Airbnb Price prediction and Sentiment Analysis

Final Project Report

Machine Learning I: Large Scale Data Analysis and Decision Making

(MATH80629A.H2021)

Students: Amirhossein Hajigholam Saryazdi, Fatemeh Zand

Winter 2021

Contents

Introduction

Airbnb is one of the top platforms in providing accommodation and has always been competing with other residential units like hotels. Price is one of the most important factors affecting customers' decision when booking a place at Airbnb. Lower prices increase chances of bookings but on the other hand, it is important to consider the fact that it should not be too low to negatively affect the profit making of the hosts. Thus, balanced and fair pricing plays an important role in this industry, to have satisfied hosts and customers and consequently have positive impacts on tourism and the e-commerce.

There are numerous factors affecting the pricing of an Airbnb. One of the most important factors is the sentiment of the reviews the listings in Airbnb receive. It is indispensable that reviews play an important role in wider public opinion and people pay attention to the online reviews when making a purchase, booking a place to stay, etc. Moreover, the extreme sentiment in the reviews (whether it is positive or negative) spreads very quickly and affects decision behavior of the customers and the price. So, it is important to consider the effects of the sentiments of the reviews in price prediction models.

In this project, we have built several price predictions models for the listings in Airbnb and we have also represented a comparison between different models' performances. The input we have used for our models is Toronto Airbnb data from Inside Airbnb, that provides open source and non-commercial data about Airbnb listings usage in many cities all around the world. The dataset contains numerical and text data about the available listings, and we have extracted more than 20 features that are the most important and representative of our prediction task. To create the price prediction models, we have developed linear regression, ridge and Lasso regression, support vector regression (SVR), random forest regression, XGBoost model and a neural network model. Furthermore, we study how sentiment analysis of guests' reviews improves the prediction compared to when review score rating is used in the prediction models. At the final step, we explore the effects of combining two different datasets to see if it improves the model results when it is tested on a completely new city in comparison with a model which is trained on a single dataset.

Relevant Work

The literature focusing on price prediction can be categorized into two streams. The first focuses on rental/purchase price prediction of non-shared residential units like homes. examples can be seen in the studies by (Ma et al., 2018; Yu and Wu, 2016) where methods like linear regression, random forest and SVR are used and a comparison of their performances for prediction is also represented.

Another stream of literature focuses on the price prediction of shared residential units and is relevant to our study. Using quantile regression and least squares methods, Wang and Nicolau (2017) analyzed the price determinants of the Airbnb listings. In a similar context, Masiero et al. (2015) studied the relation between holiday homes and their prices using a quantile regression model. Again, through using linear regression, Teubner et al. (2017) analyses reputation related features for price prediction of the shared units. To create a price prediction model for Airbnb in different cities, Li et al. (2016) used a clustering method called Multi-Scale Affinity Propagation. As a clustering feature, they used the distance to the city landmarks in their model then used linear regression to obtain the clusters for different prices.

For the first time, Kalehbasti et al. (2019) considers sentiment of the reviews as a feature for their dataset and performs several prediction methods to create price prediction models. Cai et al. also consider Airbnb price prediction using regression models and the novelty of their work is to directly consider text data (from the reviews) as input variable for their models.

Without transforming the problem into a classification problem or performing evaluations on logarithmic scale, our project also aims to work on the original price regression problem. The novelties of our project is that particularly, we aim to answer to the following three questions:

- ❑ What is the best regression model for price prediction of Toronto Airbnb listings?
- ❑ Does Sentiment Analysis of the reviews really help for a better price prediction?
- ❑ Can further improvements be made in price prediction of an unseen dataset by building a generalizable model on datasets of two or more different cities instead of a single city?

Data Description and Preprocessing

Eliminating Irrelevant and Uncorrelated Features

The dataset that we have chosen consists of 21618 Airbnb locations and has 106 features. Some of these features are irrelevant or cannot be used for our price prediction task. So, we eliminate features like “host name”, “picture url”, or “summary” that cannot be incorporated into our model. Moreover, there are some features that their information is also represented in other features like “availability_30”, “availability_60”, “availability_90” and “availability_365”. Here we only keep the last one and remove all other features. Consequently, after the first round of evaluation, 26 features remain. At the second steps, we dropped two numeric features based on low correlation with the price.

