Price prediction of apartments in Barcelona

Viktória Kónya

10 February 2022

Introduction

The purpose of this report is to present the predictive models for Airbnb apartment pricing focusing on apartments for rent in Barcelona. The report gives a methodological overview of the model building steps and compares the performance of the models used for price prediction. The documentation consists of four main sections: part 1 presents the business problem and sample design, part 2 summarizes the descriptive statistics of the predictors, part 3 presents the different model specifications and finally, part 4 compares the predictive performance of the models and concludes about the most accurate specification.

Business problem

The client company operates small and medium size apartments in Barcelona hosting 2 to 6 guests. As a new entrant to the short-term rental market, the client wants to use a price prediction model which can help to set the prices of the its newly listed apartments on the Airbnb website. Using openly available data about the existing Airbnb listings our task is to provide the most accurate pricing model for our client.

Data source and sample design

Data of existing apartment listings in Barcelona was collected from the Airbnb website with web scraping for the 7th December 2021 December date. The below points summarize the sample creation steps in order to fit the dataset to the business problem.

- Our client operates with only entire home rentals hence hotel room, shared and private room rentals were excluded from the sample.
- Irrelevant property types were excluded such as boat, barn, villa, camper.
- Apartments suitable for 2-6 guests and with less than 6 individual beds were considered only.
- Apartments without bathroom were excluded from the sample.
- Apartments with more than 90 days minimum rent days were excluded as our client operates with short-time apartment rents.
- Apartments with price per night above 600 USD were excluded from the sample as this pricing is out of the scope of our client.

The final dataset used for the model building contained 7139 listings.

Predictors of hotel prices

For the price prediction it was assumed that three main components contribute to the variability of the apartment prices: the characteristics of the apartments (number of beds, bathrooms, accommodates, location from the city center, type of the property, neighborhood, minimum number of guest nights, number of amenities in the listing), their review information (review score, number of reviews) and the amenities in the listings that might attract the attention of the guests. Host related information was excluded from the analysis as it is not relevant for the pricing of a new entrant. The full list of the variables with their description can be found in **A1.**

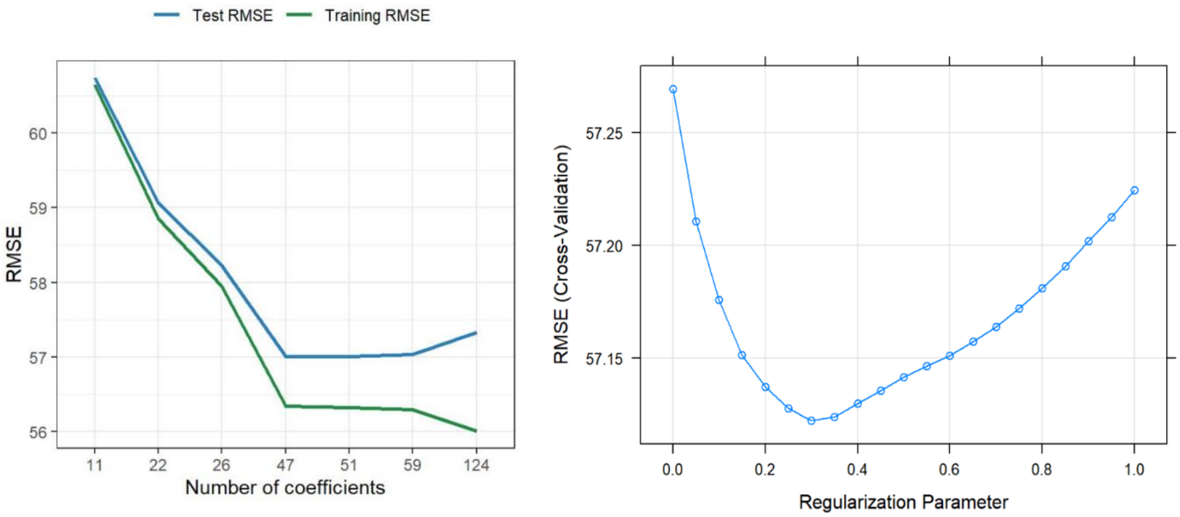
Modelling

For the modelling, train and holdout sets were created by randomly splitting the dataset into two partitions. 70% of the listings (4999 observations) were added to the train set and 30% of the listings (2140 observations) were kept for the holdout set. In case of all the presented models the modelling was done on the train set and the performance of the selected models were evaluated on the holdout set. In order to select the model with the best performance four modelling methods was compared: OLS regression with 7 different specifications, LASSO estimate, the Decision tree and Random forest algorithms. The list of variables used in each model estimation can be found in **A2.**

OLS

As a starting point, basic OLS regressions with different specifications were used to predict the apartment prices. Starting from the simplest model specification using only the basic characteristics of the apartments more and more

