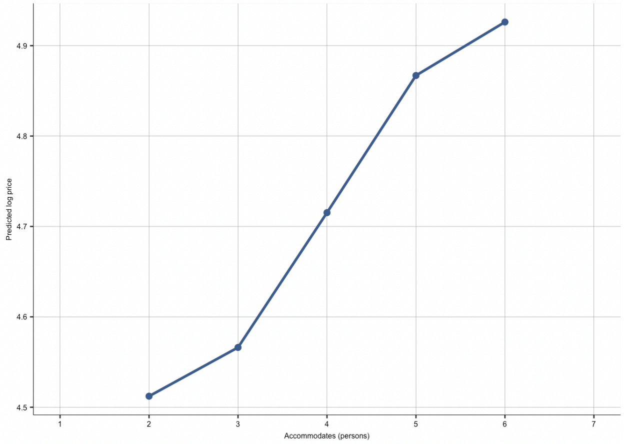
The following graphs show the variable importance plots. At cutoff point = 0.15, 11 variables are important for predicting ln price, and the largest three are the number of bathrooms, the number of accomodates, and the number of days when an appartment was available during the last 60 days. The second plot shows the importance of combined factors, and the situation has changed: neigbourhood is the most important variable, then go availability during the last 60 days, numbers of bathrooms and accomodates, then - factor of beds.



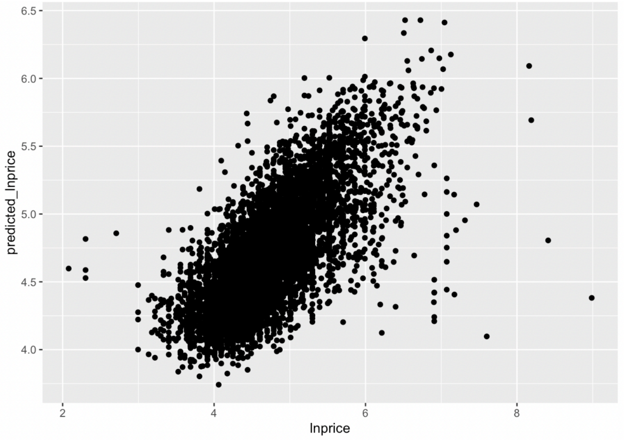


The following two figures demonstrate partial dependence plots for variables f_neighborhood and n_accomodates (more can be found in an additional document).





Finally, there is a scatter plot comparing predicted ln prices with the actual ones. I would say that the prediction results are comparatively well, even though there are some quite noticeable extreme values.



Comparing Performances

Finally, let's compare the random forest modal performance with other algorithms. All models are used with a 5-fold cross validation. The results from the two tables below suggest that the best model is GBM with basic tuning, and we should choose it for the predictions; however, random forest 3 is the best among all other remaining models (and it is easier to be implemented compared to GBM).

Table 1:

	CV RMSE
OLS	0.408
LASSO (model w/ interactions)	0.406
CART	0.454
Random forest 1	0.411
Random forest 2	0.402
Random forest 3	0.398
GBM (basic tuning)	0.393

Table 2:

	Holdout RMSE
OLS	0.411
LASSO (model w/ interactions)	0.410
CART	0.460
Random forest 1	0.415
Random forest 2	0.406
Random forest 3	0.402
GBM (basic tuning)	0.396