Shah_business

Shah Ali

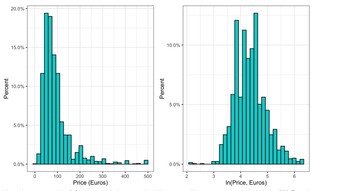
2/10/2022

Executive summary:

In this project I am tasked to help a company which is operating small and mid-size apartments hosting 2-6 guests to set the on optimum rental price for its apartments in the city of **Puglia** in Italy. To achieve this I will build a price prediction model based on data provided by Inside Airbnb which was scraped between December 30, 2021 and December 31, 2021. The prediction was built on several features of the accommodation such as the number of bedrooms, bathrooms, the number of guests it accommodates as well as on information about the host, including the scores of ratings given to apartment. Moreover, on several variables aimed at describing different kinds of amenities offered by the apartment already in the city. I analyzed 4 different kinds of models, to find the best prediction, namely OLS, CART, Random forest and GBM. Out of these the I chose the prediction results of random forest model to be the best one. To see all the codes and files I created for this project please visit my Github

Data Cleaning

The data for the chosen destination was a large dataset with more than 33000 observations and 74 variables. This data required intense cleaning before I could begin our analysis, I performed basic cleaning that involved filtering the property type to the ones of our interest that are apartments so I selected all observation of Entire apt, Entire serviced apartments, Entire Loft and Entire Condominium, and then I filtered the number of guests to 2 to 6. The main part of the cleaning process rested on cleaning the amenities. In our dataset all the amenities were written together in one large vector with total of 1442 unique amenities. Since I am interested in analyzing the effect of each amenity on the target variable I needed to create meaningful dummy variable out of the names of amenities. I separated each amenity from the list, grouped them together if they were similar (Wifi & Internet, TV and cable) and chose only those which had at least 1% of measurable observations. In this way I was able to narrow down the amenities to 85.



Note: Apartments with 2-6 accommodation capacity. Histogram without extreme values (price < 500 Euros)

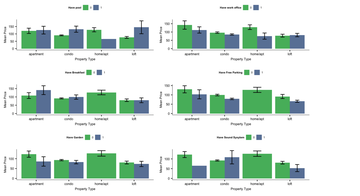
In order to deal with the missing values, I initially dropped those variables that had more than 50% of the missing values and using domain knowledge, I dropped those variables as well which will play no role in our analysis for example picture_url, name, host response rate etc. For the rest I applied imputation. Some of the main variables like number of bathrooms, bedrooms and beds had less than 1% of missing values, for these variables I used logical assumptions while imputing values. For instance, for bathrooms I imputed the median value in place of missing values. For bedrooms I assumed 2 people can share one room thus I imputed the value of reminder of the division by 2. For number of beds I assumed that for 3 people atleast 2 beds are needed. I didn't not create a flag variable as the number of missing values are less, however for variables with more than 30% missing values I created a flag variable with a value of 1 and imputed median value. In this way our dataset was left with no missing values.

During the cleaning of the target variable, price, I dropped all the observations where price was not available as we should impute in our target variable. Since the analysis focused on small to medium size of apartments, I dropped the price above 500 Euros which were less than less than 1% of observations considering them as extreme values. Then the distribution of the price was checked which was found to be bit right skewed due to the presence of high values such as 500 Euros. I decided to proceed with price variable without the log transformation for this analysis. While it is true that that log will bring us closer to normal distribution of the variable, but it will also lead to errors when transformed back to normal prices.

Variables and feature engineering

After data cleaning the variables we are left with are broadly divided as the following.

