Predicting Airbnb prices in Berlin Using Feature Engineering Techniques and Supervised Models

Eliott Barbot, Thomas Brilland, David De La Hera Carretero, Guillaume Dugat
Project Group: TD1
Mention IA, CentraleSupélec, France
https://gitlab-student.centralesupelec.fr/2019brillandt/apprauto-airbnb-berlin

***Abstract*—We present various methods to predict the price of a night's accommodation offered on the short-term rental website Airbnb in the city of Berlin. The models we experimented with are linear regression, bagging, Epsilon Support Vector Regression, Decision trees, random forests, AdaBoost, Gradient Boosted Tree and XGBoost. Those models were applied to different transformations (scenarios) of our data set to see its performance and evaluate the best one. The best results were obtained with AdaBoost and XGBoost, depending on the error metric.**

I. Introduction

The data set employed in the project aims to help us to predict the price of a night's accommodation offered on the website Airbnb in the city of Berlin. It includes a wide range of different features such as its location, number of chambers, number of beds, etc.

The objective of this project is to predict as well as possible the variable objective which is the price of the accommodation by using the different techniques learned in the course of Apprentissage Automatique.

First, we will explain the different approaches we used. We will describe precisely the models and the processing scenarios we implemented. Then, we will provide the results and analyze them. Finally we will conclude and give our best model and scenario.

II. Models and Methods

A. Metrics and Evaluation

For evaluation, we started by splitting our dataset in a training set and validation set : our validation set contains 20% of the data. It allows us to evaluate our model on data it didn't encounter during training.

The metrics we chose to use to evaluate our models are Root Mean Square Error (RMSE) but also Mean Absolute Error. In order to see how the model performs in general, we also used a metric that corresponds to the errors for 0.1, 0.25, 0.5, 0.75 and 0.9 quantiles (for example the 0.1 quantile error means 10% of our model predictions are better).

B. Baseline Preprocessing

The first step in order to apply the models into our data was to preprocess the data as it was neither proper nor useful in the way it was given to us. In order to accomplish that, we developed a pipeline to carry on the task.

First, we kept the features that were useful among all the features that were given in the raw data, such as number of rooms, number of bathrooms, latitude, longitude... In total we kept fifteen features to work with. This step was also conditioned by the lack of data in some of the features, which led us to the decision of deleting some of the features. For instance, we had to delete the host response time or the square feet (as it was lacking in 98% of the instances) although the latter would have provided really useful information to our project. Next, we had to handle the remaining missing values for the columns we kept. For this purpose, we did listwise deletion. Beyond that, we also remove the instances whose price was above 250€, since they appeared as outliers and could disturb the fitting of the models.

Once we had data without lacking values, we started to transform the features such as dates or nominal features into a way that our models could handle. For the dates, we decided to transform it into a numeric value by comparing it to a reference fixed by us, allowing us to have the number of days up to that reference as output of the transformation. To transform the nominal data, such as the neighbourhood group, the property type and the room type, we transformed them into dummy variables so our models could work with them (one-hot encoding). This second transformation lead to an increase of our number of features.

Next, we transformed the boolean features with true and false into numeric boolean values with 0's and 1's, so our model could work with them.

For the location (latitude and longitude), we decided to add as features of our data set the distance of every instance to three places of interest of Berlin (chosen arbitrarily): Alexanderplatz, Brandebourg and Gendarmenmarkt, as we thought that the result would be more relevant to our models than the location itself, although we left them among our features in case they were relevant to our models (for example we can suppose there are differences between east Berlin and west Berlin).

Finally, we transformed some string data representing numbers into its correct type, as it would have caused some problems in the modeling otherwise.

Once we had all the data properly preprocessed we added a possible rescaling of the features with mean zero and variance one (standardization) since it can help the training for some models.

C. *Baseline Approach*

We wanted to compare the results of our models to a simple baseline, so we also tried to predict the prices with simple models : a random predictor, a mean regressor and a nearest neighbor predictor. The goal is to verify that the complex models that we train are more efficient than these simple models.

We implemented the random predictor so that it matches the prices repartition inside our dataset: for example, 36% of instances have a price between 20 and 40, so our random predictor will return 36% of instances in this interval. The mean estimator systematically predicts the mean value of the training dataset. The so-called nearest neighbor predictor computes and returns the average price among the 30 closest neighbors in terms of location (for this purpose we implement a KNN on the latitude and longitude). Those baselines help us have an idea of scores for 'stupid models' so we can compare them to our trained models.

D. *Supervised Models*

To predict the price of a night, we tried to train different models that we studied in the course, as well as some learned by ourselves :

- Linear regression
- Bagging
- Epsilon-Support Vector Regression
- Decision trees and random forests
- Boosting : AdaBoost, Gradient Boosted Tree and XG Boost

E. *Scenarios*

After applying our preprocessed data to the different models developed, we decided to compare the results with other scenarios where we had chosen a reduced set of characteristics or a transformation of the features' space.