Fig1. A. OLS model selection, B. LASSO regularization



predictors were added to each model (including interaction terms) in order to improve its predictive power. The RMSE was calculated with 5-fold cross validation to get the proper estimate of the measure. **A3.** summarizes the variable groups used in each model. Different model specifications were compared based on the test RMSE from the cross-validation. **Fig1. A.** shows the relationship between the model complexity and the train and test RMSE across OLS model versions. The graph illustrates well, that as we move towards more complex models with interaction terms and additional predictors, the train RMSE indicates better fit, however the test RMSE starts to increase when we add more interaction terms to the model. Comparing the models on the test RMSE suggests that Model 4 with 47 predictors could be used for comparison with other methods. Details of the goodness of fit measures can be found in **A4.**

Estimation with LASSO

The main drawback of the OLS regression is that we are unable to estimate and compare each possible combination of the available predictors in order to choose the best model. For variable selection the LASSO estimate provides a powerful alternative. The biggest advantage of LASSO is that it provides automatic variable selection by shrinking the coefficients of noisy predictors towards zero. For the input predictor list, the predictors of the most complex Model 7 were used which includes the interaction terms. Similarly to the OLS, 5-fold cross-validation was used for the RMSE calculation.

Fig1. B. shows the relationship between the regularization parameter and the RMSE. The lowest RMSE estimate indicates that the optimal shrinkage parameter is 0.35 where the model selected 73 predictors with non-zero parameters out of the total 121 variables.

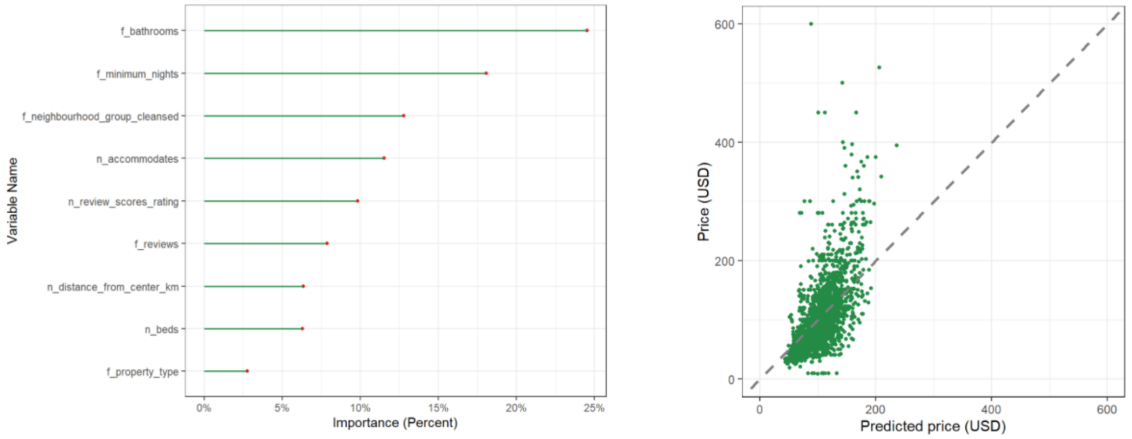
Decision tree with pruning

LASSO seems to be a reasonable choice if we do not want to compare dozens of different model specifications. However, the non-linearities added to the model (interaction terms, polynomials) are still need to be defined in advance. The next prediction model is based on the Decision tree method which can capture these patterns in the data automatically. For the Decision tree model, the variable list of the broadest model without interactions (Model 4) were used and again, 5-fold cross validation was set as train control. The main concern with the binary splitting algorithms is that the stopping rule, which determines the complexity of the final model, needs to be defined in advance if we want to build the tree in one step. Instead of defining an arbitrary stopping we can also create our model by building a very complex tree and then remove branches of the tree that do not provide discriminatory power to subset the apartments. In my analysis I followed the latter approach and started by building a very big, overfitted tree and then removed the unimportant branches. The tree of the pruned model can be found in **A5.**

Random Forest

The decision tree is a good starting point but the main concern with the it is that the method has high reliance on the selected variables and choice of the tuning parameters: too strict criterion leads to poor prediction while too lenient stopping rules can easily result in an overfitted tree. Ensemble methods provide a solution for this problem by estimating multiple trees that are different from each other and take their average. In the next estimation I am going to use the Random Forest algorithm with