Numeric variables: These variables define size for example, number of beds, number of accommodates, number of bathrooms and bedrooms, the minimum number of nights required to rent the place and the availability of the apartment in 365 days of the year. *Factor variables:* These are categorical variables that are either in string or numeric form. For example, the neighbourhood the apartments is in or the type of property (apartment, condominium or loft). *Dummy variables:* These are binary variables mostly describing the amenities offered in the property such the Wifi, pool, beachfront facing property, kitchen supplies, elevator etc. *Review variables:* These describe the reviews and ratings characteristics for an individual property. Number of reviews, the rating for those reviews and the mean monthly reviews received. *Host variables:* These variables describe the characteristics of the host of the property. They are binary variables like host is a super-host or if the host is verified



After the variables are set, we need to find the interaction terms between property type and amenities to include in our prediction models. In order to evaluate which interactions terms to use, I logically selected those amenities for our analysis which have an effect on the price. I used the helper functions provided by Bekes & Kézdi to help me plot the histogram which help me with the logical selection. The amenities whose inclusion showed more than 10 Euros of difference on price were included in our model. In this way I was left with 40 important amenities.

The next step was to perform LASSO using all the variables (that is most complex model). The approach that I have used in this analysis is to use LASSO as a variable selection method and use the non-zero predictors given out by the LASSO for all of the subsequent prediction models (OLS, Random Forest, CART and GBM). After running LASSO we are left with 102 predictors which we shall be using for the price prediction models.

Cross validation & Holdout

To improve model performance, I created work set (train and test) sample and the holdout set from the dataset. The value of k was set at 5 which ensured 80% of the 2100 observations for training and 20% for test. The models that I will build later are each trained through 5-fold cross-validation so as to render the least overfitting coefficients, taking the average of each model's 5 test folds, warranting the RMSE and BIC statistics with no assumptions.

OLS Predictive Model

Versions of the Airbnb Apartment Price Prediction Models

Model	Predictor Variables
M1	~ n_accommodates
M2	~ n_accommodates+ basic_lev
M3	~ n_accommodates+ basic_lev + basic_add + reviews + host + dummies
M4	~ n_accommodates+ basic_lev + basic_add + reviews + host + dummies + interaction

I built 4 Models to run my OLS prediction on using different number of predictors that I got from the LASSO. The first model had only one predictor, the n_accommodates variable, model 2 had 5 predictors, model 3 had 67 and model had 102 variables respectively. Model 4 was the overfitted model with all the variables. According to my OLS regression results, model 3 proved to be the best predictive model out of these 4. As can be seen below. Even though we see a higher BIC value in model 3 than in model 2, model 3 has the lowest RMSE in the test set. When BIC and Cross validation produce conflicting results, we should prefer to side with the cross-validation results since it's not based on auxiliary assumptions (BEKES. 2021).

Comparing Model Fit Model Measures

Model	Coefficients	R_squared	BIC	Training_RMSE	Test_RMSE
M1	2	0.031	19665	66.8	66.7
M2	8	0.084	19612	65.0	65.1
M3	71	0.227	19785	59.7	63.0
M4	194	0.294	20545	57.0	70.8
LASSO	92	0.142	NA	NA	62.9

Random Forest

Random Forest is the machine learning predictive algorithm that uses the results of many regression trees. The algorithm creates large number of trees and averages the prediction of them to achieve one average. In this way better prediction is created as opposed to a single model. In our case we used the same holdout and work set and run our model with tuning parameter to allow cross validation to find the tuning parameter value with lowest RMSE value.

The tuning parameter choice is kept same as did in the case by Bekes and Kézdi. Used 3-by-3 for defining mtry tuning parameter and minimum number of observations in each tree split. For our number of variable (mtry) we took 8, 10 , 12 and for the (min. nodes) in terminal nodes we have taken (5,10,15). The result from the cross validated test set RMSE values on the combination is shown below. Based on this regression the lowest RMSE value was 56.2 at the minimum node of 5 and mtry of 12 as shown below.

Random Forest RMSE by tuning parameters

nodes	8	10	12
5	58.2	57.8	57.7
10	58.9	58.7	58.5
15	59.4	59.2	59.0

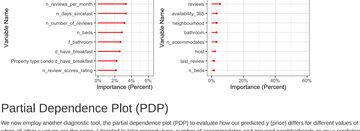
Further I can an auto tune version of random forest allowing algorithm to pick up the tuning parameter itself. The cross-validated RMSE for autotune produced slightly better results with cross-validated RMSE of 57.1 at 5 terminal nodes and mtry of 89 variables.