Handling Missing Values

Features like “host acceptance rate”, “square feet”, “weekly price”, “'host response time'”, and “host response rate” are going to be dropped because of their high number of missing values. This will not affect the model since there are other features for evaluating the same concepts. For the Nan's in the two features “Security deposit” and “Cleaning fee”, it is likely that the hosts do not have a security deposit or charge any extra cleaning fee as it is also seen in practice. So, we will simply replace the null values with 0.00 for these two features.

For all the features that have less than 1000 missing values (less than 0.05 of our data), we drop the missing value (NaN) entries. Examples of these features are “host is superhost”,

“host total listings count.”, “host identity verified.”, “zip code”, and “bathrooms”. There are also 3968 missing values in 'review scores rating' and there are 3702 listings with 0 reviews. It seems that for the cases where there are no reviews, the review score rating is set equal to NaN. Here we replace all NaN values that have no review with 'No Reviews'.

Feature engineering

One approach is to use “latitude” and “longitude” of the location of the Airbnb residential units and calculate their distances to downtown Toronto. But we believe that the proximity to downtown is not necessarily a good measure and prefer to use zip codes instead. So, we will only keep the first three letters of zip codes, as a neighborhood indicator.

Also, for the “review scores rating” we convert them into buckets, categorizing them based on their range of score rating. So, considering ranges of [0-9], [10-19], ..., [95-100] and the “no reviews” category, we will have 13 categories for this feature.

Exploratory Data Analysis

In order to provide further insights into our data, a series of explanatory plots are generated from our dataset. Figure 1(a) shows the frequency of the price in our Airbnb listings. As can be seen, most of the prices are below $2000 per night.

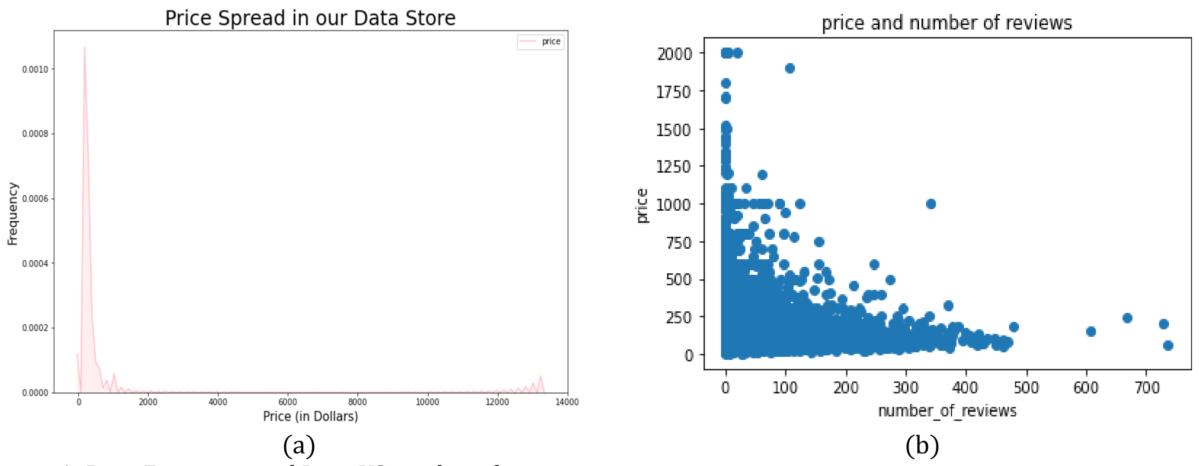
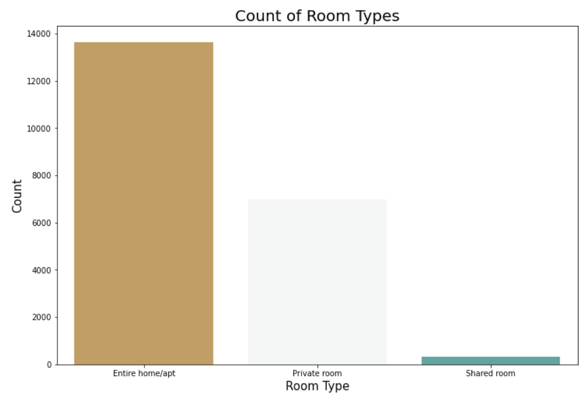