Fig1. A. Variable importance plot, B. Actual and predicted prices



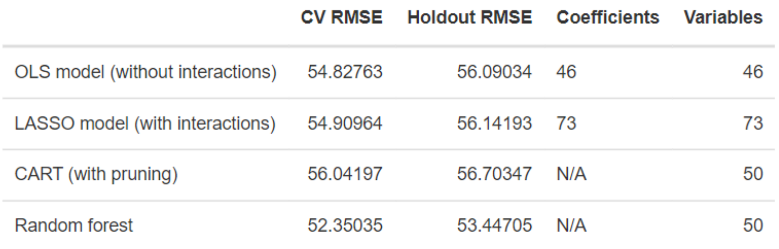
the following settings: we are going to build 500 models, for each model we will only use 8 of the available predictors which will ensure that the models are going to be different and as stopping criteria I will restrict the minimum number of observations predictors have the most significant contribution to the price prediction model, the dummies created from the factor variables were grouped together. **Fig2. A.** ranks the variables by their importance. As we can see, all the basic features of the apartments are among the most influential predictors.

Before we compare our findings, we should take a look at how the predictions from the model compare to the actual apartment prices. **Fig2. B.** shows the y-$\hat{y}$ plot of the model predictions. As we can see, the model prediction seems fairly accurate for apartments with prices below 200 USD. However, for pricy apartments the model highly under predicts the prices which can be due to some special features of these apartments that were not included in the model.

Model comparison and summary

In order to conclude about our prediction model, let's compare their performance measures. The next table lists the cross validated RMSE, the RMSE calculated for the holdout set and the number of predictors included in each model.

Table1. Model comparison



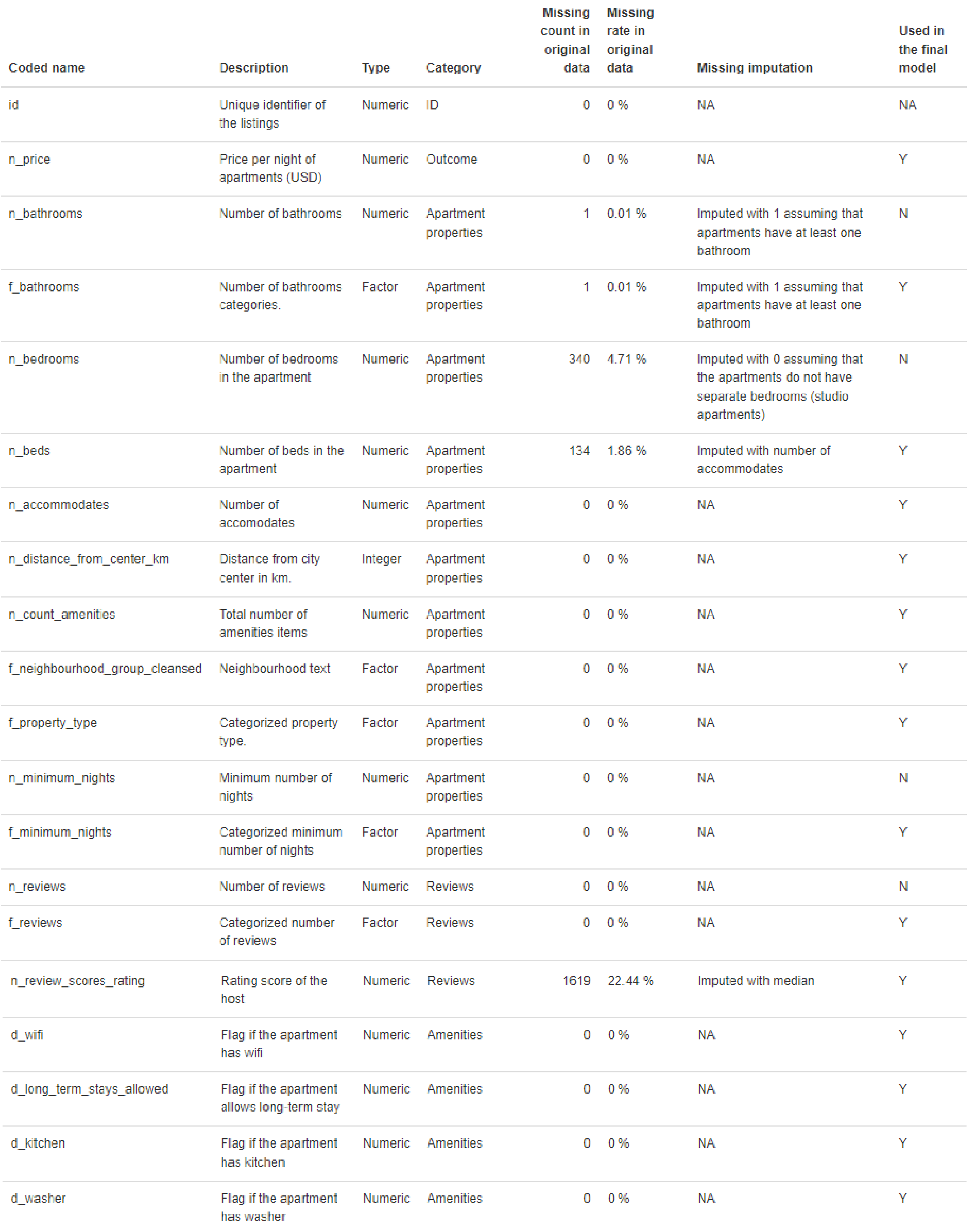
	CV RMSE	Holdout RMSE	Coefficients	Variables
OLS model (without interactions)	54.82763	56.09034	46	46
LASSO model (with interactions)	54.90964	56.14193	73	73
CART (with pruning)	56.04197	56.70347	N/A	50
Random forest	52.35035	53.44705	N/A	50