CART & GBM

I further tested the CART model with pruning to see how my prediction works based on the Test RMSE value. The lowest test RMSE was 62.8, highest so far. This is understandable as the CART created a single tree which tends to overfit the data despite pruning. Running GBM got us close to the lowest Test RMSE (GBM RMSE = 57.2) but autotuned Random Forest is still the winner.

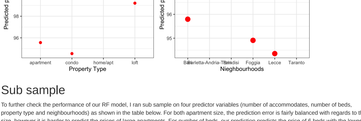
Variable Importance

Variable importance is a helpful diagnostic tool that helps us gain useful insight about how much each variable contributes towards predicting our target variable. The results of variable importance, from both self-tuned and auto-tuned Random Forest were mostly same. The following top 10 important variable plot shows variables which ones had the largest MSE reduction in auto tuned model. It shows that number of days of availability is the most important x predictor with more than 6% importance. In second chart these variables are grouped based on their factor groups, it turns out that amenities account for more than 50% of importance.



Partial Dependence Plot (PDP)

We now employ another diagnostic tool, the partial dependence plot (PDP) to evaluate how our predicted y (price) differs for different values of x when all other x values are the same. I decided to take property type, number of accommodates and grouped neighborhoods as my x predictors. The graphs for each are shown below. A graph for property type shows that entire apartments have the highest average predicted price. In the second graph we see that as the number of accommodates increases, the average prices tend to increase linearly. Third graph shows how average prices vary in different neighborhoods while other predictors are held the same.



Sub sample

To further check the performance of our RF model, I ran sub sample on four predictor variables (number of accommodates, number of beds, property type and neighbourhoods) as shown in the table below. For both apartment size, the prediction error is fairly balanced with regards to the size, however it is harder to predict the prices of large apartments. For number of beds, our prediction predicts the price of 6 beds with the lowest prediction error followed by 2 and 3 beds as the number of observations are high but for others we need to collect more data. Similarly, in sub sample test of neighborhood we see that our model predicts the prices of Barletta-Andria-Trani neighborhood with lowest prediction error, followed by Brindisi which PDP suggest. This is a deviation with PD plot can be accredited to the limitation in data and small number of observations for Barletta-Andria-Trani. Drilling down into the property type, the sub sample supports the result of PD plot. The home/apartment has the lowest prediction error.

Performance across sub-sample

Var.1	RMSE	Mean.price	RMSE.price
Apartment size	NA	NA	NA
large apt	54.6	92.5	0.591
small apt	52.6	88.3	0.596
Beds	NA	NA	NA
2	53.6	94.0	0.571
3	46.5	85.8	0.542
4	59.4	81.9	0.726
5	43.4	91.8	0.473
6	89.9	145.6	0.618
Property Type	NA	NA	NA
apartment	65.9	121.3	0.543
condo	55.6	90.7	0.613
loft	41.1	82.5	0.499

Conclusion

After conducting this analysis, according to the results my best model was the Random Forest model with auto tuning as that gave the lowest cross validated RMSE value. While there were different data preparation steps that had to performed in our data than compared to the 2017 case study on Airbnb in London by Bekes and Kezdi, (owing to the difference in data) the final predictive pricing model was in coherence. The lowest CV RMSE value for Puglia comes out to be 56.2 and while that of London is 44.5 we also need to consider other factors such as inflation rate, country's GDP and tourism in these cities they have an effect on the data and prices. This model predicts the prices of the home/apartment, accommodating at-most 3 guests, in Brindisi neighborhood with lowest prediction error. This model can be applied for predicting prices of apartments with these specifications on live dataset provided there is an high external validity of that dataset with this data.

Horse Race of Models CV RSME

	CV RMSE
OLS	63.0
CART	66.7
Random forest 1: Tuning provided	57.7
Random forest 2: Auto Tuning	57.4
GBM	59.7

Credits: Codes by Bekes and Kézdi the author of DATA ANALYSIS FOR BUSINESS, ECONOMICS, AND POLICY. [CAMBRIDGE UNIV PRESS. served to be extremely for this analysis. They provided the inspiration for starting and completing the project.