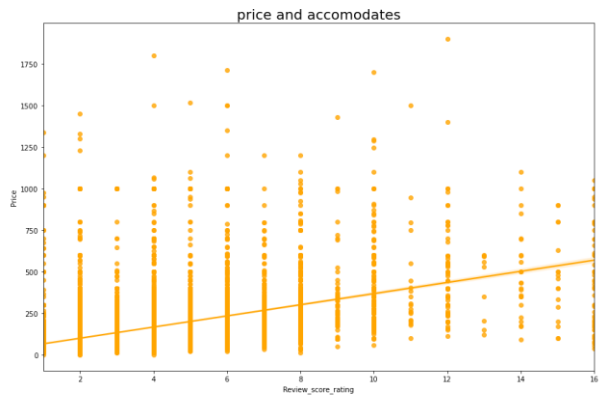
Figure 1- Price Frequency and Price VS number of reviews

The Figure 1(b) shows the relation between number of the reviews and the price. It can be concluded that properties with higher prices have lower reviews possibly because of the fact that they are visited less often and as the price decreases, the review count will increase. Moreover, the majority of the properties have review count below 500 if we do not consider the outliers.

Count of the room types are shown in Figure 2(a). It can be seen that “Entire home/apt” is the most common room type in our listing. Figure 2(b) is shown to illustrate the relation between price and accommodates. As expected, there is an upward trend in price as accommodates increases.



(a)



(b)

Figure 2 - Count of Room Types and Price VS Accommodates

It can be concluded from Figure 3 that the properties that have availability are mostly prices below $1000. The high priced properties can be places that become available seasonally like boutiques or part-time rentals.

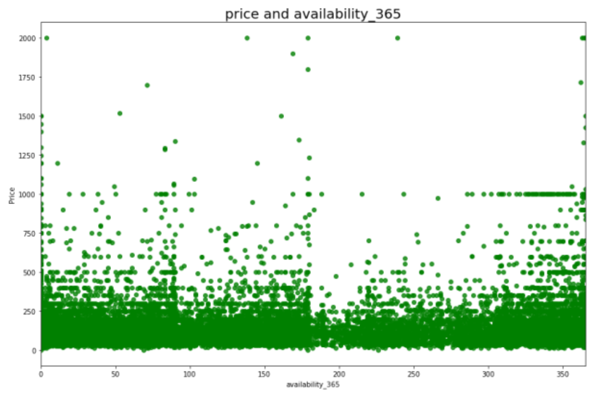


Figure 3 - Price VS Availability 365

Outliers and Duplicate Data

Based on the explanatory analysis results, we dropped values in price that are greater than 2000 to eliminate outliers from our dataset. To further improve the performance of our model, we do an outlier treatment using the Interquartile Range (IQR) approach. Finally, there are18 features as shown in the Table 1.

Table 1 - Variable Types

Variables					
Price	Accommodates	Property type	Cancellation policy	Guests included	Instant bookable
Host is super host	Bathrooms	Room type	availability_365	Cleaning fee	Host identity verified
Host total listings count	Bed type	Review scores rating	Extra people	Zip code	Security deposit

After completely preprocessing the data and categorizing features of type object, all our 19 features will be either of types int64, float64, or uint8.

Model Development

Since we are going to build price prediction algorithms, "price" will be our dependent variable and the rest of the variables are our independent variables. We create two different datasets namely Y and X for our dependent and independent variables respectively.

Splitting and Scaling the Data

To bring all the numeric values of our dataset to a common scale, in this step, we standardize our dataset. First, we separate features with binary values from the rest as we do not need to standardize them. Next, we import the preprocessing library to standardize our feature variables. Finally, we concatenate binary variables back to the standardized data. We can now run the price prediction models on our standardized and non-standardized dataset.

Linear Regression Model

As baseline, we used linear regression model that by fitting a linear regression equation, attempts to model the relationship between two variables in our dataset. In this case, our dependent variable is "Price" and our independent variables are all of the features that we have extracted.