The cross-validated RMSE is the highest for the CART model which makes an average error of about 56 USD in the price prediction. In contrast, the error of the Random forest model is about 4 dollars lower in the training sample. In this case, the conclusion based on solely the cross-validated RMSE would favor the Random forest model. If we look at the holdout RMSE values, the figures indicate that the performance of each model is expected to be somewhat worse on the live data and this drop in the performance is the most pronounced in the OLS and LASSO models. Note that the Random forest model outperforms all the other methods on both the train and holdout samples hence from modelling perspective the reasonable choice would be to use the Random forest model for price prediction.

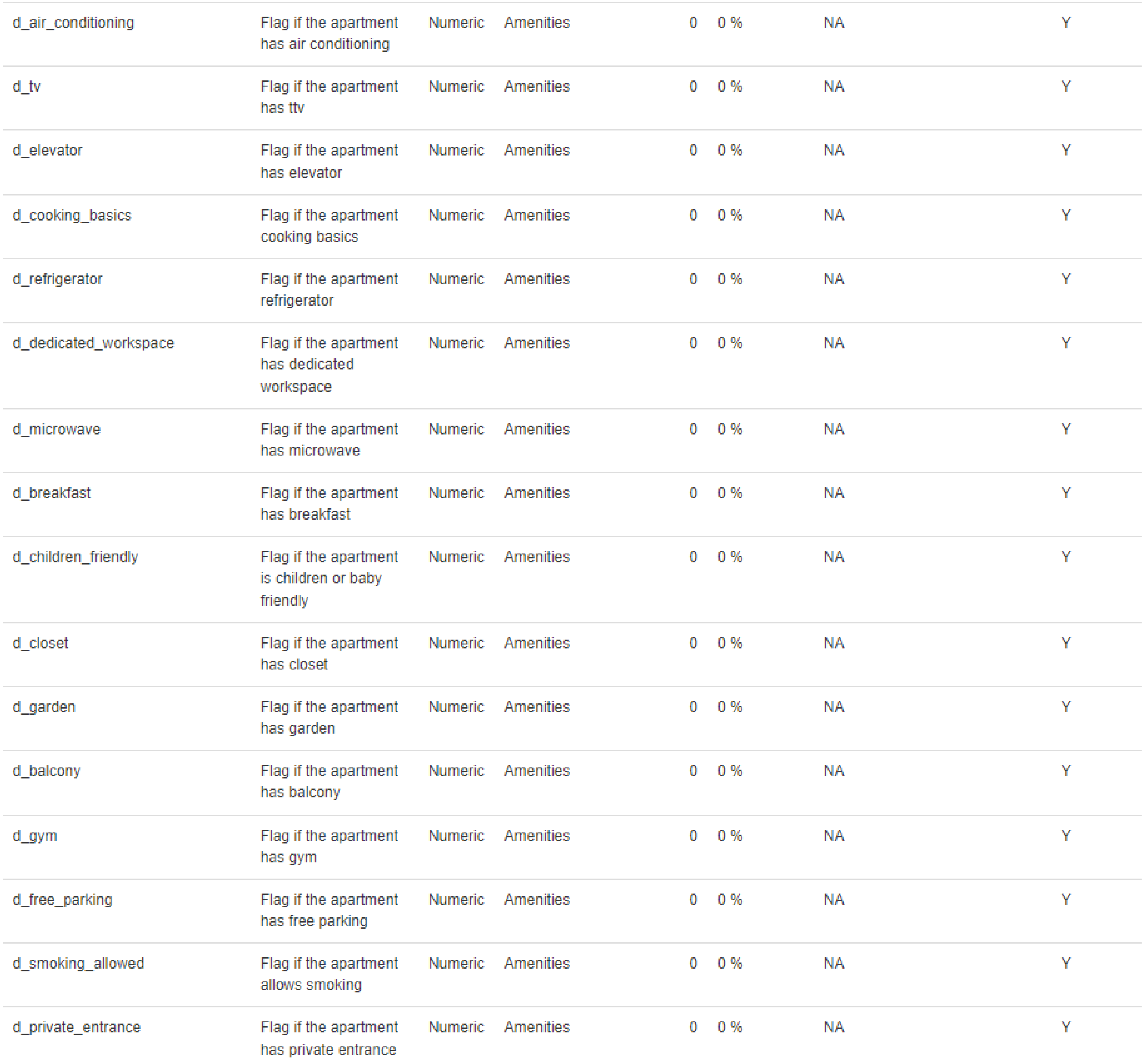
If we look at the choice between models from business perspective, interpretability is a crucial question. We could see that the Random forest model favored the basic properties of the apartments which can be easily included in an OLS framework. The OLS model has the advantage that it is able to answer questions such as how much higher prices we can set if we have an apartment 1 km closer to the city center. In this case the price is that the average error of the prediction is roughly 2 dollars higher. If the interpretability has key importance then moving to the simplest OLS model could also be a rational choice.