The first scenario we planned to implement was one where we included a reduced set of features of our data set. In order to achieve that, we wanted to create an algorithm that evaluated (by a cross validation on the train dataset) the best model for every combination of our features. As the number of features was quite big, it was not feasible to implement that algorithm, so we decided to implement a simplified version that enabled us to execute the code without a server: the stepwise method. We implemented the forward and backward version of that algorithm, so we had two different scenarios to compare.

The second scenario we decided to implement was one with the Principal Component Analysis (PCA) technique, so we would search for the principal components of our dataset, allowing us to reduce the number of variables used by our model by only choosing the most relevant components.

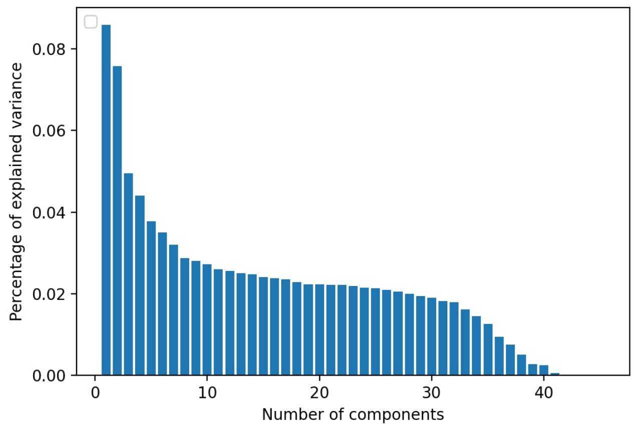

Figure 1. Variance explained with each principal component

As a third scenario, we decided to implement also the Partial Least Squares (PLS) technique, as it might be possible that with the PCA technique we would not find the suitable directions to our variable objective: the price.

III. Results

A. *Scenario selection*

With the stepwise scenarios we realized that both of the algorithms arrived to almost the same reduced data set, but we kept the two scenarios to compare its performances.

In the PCA scenario we had to chose the number of principal components we wanted to use depending on the variance explained by the model. To help us to chose the number of components we decided to plot the variance explained by each component as we can see in Figure 1. We decided to choose 7 components as it was the number where we could find the inflexion point of the graphic. We decided to use the same number of components to the PLS scenario.

B. *Model Results*

In the tables below, we can observe that our models perform slightly better than the baselines, with both metrics. However, between the models and scenarios, there are no major differences in the results that allow us to clearly claim that a model or a scenario is better than the others. We can note that the LinearRegression, the simplest model in the list, obtains comparable results to those of the more complicated ones.

We can still select the best models: if we want to minimize the RMSE, we can use the Ada Boost Model using the PLS algorithm ; if we want to minimize the Mean Absolute Error, we can use the XG Boost with the Backward selection. With the quantile evaluation, we

observe that we have :
10% of predictions are better than an error of 2.7€.
50% of predictions are better than en error of 13.6€.

For all models, we can see the results in Table I and Table II (see next page).

IV. Conclusion

Globally, our developed models are better than the baselines approaches. However, there are no significant differences among the different scenarios and models.

The best root mean square error was obtained by the model ADABoost in the PLS scenario, with 39,76. The best mean absolute error was obtained by the model XGBoost in the backward selection scenario, with 20,20.

Models	Baseline	PCA	PLS	Forward Selection	Backward Selection
Random prediction (baseline)	62.29	-	-	-	-
Mean prediction (baseline)	50.18	-	-	-	-
Nearest Neighbors (baseline)	49.27	-	-	-	-
Linear Regression	39.93	41.34	39.94	39.92	39.92
SVR	41.42	41.57	40.78	41.60	41.43
Decision Tree	41.10	41.03	40.31	40.57	40.23
Random Forest	40.12	40.57	39.98	39.98	40.02
Bagging	40.11	40.60	39.94	40.27	40.30
Ada Boost	40.89	41.04	**39.76**	40.35	40.39
Gradient Boosted Tree	41.01	41.23	40.31	39.87	40.01
XG Boost	41.59	42.46	41.36	41.57	41.46

Table I
COMPARISON OF THE ROOT MEAN SQUARE ERROR OF DIFFERENT MODELS WITH DIFFERENT PREPROCESSING SCENARIOS

Models	Baseline	PCA	PLS	Forward Selection	Backward Selection
Random prediction (baseline)	40.28	-	-	-	-
Mean prediction (baseline)	29.62	-	-	-	-
Nearest Neighbors (baseline)	28.74	-	-	-	-
Linear Regression	20.78	21.59	20.81	20.77	20.75
SVR	20.54	20.44	20.50	20.22	20.54
Decision Tree	21.18	21.31	21.15	21.15	20.84
Random Forest	20.39	20.85	20.82	20.38	20.42
Bagging	20.61	21.33	20.81	20.82	20.80
Ada Boost	20.88	21.22	21.22	21.26	21.23
Gradient Boosted Tree	20.32	21.31	20.41	20.39	20.37
XG Boost	20.27	20.81	20.52	20.37	**20.20**

Table II
COMPARISON OF THE MEAN ABSOLUTE ERROR OF DIFFERENT MODELS WITH DIFFERENT PREPROCESSING SCENARIOS