Ridge and Lasso Regression Models

We also used ridge and Lasso regression models on our dataset which are extensions of linear regression and are basically a regularized linear regression model. The loss function of the Ridge Regression Model is: $L = \sum(\hat{Y}_i - Y_i)^2 + \lambda \sum \beta^2$

Where $\hat{Y}_i$ is our prediction, Y_i is the actual data and β is the estimation of the coefficients. λ is also a coefficient for controlling the penalty term. According to this formula, $\lambda \sum \beta^2$ means that high values of β results in punishing of the loss function. The loss function of the Lasso Regression model as the following and its difference is in the regularization term which is an absolute value: $L = \sum(\hat{Y}_i - Y_i)^2 + \lambda \sum|\beta|$

Support Vector Regression

SVR is a linear regression that can be used to solve both linear and non-linear problems and basically aims to find a line or hyperbole with which the data is separated into classes. In this project, we use an SVR with Linear Kernel model on our dataset.

XGBoost Model

Next, we run the XGBoost model from xgb package on our dataset. This ensemble method is a regularized and optimize version of the decision-tree gradient boosting algorithm and since its release about 2015 has dominated structured or tabular datasets. Following hyperparameters of this model are tuned through a grid search and cross-validation:

- Learning rate =Rate at which our model learns patterns in data. After every round, it shrinks the feature weights to reach the best optimum => 0.1
- n estimators = Number of trees one wants to build =>500
- Max depth = Determines how deeply each tree is allowed to grow during any boosting round. => 5
- Colsample bytree = Percentage of features used per tree. => 0.5
- Gamma = Specifies the minimum loss reduction required to make a split. => 0.1

Figure 4 shows the actual prices (x-axis) against our best predictions (y-axis) by XGBoost model which is our best performing model. The slope of the fitting line is lower than 1 and this shows that our best model tends to underestimate the price of listings.

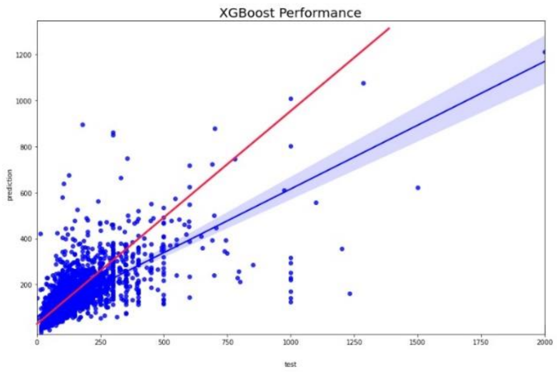


Figure 4 - XGBoost Performance

Random Forest Model

Here we use RandomForestRegressor function of sklearn.ensemble. At training time of this method, multitude of decision trees are constructed which are then used for regression/classification tasks on our test data. hyperparameters are tuned using a grid search and cross-validation. Some of the important hyper parameters to tune in Random Forest are:

- n estimators = number of trees in the forest => 200
- Max features = max number of features considered for splitting a node => sqrt
- Max depth = max number of levels in each decision tree => 60
- Min samples split = min number of data points in a node before its split => 4
- Min samples leaf = min number of data points allowed in a leaf node => default
- Bootstrap = method for sampling data points (with or without replacement) => default

Neural Network

We adopted 3 fully connected layers in the multilayers perceptron (MLP). “adam” optimizer is used for the model and in terms of activation functions, “Relu” is used for hidden layers and a linear activation function for the output layer, as it is being used for a regression task. For the same reason, the loss function selected to be mean-squared-error. For the first run, epochs, batch size and other hyperparameters are selected based on (Kalehbasti et al., 2019), which are as follows:

- Units in the first hidden layer: =20
- Units in the second hidden layer = 5
- Learning rate = 0.001
- Decay rate = 0.0001
- Epochs = 500
- Batch size = 256

As hyperparameter tuning with grid search is not feasible for this model, different values for hyperparameters are tried manually, for example adding a hidden layer, increasing nodes, etc., but they do not improve model performance significantly.

Comparison of the results

Table 2 shows RMSE, MAE and R^2 for all the models we used for price prediction of Toronto Airbnb listings. As it is mentioned in the "Splitting and Scaling data" section, we also tried models with standardized dataset, but this practice increases or does not change errors of models. So, for the purpose of brevity, we are not going to report those results.

Table 2- Model Performance

	RMSE	MAE	R^2
Linear Regression	90.9466	49.4948	40.57
Ridge CV Model	91.0204	49.5267	40.47
Lasso CV Model	90.9611	49.4486	40.55
SVM	94.36	43.5	45.15
Random Forest*	83.87	43.43	0.49
XGBoost	**83.85**	**42.3**	0.49
Neural Network	87	44.61	0.45

*As for the Decision-tree-based methods no scaling is required, the results of standardized data are not provided.