Appendix

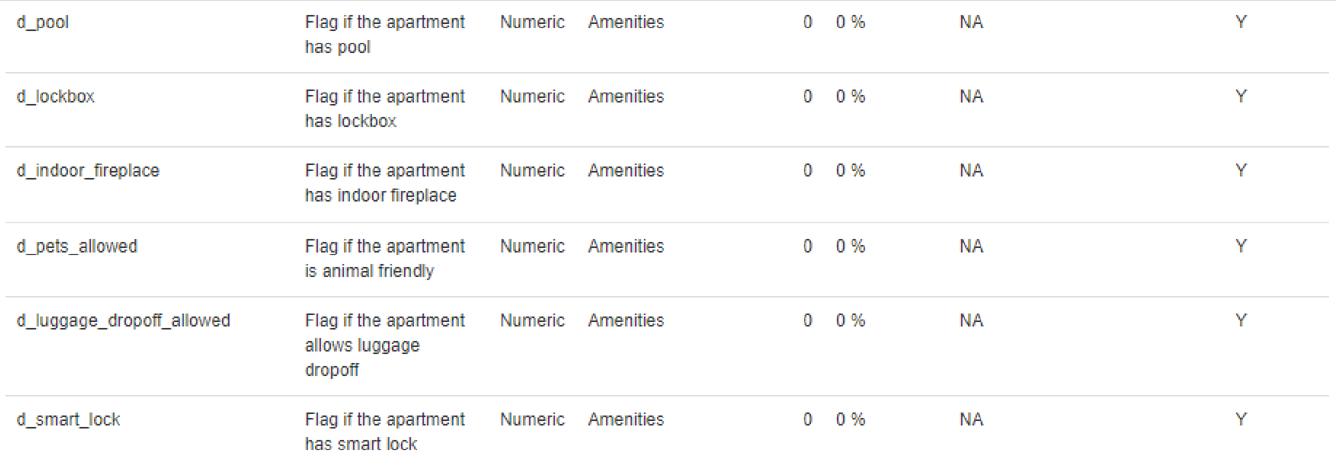
A1. Variable list



Coded name	Description	Type	Category	Missing count in original data	Missing rate in original data	Missing imputation	Used in the final model
id	Unique identifier of the listings	Numeric	ID	0	0 %	NA	NA
n_price	Price per night of apartments (USD)	Numeric	Outcome	0	0 %	NA	Y
n_bathrooms	Number of bathrooms	Numeric	Apartment properties	1	0.01 %	Imputed with 1 assuming that apartments have at least one bathroom	N
f_bathrooms	Number of bathrooms categories.	Factor	Apartment properties	1	0.01 %	Imputed with 1 assuming that apartments have at least one bathroom	Y
n_bedrooms	Number of bedrooms in the apartment	Numeric	Apartment properties	340	4.71 %	Imputed with 0 assuming that the apartments do not have separate bedrooms (studio apartments)	N
n_beds	Number of beds in the apartment	Numeric	Apartment properties	134	1.86 %	Imputed with number of accommodates	Y
n_accommodates	Number of accomodates	Numeric	Apartment properties	0	0 %	NA	Y
n_distance_from_center_km	Distance from city center in km.	Integer	Apartment properties	0	0 %	NA	Y
n_count_amenities	Total number of amenities items	Numeric	Apartment properties	0	0 %	NA	Y
f_neighbourhood_group_cleansed	Neighbourhood text	Factor	Apartment properties	0	0 %	NA	Y
f_property_type	Categorized property type.	Factor	Apartment properties	0	0 %	NA	Y
n_minimum_nights	Minimum number of nights	Numeric	Apartment properties	0	0 %	NA	N
f_minimum_nights	Categorized minimum number of nights	Factor	Apartment properties	0	0 %	NA	Y
n_reviews	Number of reviews	Numeric	Reviews	0	0 %	NA	N
f_reviews	Categorized number of reviews	Factor	Reviews	0	0 %	NA	Y
n_review_scores_rating	Rating score of the host	Numeric	Reviews	1619	22.44 %	Imputed with median	Y
d_wifi	Flag if the apartment has wifi	Numeric	Amenities	0	0 %	NA	Y
d_long_term_stays_allowed	Flag if the apartment allows long-term stay	Numeric	Amenities	0	0 %	NA	Y
d_kitchen	Flag if the apartment has kitchen	Numeric	Amenities	0	0 %	NA	Y
d_washer	Flag if the apartment has washer	Numeric	Amenities	0	0 %	NA	Y

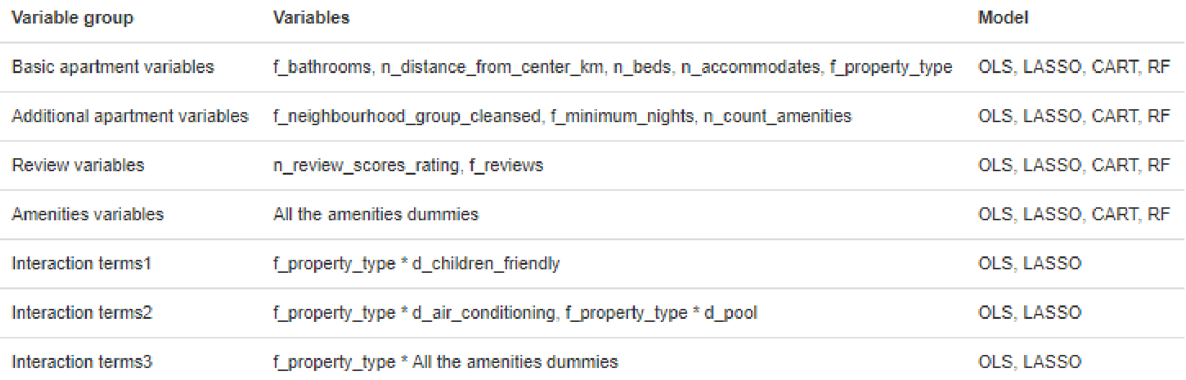


d_air_conditioning	Flag if the apartment has air conditioning	Numeric	Amenities	0	0 %	NA	Y
d_tv	Flag if the apartment has ttv	Numeric	Amenities	0	0 %	NA	Y
d_elevator	Flag if the apartment has elevator	Numeric	Amenities	0	0 %	NA	Y
d_cooking_basics	Flag if the apartment cooking basics	Numeric	Amenities	0	0 %	NA	Y
d_refrigerator	Flag if the apartment refrigerator	Numeric	Amenities	0	0 %	NA	Y
d_dedicated_workspace	Flag if the apartment has dedicated workspace	Numeric	Amenities	0	0 %	NA	Y
d_microwave	Flag if the apartment has microwave	Numeric	Amenities	0	0 %	NA	Y
d_breakfast	Flag if the apartment has breakfast	Numeric	Amenities	0	0 %	NA	Y
d_children_friendly	Flag if the apartment is children or baby friendly	Numeric	Amenities	0	0 %	NA	Y
d_closet	Flag if the apartment has closet	Numeric	Amenities	0	0 %	NA	Y
d_garden	Flag if the apartment has garden	Numeric	Amenities	0	0 %	NA	Y
d_balcony	Flag if the apartment has balcony	Numeric	Amenities	0	0 %	NA	Y
d_gym	Flag if the apartment has gym	Numeric	Amenities	0	0 %	NA	Y
d_free_parking	Flag if the apartment has free parking	Numeric	Amenities	0	0 %	NA	Y
d_smoking_allowed	Flag if the apartment allows smoking	Numeric	Amenities	0	0 %	NA	Y
d_private_entrance	Flag if the apartment has private entrance	Numeric	Amenities	0	0 %	NA	Y