Sentiment Analysis

We did sentiment analysis of listing reviews using *sentiment* function of TextBlob library. This function returns polarity and subjectivity. Polarity is a float lying between –1 (negative) and 1 (positive). In this section, first we calculate the average sentiment polarity of each place which lies between –0.6 and 1. To capture the nuance difference between these sentiments, we bucketize this range with 0.2 interval and a make a new feature. That is [-0.6, -0.4], …, [0.8,1] and at the end encode these categories at the time of modeling. Next, we remove the "review sore rating" feature from our dataset and will add this new self-defined feature to the dataset. We run our best performing price prediction model "XGBoost" on this newly defined dataset and calculate RMSE, MSE, and R^2. Results and comparisons are provided in the Table 3.

Table 3 - Sentiment Analysis Effect

	RMSE	MAE
Model with review score rating	80	42.3
Model with sentiment analysis	**78**	**41.89**

Building a Generalized Model

One of our goals in this project is to investigate if training a model on multiple datasets would perform better in predicting the price in an unseen and new Airbnb listing dataset. To this aim, we trained XGBoost once on the whole Toronto listing, and once on the whole combination of Toronto and New York listings, then test them separately on Berlin data set as the unseen data. Comparing the errors from the table 4 shows combining datasets would not necessarily result in a model with lower error compared to when it is trained on a single dataset. It is worth mentioning that we also tried these with Neural Networks model but did not get a better result than XGBoost.

Table 4 - Generalized Model Results

	RMSE	MAE	R2
Train on Toronto	59.0	39.03	0.01
Train on Toronto and NY	67.0	49.81	-0.25

Conclusion

The XGBoost model achieved the best fit and highest accuracy among all other models. Adding the sentiment of the reviews as a feature to our dataset improved the accuracy of the XGBoost model. Moreover, results showed that combining datasets from different cities does not necessarily improve the price prediction task for a new city compared to when models are trained on a single dataset. In the future, it would be beneficial to compare other cities, apply other models with the same or different features, and try other sentiment analysis algorithms on reviews.

Revision

As discussed during the class presentation, Table 5 represents test set variance (theoretical minimum MSE) and error when using train set mean as a prediction to have two base lines and better sense of the models' error. The result shows we are far better than the baselines.

Table 5 - Revision results

	MSE	RMSE	MAE	R2
Train set mean as a prediction	13918.53	118.0	72.78	0
Test set variance	13946.55	118.1	-	-
XGBoost	6447.65	80.0	42.3	0.49

We also advised to compare the results of the second part with the situation in which there is no feature about reviews, that is running the model without “review score ratings” and reviews sentiment analysis. In this case RMSE and MAE are 81 and 42.65 respectively. Comparing them with the Table 3, not considering a reviews-related feature leads to higher error.

References

Cai, T., Han, K., & Wu, H. Melbourne Airbnb Price Prediction.

Kalehbasti, P. R., Nikolenko, L., & Rezaei, H. (2019). Airbnb price prediction using machine learning and sentiment analysis. arXiv preprint arXiv:1907.12665.

Li, Y., Pan, Q., Yang, T., & Guo, L. (2016). Reasonable price recommendation on Airbnb using Multi-Scale clustering. Paper presented at the 2016 35th Chinese Control Conference (CCC).

Ma, Y., Zhang, Z., Ihler, A., & Pan, B. (2018). Estimating warehouse rental price using machine learning techniques. International Journal of Computers Communications & Control, 13(2), 235-250.

Masiero, L., Nicolau, J. L., & Law, R. (2015). A demand-driven analysis of tourist accommodation price: A quantile regression of room bookings. International Journal of Hospitality Management, 50, 1-8.

Teubner, T., Hawlitschek, F., & Dann, D. (2017). Price determinants on Airbnb: How reputation pays off in the sharing economy. Journal of Self-Governance and Management Economics, 5(4), 53-80.

Wang, D., & Nicolau, J. L. (2017). Price determinants of sharing economy based accommodation rental: A study of listings from 33 cities on Airbnb. com. International Journal of Hospitality Management, 62, 120-131.

Yu, H., & Wu, J. (2016). Real estate price prediction with regression and classification. CS229 (Machine Learning) Final Project Reports.