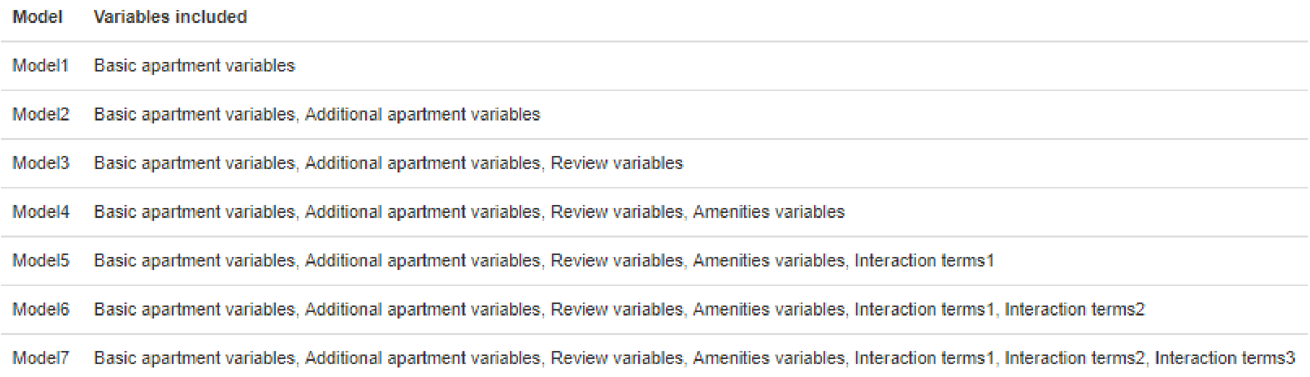
d_pool	Flag if the apartment has pool	Numeric	Amenities	0	0 %	NA	Y
d_lockbox	Flag if the apartment has lockbox	Numeric	Amenities	0	0 %	NA	Y
d_indoor_fireplace	Flag if the apartment has indoor fireplace	Numeric	Amenities	0	0 %	NA	Y
d_pets_allowed	Flag if the apartment is animal friendly	Numeric	Amenities	0	0 %	NA	Y
d_luggage_dropoff_allowed	Flag if the apartment allows luggage dropoff	Numeric	Amenities	0	0 %	NA	Y
d_smart_lock	Flag if the apartment has smart lock	Numeric	Amenities	0	0 %	NA	Y

A2. Variable groups



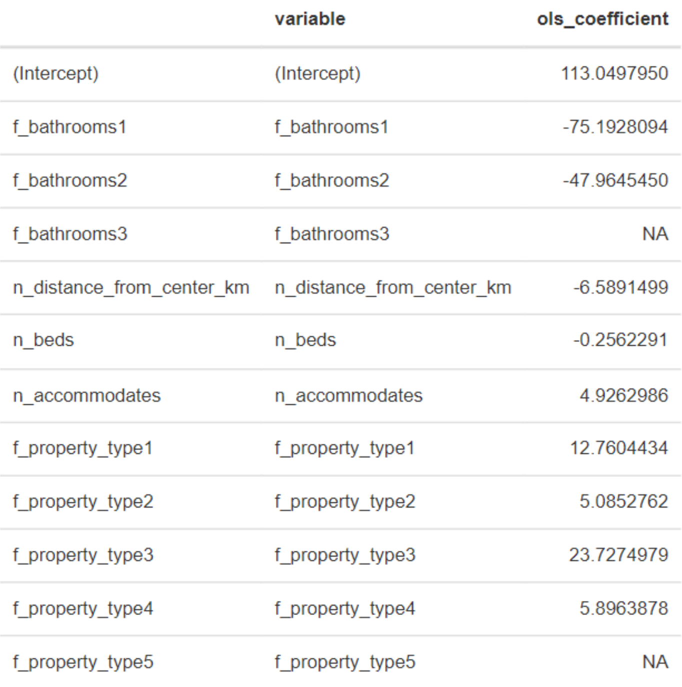
Variable group	Variables	Model
Basic apartment variables	f_bathrooms, n_distance_from_center_km, n_beds, n_accommodates, f_property_type	OLS, LASSO, CART, RF
Additional apartment variables	f_neighbourhood_group_cleansed, f_minimum_nights, n_count_amenities	OLS, LASSO, CART, RF
Review variables	n_review_scores_rating, f_reviews	OLS, LASSO, CART, RF
Amenities variables	All the amenities dummies	OLS, LASSO, CART, RF
Interaction terms1	f_property_type * d_children_friendly	OLS, LASSO
Interaction terms2	f_property_type * d_air_conditioning, f_property_type * d_pool	OLS, LASSO
Interaction terms3	f_property_type * All the amenities dummies	OLS, LASSO

A3. Variable list of OLS models



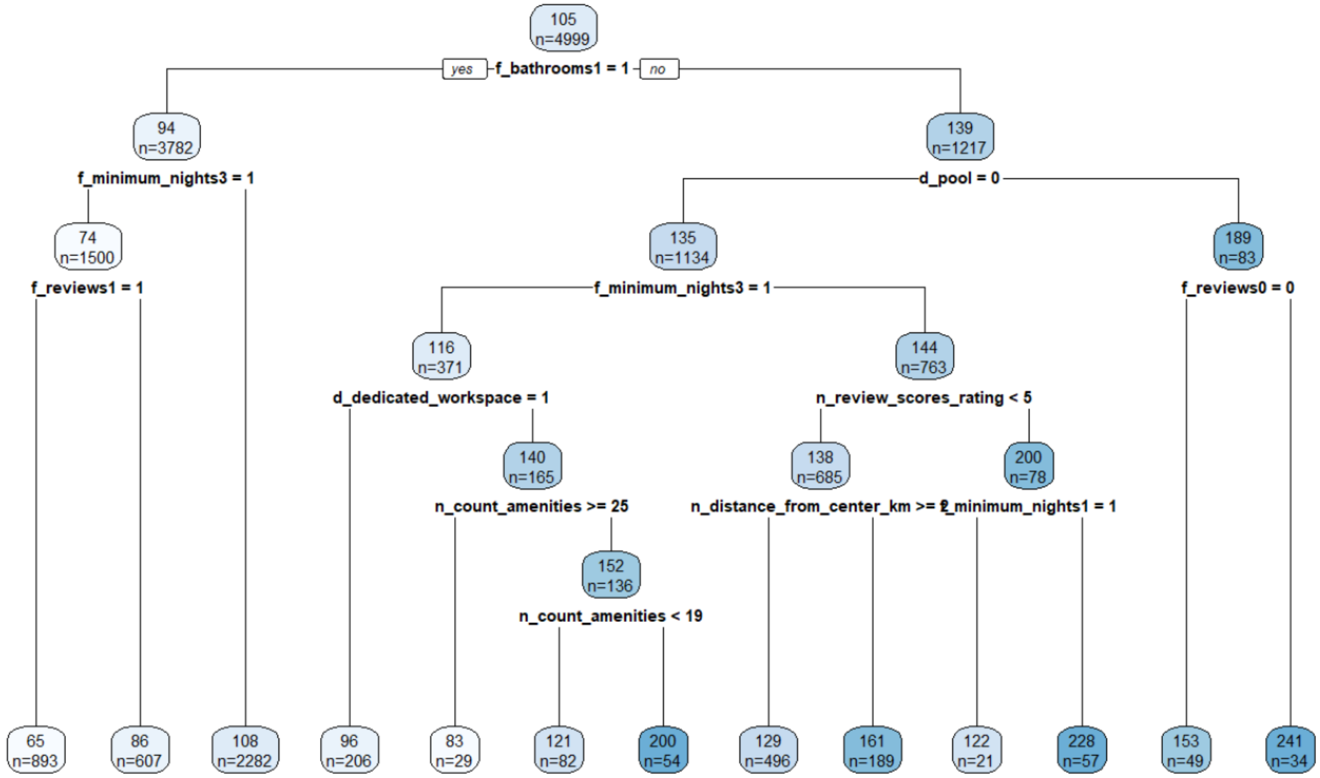
Model	Variables included
Model1	Basic apartment variables
Model2	Basic apartment variables, Additional apartment variables
Model3	Basic apartment variables, Additional apartment variables, Review variables
Model4	Basic apartment variables, Additional apartment variables, Review variables, Amenities variables
Model5	Basic apartment variables, Additional apartment variables, Review variables, Amenities variables, Interaction terms1
Model6	Basic apartment variables, Additional apartment variables, Review variables, Amenities variables, Interaction terms1, Interaction terms2
Model7	Basic apartment variables, Additional apartment variables, Review variables, Amenities variables, Interaction terms1, Interaction terms2, Interaction terms3

A4. OLS estimates



	variable	ols_coefficient
(Intercept)	(Intercept)	113.0497950
f_bathrooms1	f_bathrooms1	-75.1928094
f_bathrooms2	f_bathrooms2	-47.9645450
f_bathrooms3	f_bathrooms3	NA
n_distance_from_center_km	n_distance_from_center_km	-6.5891499
n_beds	n_beds	-0.2562291
n_accommodates	n_accommodates	4.9262986
f_property_type1	f_property_type1	12.7604434
f_property_type2	f_property_type2	5.0852762
f_property_type3	f_property_type3	23.7274979
f_property_type4	f_property_type4	5.8963878
f_property_type5	f_property_type5	NA

A5. Pruned decision tree


105
n=4999
yes
f_bathrooms1 = 1
no
94
n=3782
139
n=1217
f_minimum_nights3 = 1
d_pool = 0
74
n=1500
135
n=1134
189
n=83
f_reviews1 = 1
f_minimum_nights3 = 1
f_reviews0 = 0
116
n=371
144
n=763
d_dedicated_workspace = 1
n_review_scores_rating < 5
140
n=165
138
n=685
200
n=78
n_count_amenities >= 25
n_distance_from_center_km >= 2
f_minimum_nights1 = 1
152
n=136
n_count_amenities < 19
65
n=893
86
n=607
108
n=2282
96
n=206
83
n=29
121
n=82
200
n=54
129
n=496
161
n=189
122
n=21
228
n=57
153
n=49
241
n=34